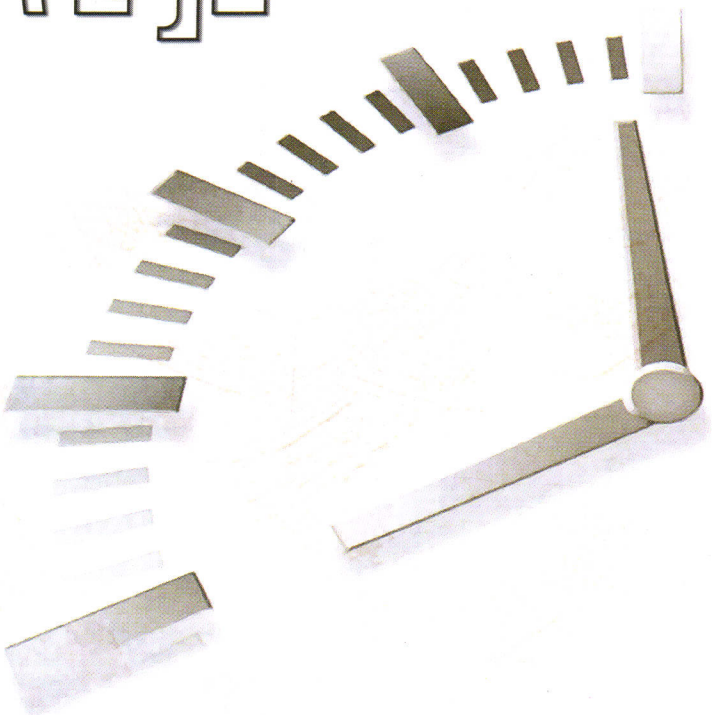
RWTUV
TÜV CERT EN ISO 9001
دارای گواهینامه ایزو ۹۰۰۱
در زمینه نشر کتاب و طراحی جلد

مؤسسه فرهنگی هنری
دیباگران تهران

خودآموز PHP در ۲۴ ساعت

(ویرایش دوم)



در این کتاب می‌آموزید:

- نصب و راه‌اندازی PHP
- بررسی جزئیات زبان PHP
- بررسی فرمها و فایل‌ها
- دست‌یابی به بانک اطلاعاتی MySQL
- برنامه‌نویسی وب
- دنباله‌های کاراکتری و عبارت منظم
- ثبت وضعیت جلسات و کوکی‌ها
- PHP و XML
- موتور الگوسازی Smarty

مترجمان: مهندس علی ناصح
محمد ناصح

به نام خدا

مؤسسه فرهنگی هنری
دیباگران تهران

خودآموز PHP

در ۲۴ ساعت

مترجمان
مهندس علی ناصح و محمد ناصح

دارنده گواهینامه ISO 9001/2000
در زمینه نشر کتاب و طراحی جلد

Sams Teach Yourself PHP In 24 Hours
خودآموز PHP در ۲۴ ساعت

مترجمان : مهندس علی ناصح - محمد ناصح

ناشر : مؤسسه فرهنگی هنری دیباگران تهران

حروفچینی و صفحه‌آرایی : مجتمع فنی تهران

طرح روی جلد : مجتمع فنی تهران

چاپ : پیام حق

نوبت چاپ : اول

تاریخ نشر : شهریور ماه ۱۳۸۳

تیراژ : ۳۰۰۰ نسخه

قیمت : ۵۷۰۰۰ ریال

شابک : ۵- ۴۲۶-۳۵۴-۹۶۴

ISBN : 964-354-426-5

زاندسترا ، مت Zandstra , Matt

خودآموز PHP [پی.اچ.پی] در ۲۴ ساعت/نوشته مت زاندسترا؛ مترجمان علی ناصح، محمد ناصح .--تهران: مؤسسه فرهنگی هنری دیباگران تهران، ۱۳۸۳.

۶۸۸ ص.:مصور، جدول.

ISBN 964-354-426-5

فهرستنویسی بر اساس اطلاعات فیپا.

عنوان اصلی: Sams Teach Yourself PHP In 24 Hours, 3rd ed. , c2004 .

۱.پی.اچ.پی(زبان برنامه‌نویسی کامپیوتر). ۲.وب--سایتها--طراحی.الف. ناصح، علی، ۱۳۵۲- ،مترجم.ب.ناصح،محمد،مترجم.ج.عنوان.

QA۷۶/۷۳/پ۸۶ز۲ ۰۰۵/۱۳۳

۱۳۸۳

کتابخانه ملی ایران ۸۳-۸۰۶۸م

آدرس: سعادت آباد، میدان کاج . خ سرو شرقی. روبه‌روی خ علامه . ساختمان شماره ۹۷

تلفن : ۷- ۲۰۹۸۴۴۶ صندوق پستی: ۹۴۳/ ۱۴۳۳۵

فهرست مندرجات در یک نگاه

فهرست مندرجات

« بخش دوم: اجزای زبان PHP»

«بخش چهارم: توسعه PHP»

مقدمه ناشر

حمد و سپاس ایزد منان را که با الطاف بیکران خود این توفیق را به ما ارزانی داشت تا بتوانیم در راه ارتقای دانش عمومی و فرهنگ این مرز و بوم در زمینه چاپ و نشر کتب علمی دانشگاهی، علوم پایه و پزشکی به ویژه علوم کامپیوتر و انفورماتیک گامهایی هر چند کوچک برداشته و در انجام رسالتی که بر عهده داریم مؤثر واقع شویم. گستردگی علوم و توسعه روزافزون آن، شرایطی را به وجود آورده که هر روز شاهد تحولات اساسی چشمگیر در سطح جهان هستیم. این گسترش و توسعه نیاز به منابع مختلف از جمله کتاب را به عنوان قدیمی‌ترین و راحت‌ترین راه دستیابی به اطلاعات و اطلاع‌رسانی، بیش از پیش روشن می‌نماید. در این راستا ، واحد انتشارات مؤسسه فرهنگی هنری دیباگران با همکاری جمعی از اساتید، مؤلفان، مترجمان، متخصصان، پژوهشگران، محققان و نیز پرسنل ورزیده و ماهر در زمینه امور نشر درصدد هستند تا با تلاشهای مستمر خود برای رفع کمبودها و نیازهای موجود، منابعی پربار، معتبر و با کیفیت مناسب در اختیار علاقه‌مندان قرار دهند.

کتابی که در دست دارید با همت "مهندس علی ناصح و محمد ناصح" و تلاش جمعی از همکاران انتشارات میسر گشته و شایسته است از یکایک این گرامیان تشکر و قدردانی کنیم.

ویراستاری: خانم هما تیموری

ویرایش و صفحه‌آرایی کامپیوتری : خانمها ملوک احمدسلطانی و معصومه باقری

طراحی روی جلد : خانم محبوبه توکلی

امور چاپ و نشر : آقای حیدر شفیعی

ناظر چاپ : آقای کریم براغ

در خاتمه از عموم هموطنان عزیز و دانش پژوهان گرامی خواهشمندیم ما را با ارائه پیشنهادها و انتقادهای خود در بهبود کمی و کیفی کارهای انجام شده راهنمایی نمایند تا بتوانیم در آینده کتابهایی با کیفیت بهتر تقدیم حضورشان کنیم.

مدیر انتشارات

مؤسسه فرهنگی هنری دیباگران تهران

درباره مؤلف

Matt Zandstra (به نشانی matt @ corrosive. Co. uk)، یک مشاور فنی در زمینه اطلاع‌رسانی است. وی به‌همراه شریک تجاری خود، آقای Max Guglielmino مشغول اداره شرکتی با نام Corrosive Web Design (به آدرس اینترنتی http://www. corrosive. Co. uk) می‌باشد که در زمینه قالب‌بندی اطلاعات، تسهیل استفاده و بهره‌برداری از اطلاعات و ایجاد محیطهای عملیاتی و اطلاعاتی به صورت تخصصی فعالیت می‌کند. پیش از تألیف ویرایش دوم کتاب حاضر، Matt مشغول طراحی زبان اسکریپت جدیدی مبتنی بر XML و Java به‌همراه مفسر مربوطه، به منظور استخراج محتویات مورد نیاز از درون صفحات وب بود.

وی در حال حاضر مشغول بررسی‌هایی در زمینه طراحی الگوها، تستهای واحد و برنامه نویسی است. با این که نسبت به گذشته دچار اضافه وزن شده است، اما دوچرخه سوار قابلی است. طبق گفته خودش مشغول نوشتن یک رمان است که البته به درازا کشیده است. هم‌اکنون Matt به‌همراه همسر خود Louise McDougall و دختر کوچکش Holly در شهر Brighton انگلستان زندگی می‌کند.

Matt این کتاب را به پدر خود تقدیم می‌کند.

قدردانی

از Louise بابت همراهی ما از لندن به برایتون حین نوشتن این کتاب تشکر می‌کنم. پس از مراجعت چنین به نظر می‌رسید که میزکارم در اتاق زیباتری جای گرفته است. هم‌چنین از وی به‌خاطر همراهی مثال زدنی‌اش قدردانی می‌کنم. همین‌طور از Holly کوچولو به علت نحوه‌ای که جمله " OH NO! Dropped it! "را به زبان می‌آورد. همه اینها باعث شد تا نوشتن این کتاب به سادگی میسر شود.

بار دیگر از Scott Meyers ، Jill Hayden و Andrei Zmievski از انتشارات Sams قدردانی و تشکر می‌کنم.

مثل همیشه از همکارم Max Guglielmino بابت زحماتی که در نبود من کشیده و به نوعی موجبات و تسهیلات لازم جهت تکمیل کتاب را فراهم نموده، متشکرم.

در آخر از اعضای گروه Citi pagas (به نشانی اینترنتی (http: // www. citi pages. Net خصوصاًCharlie، James، Jim، Tolan و Rares بابت کمکهایشان قدردانی می‌کنم.

Matt Zandstra

نظرات خود را به ما بگویید!

به عنوان خواننده کتاب، شما مهم‌ترین مخاطب ما هستید و ما ارزش فوق‌العاده‌ای برای نظرات شما قائل هستیم. نظرات شما درمورد میزان درستی نحوه انجام کارما، چگونگی بهبود کیفیت کار، زمینه‌های مورد علاقه شما و کلاً هر گونه پیشنهادی در جهت اهدافی که دنبال می‌کنیم، ارزش زیادی برایمان دارند. در صورت تمایل می‌توانید نظرات و پیشنهادات خود درمورد این کتاب و هر گونه پیشنهادی را در جهت بهبود کیفیت کتابها، به دو طریق پست الکترونیکی یا پست معمولی برای ما ارسال کنید.

لطفاً به این نکته توجه کنید که به هیچ عنوان به نامه‌هایی که درمورد مسائل فنی کتاب درخواست کمک یا راهنمایی بیشتر دارند، پاسخ نخواهیم داد. همچنین توجه شما را به این نکته جلب می‌کنیم که به‌علت حجم نامه‌های دریافتی از پاسخ به تمامی آنها معذوریم.

لطفاً در متن نامه خود به عنوان کتاب مورد نظر و نام مؤلف آن اشاره نموده و شماره تلفن یا فکس خود را ذکرکنید؛ در این‌صورت پیشنهاد شما به‌دقت بررسی شده و با مؤلف و ویراستاران مربوطه در میان گذاشته خواهد شد.

آدرس پست الکترونیکی: webdev @samspublishing. Com

آدرس پستی: Mark Taber

Associate Publisher
Sams Publishing
201 West 103rd street
Indianapolis , IN 46290 USA

مقدمه

کتاب حاضر در مورد PHP است. یک زبان اسکریپت نویسی کد باز برای وب که مدت زمان زیادی نیست که به لیست زبانهایی چون ASP, Perl و Java ، یعنی زبانهایی که جهت ایجاد محیطهای پویایonline مورد استفاده قرار می‌گیرند، پیوسته‌است. دراین کتاب مبحث برنامه‌نویسی نیز مورد بررسی قرار می‌گیرد. با حجمی که پیش از تألیف برای کتاب در نظر گرفته شده بود، بدیهی است که امکان تدوین یک راهنمای کامل برنامه‌نویسی با PHP یا پرداختن به کلیه توابع و تکنیکهای قابل استفاده درPHP وجود نداشت. با این همه، در صورتی‌که شما یک برنامه‌نویس قهار بوده و قصد فراگیریPHP را دارید یا به‌تازگی کار اسکریپت(برنامه)نویسی را آغاز کرده‌اید، بخشهای مختلفی که در این کتاب در نظر گرفته‌ایم، می‌تواند نقطه مناسبی برای شروع برنامه‌نویسی شمابا PHP تلقی‌شود، چرا که به اندازه کافی اطلاعات مورد نیاز را در اختیارتان قرار می‌دهد.

مخاطبین کتاب

این کتاب از همان ابتدا شما را با دانش عملی مناسبی که جهت برنامه نویسی با زبان PHP4 بدان‌نیاز دارید، آشنا می‌کند. در نوشتن این کتاب فرض بر این بوده که خواننده از پیش هیچ‌گونه تجربه برنامه نویسی ندارد، با این همه چنان‌چه پیشتر با یکی از زبانهای برنامه‌نویسی مثل C یا Perl برنامه‌نویسی را تجربه کرده باشید، مطالعه این کتاب را بیش از اندازه ساده و روان خواهید یافت.

PHP4 یک زبان برنامه‌نویسی برای وب است. برای اینکه حداکثر بهره را از این کتاب ببرید لازم است تا با مفاهیم مربوط به وب (و یا به‌طور کامل وب جهانی یا World Wide Web) به ویژه زبان نشانه‌گذاری HTML آشنا باشید. در صورتی‌که این اولین برخورد شما با مفاهیم ذکرشده است، باز هم می‌توانید از مطالب موجود در متن این کتاب بهره‌مند شوید. با این همه لازم است تا ابتدا مروری هر چند سریع و کوتاه بر HTML و مفاهیم مربوطه داشته باشید، اما چنان‌چه در حال حاضر توان ایجاد اسناد ساده وب را داشته و با ایجاد جداول ساده در HTML مشکلی ندارید، نیازی به این کار نیست.

PHP4 به‌گونه‌ای طراحی شده است که به‌خوبی توانایی کار با بانکهای اطلاعاتی را داشته باشد. دراین‌راستا برخی از مثالهای کتاب جهت بهره‌مندی از بانکهای اطلاعاتی ایجاد شده توسط MySQL نوشته شده‌اند. MySQL یک سیستم مدیریت بانک اطلاعاتی است که استفاده از آن بر روی برخی از محیطهای عامل بدون پرداخت کمترین هزینه‌ای ممکن می‌باشد. ما در این کتاب مقدمه کوتاهی درباره SQL ذکر کرده‌ایم، اما اگر قصد استفاده جدی از PHP جهت کار با بانکهای اطلاعاتی را دارید، توصیه می‌کنیم که مدت زمانی را صرف مطالعه در زمینه بانکهای اطلاعاتی و SQL نمایید. در این مورد منابع

مقدمات مناسبی را بر روی اینترنت خواهید یافت. اگر بانک اطلاعاتی مورد نظرتان بانکی غیر از MySQL ، است خواهید دید که به‌راحتی با تغییر کمی که بر روی مثالهای مربوطه در کتاب خواهید داد، می‌توانیدتوابع PHP معادل را جهت ایجاد پرس و جوها و بهره‌مندی از بانک اطلاعاتی مورد نظرتان ایجاد کنید.

سازمان کتاب

این کتاب به چهار بخش زیر تقسیم شده است:

- بخش اول مقدمه‌ای بر PHP4 است.
- بخش دوم ویژگی‌های اصلی زبان PHP4 را تشریح می‌کند. در صورتی‌که به‌تازگی به جرگه برنامه‌نویسان پیوسته‌اید، لازم است تا توجه ویژه‌ای به این بخش داشته باشید.
- بخش سوم PHP4 را با شرح بیشتری نسبت به بخشهای قبلی مورد بررسی قرار می‌دهد. توجه‌این‌بخش بر روی توابع و تکنیکهایی است که برای تمامی برنامه‌نویسان PHP از اهمیت بالایی برخوردارند.
- و بالاخره بخش چهارم که شامل بررسی کدهای کتابخانه‌ای PHP4 می‌باشد. در این بخش هم در مورد کدهایی که خودتان تولید می‌کنید و هم درباره کدهای آماده‌ای که از طریق سایر تولیدکنندگان عرضه می‌شوند، بحث خواهیم کرد.

بخش اول کتاب ساعتهای اول تا سوم را شامل می‌شود. در این بخش سعی شده اطلاعاتی در اختیارتان قرار بگیرد تا به‌وسیله آن بتوانید اولین اسکریپت (برنامه) PHP4 خود را ایجاد و اجرا نمایید:

- ساعت اول با عنوان " PHP : از صفحات خانگی تا سایت‌های پُرتال" به تشریح تاریخچه و نیز قابلیتهای PHP پرداخته و دلایلی را مورد بررسی قرار می‌دهد که بر مبنای آنها برنامه‌نویسان بسیاری PHP را به‌عنوان زبان برنامه‌نویسی منتخب مورد توجه قرار داده‌اند.
- ساعت دوم با عنوان " نصب PHP بر روی کامپیوتر" به بحث درباره چگونگی نصب PHP بر روی یک سیستم UNIX پرداخته و برخی از گزینه‌های مربوط به پیکربندی را که نحوه کامپایل‌کردن برنامه‌های PHP را تحت تاثیر قرار می‌دهند، بررسی می‌کند. در این ساعت همچنین به روشهای پیکربندی PHP پس از نصب آن بر روی کامپیوتر توجه شده است.
- ساعت سوم با عنوان " اولین برنامه اسکریپت" به روشهای مختلف تعبیه برنامه‌های PHP در اسناد وب می‌پردازد. در این ساعت چگونگی نوشتن متون بر روی صفحه مرورگر کاربر نیز مورد بررسی قرار گرفته است.

بخش‌دوم‌کتاب‌ساعتهای‌چهارم تا هشتم را شامل می‌شود. در این بخش، خواننده با اجزای اصلی زبان برنامه نویسی PHP4 آشنا می‌گردد:

- ساعت‌چهارم‌با عنوان " اجزای سازنده زبان PHP " به تشریح اجزای مختلف این زبان‌می‌پردازد. در این ساعت با مفاهیم متغیر، انواع داده‌ها، عملگرها و عبارات آشنا خواهید شد.
- ساعت پنجم با عنوان " اجزای کنترلی برنامه" به نحوه روند کنترل اجرای برنامه می‌پردازد. در این ساعت علاوه بر دستورالعملهای کنترلی if و switch درباره چگونگی بهره‌گیری از ساختارهای حلقه با استفاده از عبارات for و while مطالب ارزنده‌ای فرا می‌گیرید.
- ساعت ششم با عنوان " توابع" به‌چگونگی استفاده از توابع به‌منظور سازماندهی هرچه بیشتر برنامه‌های PHP می‌پردازد.
- در ساعت هفتم با عنوان " آرایه‌ها " درباره ساختمان داده ویژه‌ای با عنوان آرایه که قادر به نگهداری اطلاعات است، به بحث خواهیم نشست. در این ساعت برخی از توابعی را که PHP4 به منظور کار با آرایه‌ها ارائه کرده، بررسی خواهیم نمود.
- ساعت هشتم با عنوان " اشیا " به پشتیبانی PHP از مفاهیم شئ و کلاس اختصاص دارد. طی این ساعت به توسعه یک برنامه جالب با استفاده از مطالبی که تا بدین ساعت آموخته‌اید، خواهید پرداخت.

بخش سوم کتاب شامل ساعتهای نهم تا بیست و دوم است. در این بخش از کتاب با ویژگی‌ها و تکنیکهای مختلف زبان برنامه‌نویسی PHP آشنا می‌شوید:

- ساعت نهم با عنوان " بهره‌گیری از فرمها " به بحث در مورد چگونگی دریافت ورودی کاربران از طریق مکانیزمی که فرمهای HTML در اختیارمان قرار داده، اختصاص یافته است. در این ساعت چگونگی جمع‌آوری اطلاعات مورد نیاز از کاربران را می‌آموزید.
- درساعت‌دهم‌با عنوان" بهره‌گیری از فایل‌ها " به بررسی چگونگی کار با فایل‌ها و فهرستهای موجود در کامپیوتر خواهیم پرداخت.
- در ساعت یازدهم با عنوان " بهره‌گیری از توابع DBA " به بحث در مورد پشتیبانی PHP از سیستمهای بانکی شبه DBM که نسخه‌هایی از آنها تحت بسیاری از محیطهای عامل یافت می‌شوند، خواهیم پرداخت.
- ساعت دوازدهم با عنوان " ارتباط با بانکهای اطلاعاتی ـ بهره‌گیری از MySQL " ضمن ارائه یک بررسی مقدماتی درباره دستور زبان SQL به معرفی توابعی از PHP می‌پردازد که جهت بهره‌گیری از بانکهای اطلاعاتی MySQL به آنها نیاز دارید.
- ساعت‌سیزدهم‌باعنوان"بررسی سمت سرور" به بررسی برخی جزئیات مربوط به درخواستهای HTTP و همچنین توابع مربوط به برنامه‌نویسی شبکه در PHP می‌پردازد.

- در ساعت چهاردهم با عنوان " گرافیک و PHP " به بررسی توابعی در PHP می‌پردازیم که در رابطه‌باگرافیک و پردازش تصویر ارائه شده‌اند. به کمک این توابع به راحتی می‌توانید تصاویری از نوع GIF یا PNG را به‌طور پویا ایجاد نمایید.
- ساعت پانزدهم با عنوان " بهره‌گیری از تاریخ و زمان" به بررسی توابع و تکنیکهای مفیدی درباره محاسبات زمان می‌پردازد. در این ساعت چگونگی ایجاد تقویم را با استفاده از یک برنامه PHP نمونه نشان داده‌ایم.
- ساعت شانزدهم با عنوان " بهره‌گیری از انواع داده‌ها " بحث مجددی را درباره انواع داده‌ها به پیش می‌کشد و ضمن بحث در مورد توابعی که می‌توان جهت کار با داده‌های مختلف در برنامه‌های اسکریپت مورد استفاده قرار داد، توابع بیشتری را نیز در رابطه با آرایه‌ها معرفی می‌کند.
- ساعت هفدهم با عنوان " بهره‌گیری از رشته‌های کاراکتری " به بررسی توابعی درباره پردازش رشته‌های کاراکتری در برنامه‌های PHP می‌پردازد.
- ساعت هجدهم با عنوان " بهره‌گیری از عبارات منظم " به بررسی توابعی در مورد عبارات منظم اختصاص دارد. به کمک مطالبی که در این ساعت فرا می‌گیرید، قادر خواهید بود تا به جست وجو و جایگزینی الگوهای پیچیده در رشته‌های کاراکتری بپردازید.
- در ساعت نوزدهم با عنوان " ثبت وضعیت با استفاده از کوکی‌ها و رشته‌های پرس و جو" به بررسی روشهایی جهت ارسال اطلاعات مابین برنامه‌های اسکریپت و همچنین درخواستها می‌پردازیم.
- ساعت بیستم با عنوان " ثبت وضعیت با استفاده از توابع مربوط به ثبت جلسات " به توسعه و گسترش روشهای مورد بحث در ساعت نوزدهم می‌پردازد. در این ساعت با توابع سیستمی PHP که در رابطه با ثبت جلسات ارائه شده‌اند، آشنا می‌شوید.
- در ساعت بیست و یکم با عنوان " بهره‌گیری از محیط سرور" چگونگی فراخوانی برنامه‌های خارجی از درون برنامه‌های اسکریپت و بهره‌مندی از خروجی آنها را فرا می‌گیرید.
- ساعت بیست و دوم با عنوان" PHP4 و XML " به بحث درباره پشتیبانی PHP از زبان XML (Extensible Markup Language) می‌پردازد. در این ساعت به بحث در مورد توابعی از PHP که در رابطه با تجزیه اسناد XML ارائه شده‌اند، خواهیم پرداخت. همچنین ویژگی‌هایی را در رابطه با بررسی اسناد XML و تبدیل آنها به سایر قالبها مورد بحث و بررسی قرار خواهیم داد.

بخش چهارم کتاب شامل ساعتهای بیست و سوم و بیست و چهارم است. در طی این دو ساعت سعی می‌کنیم تا از آنچه که تا بدین جا آموخته‌ایم، استفاده نموده و کتابخانه‌هایی را که می‌توان جهت افزایش قابلیتها و عملکردهای PHP از آنها بهره گرفت، تجربه کنیم:

- ساعت‌بیست‌وسوم با عنوان " بهره‌گیری از الگوها (الگو سازی) ـ Smarty " به معرفی کتابخانه ویژه‌ای می‌پردازد که به‌واسطه آن برنامه نویسان می‌توانند منطق برنامه را از بخشهای مربوط به نمایش جدا کنند.
- ساعت بیست و چهارم با عنوان " مثالی از یک کتابخانه کاربردی: Page. inc. php " به مراحل ساخت یک کتابخانه می‌پردازد. این کتابخانه جهت کمک به انجام کارهای متداولی که حین ایجاد سایتهای پویا با آنها مواجه می‌شوید، ایجاد می‌گردد و در انتها به منظور حفاظت از کلمات عبور و همچنین کنترل دسترسی توسعه می‌یابد.

« بخش اول: شروع»

- ساعت اول: PHP : از صفحات خانگی تا سایت‌های پُرتال
- ساعت دوم: نصب PHP بر روی کامپیوتر
- ساعت سوم: اولین برنامه اسکریپت

ساعت اول

PHP : از صفحات خانگی تا سایت‌های پُرتال

به برنامه‌نویسی با PHP خوش آمدید! حین مطالعه این کتاب تقریباً با تمامی عناصر زبان برنامه نویسی PHP آشنا می‌شوید. اما پیش از این‌آشنایی‌مروری بر PHP به‌عنوان یک محصول خواهیم داشت. به‌عبارت دیگر به بحث و بررسی در مورد تاریخچه، ویژگی‌ها و آتیه این زبان همه‌گیر می‌پردازیم.
سرفصلهای این ساعت به قرار زیر است:

- **PHP چیست؟**
- **تاریخچه‌ای از PHP**
- **چه بهبودهایی در مورد PHP4 نسبت به نسخه‌های پیشین صورت گرفته است.**
- **برخی گزینه‌هایی که می‌توانند ویژگی‌های مفیدی را به برنامه‌های PHP اضافه کنند.**
- **برخی از دلایل قانع‌کننده در انتخاب PHP جهت توسعه برنامه‌ها**

PHP چیست؟

PHP یک زبان برنامه نویسی است که آوازه آن با سرعت زیادی در همه محافل برنامه نویسی پیچید. این زبان در اصل به عنوان مجموعه‌ای از ماکروها (برنامه‌های آماده) جهت کمک به افرادی که مشغول طراحی صفحات وب بودند، ایجاد شد. ولی به‌زودی اهداف بزرگ‌تر و مهم‌تری از صفحات خانگی را نشان کرد. از آن زمان تا به حال قابلیتها و توانایی‌های PHP توسعه زیادی یافته است به‌گونه‌ای که از یک‌سری ابزارها و برنامه‌های سودمند به یک زبان برنامه‌نویسی کامل، قوی و کارآمد تبدیل شده و هم‌اینک می‌توان توسط آن به مدیریت سایت‌های بسیار بزرگی که حجم وسیعی از داده‌ها را در بانکهای اطلاعاتی نگهداری می‌کنند، پرداخت.

عمومیت و مقبولیت این زبان نیز به موازات رشد توانایی‌ها و قابلیتهای آن افزایش پیداکرد. طبق آمار منتشر شده در سایت NetCraft به آدرس اینترنتی http:// www. netcraft. com طراحی و برنامه‌نویسی بیش از یک میلیون وب سایت مختلف تا ماه نوامبر ۱۹۹۹ توسط زبان PHP صورت گرفته است‌واین رقم تا ماه سپتامبر ۲۰۰۱ به رقم شش میلیون سیر صعودی خود را طی کرده است. همچنین طبق آمار منتشر شده در وب سایت http: // www. securityspace. com زبان PHP مهم‌ترین ماجول قابل استفاده بر روی وب سرور Apache است که بدین ترتیب از ماجول بسیار محبوب mod _ ssl نیز پیشی گرفته است.

عنوان رسمی PHP عبارتست از PHP: Hypertext Preprocessor (لازم به توضیح است که برای حاصل شدن نام PHP می‌بایست عنوان Hypertext Preprocessor را به صورت بازگشتی، یعنی به شکل ... Hypertext Preprocessor Hyper text تلفظ کنید ـ مترجم). این زبان یک زبان اسکریپت نویسی برای سمت سرور است که معمولاً خود در متن یک سند HTML نوشته می‌شود. برخلاف صفحات HTML معمولی، اسکریپت‌های PHP توسط سرور به سمت کلاینت ارسال نمی‌شوند بلکه توسط برنامه ویژه‌ای موسوم به موتور PHP مورد تجزیه قرار می‌گیرند. طی این فرآیند عناصر HTML از برنامه اسکریپت نوشته شده با PHP حذف شده و کد PHP باقیمانده پس از ترجمه اجرا می‌گردد. برنامه‌های PHP قابلیتهای بسیار متعدد و متنوعی همچون پرس وجو از بانکهای اطلاعاتی، ایجاد تصاویر گرافیکی، نوشتن و خواندن فایل‌ها، محاوره با سرویس دهنده‌های راه دور و بسیاری از قابلیتهای دیگر را شامل می‌شوند، از این‌رو تقریباً نمی‌توان محدودیتی را برای این عملکردها و قابلیتها متصور شد. پس از اجرای کد PHP خروجی برنامه با عناصر HTML ترکیب شده و نتیجه برای مشاهده کاربر به سمت کلاینت گسیل می‌شود.

نیازهایی که PHP می‌تواند برآورده کند

اسکریپت‌نویسی‌وزبانهای برنامه‌نویسی اسکریپت به موازات وب رشد کرده و توسعه پیدا کرده‌اند. از آنجا که نیاز به ایجاد سایت‌هایی با محتوای پویا و غیر استاتیک در سالهای اخیر بیش از گذشته احساس می‌شود بنابراین بیش از همیشه به روشها، شیوه‌ها و ابزارهایی جهت توسعه سریع و کارآمد محیطهای online نیازمندیم. با وجودی که زبان برنامه‌نویسی C را می‌توان یک ابزار کارآمد در رابطه با توسعه سریع ابزارهای سمت سرور به‌حساب آورد اما بهره‌گیری از این زبان به‌سادگی امکان‌پذیر نبوده و علاوه‌بر این در صورتی که توجه خاصی حین توسعه برنامه‌های مربوطه با این زبان برنامه‌نویسی محبوب صورت نگیرد، به‌سادگی می‌توان شکافهای امنیتی خطرناکی را انتظار داشت. از سویی دیگر، زبان Perl یک زبان برنامه‌نویسی است که در اصل جهت پردازش رشته‌های کاراکتری طراحی شده و طبیعتاً خلأ موجود در رابطه با فقدان محیطهای پویای وب را پر کرده است. زبان Perl در این رابطه از امنیت بیشتری نسبت به زبان برنامه نویسی C برخوردار بوده و کارایی پایین‌تر آن نسبت به زبان مذکور، همواره با سرعت بالای توسعه آن جبران شده است. علاوه بر این تعداد کتابخانه‌های قابل توجهی که روز به روز نیز بر تعداد آنها اضافه می‌شود، از دیگر نقاط قوت زبان Perl محسوب می‌شود. اما در این بین زبان PHP در کدام جایگاه قرار دارد؟ زبان PHP به‌طور ویژه برای توسعه برنامه‌های وب ایجاد شده است. از این‌رو مسائل و مشکلاتی که اغلب برنامه‌نویسان وب را مشغول به خود می‌کند از طریق خود زبان حل و فصل شده‌اند. به دو نمونه از این مسائل توجه کنید:

- برنامه‌نویسان Perl جهت دستیابی به داده‌ها و اطلاعاتی که کاربران وب در فرم‌های مختلف وارد می‌کنند، همواره نیازمند استفاده از یک کتابخانه خارجی بوده و یا مجبورند تا کدهای مورد نیاز را خود تولید نمایند. این در حالی است که PHP چنین اطلاعاتی را به طور خودکار در اختیار برنامه‌نویس قرار داده است.
- برنامه‌نویسان Perl جهت نوشتن برنامه‌هایی که نیازمند دستیابی به اطلاعات موجود در بانکهای اطلاعاتی هستند، به‌ناچار باید ماجول‌هایی را که چنین امکاناتی در اختیارشان قرار می‌دهند برروی کامپیوترشان نصب کنند، حال آنکه PHP ذاتاً مجموعه کاملی از بانک‌های اطلاعاتی مطرح را مورد پشتیبانی قرار می‌دهد. (البته برای کسب این پشتیبانی کامل لازم است تا هنگام نصب PHP اطلاعات مورد نیاز در این زمینه را دراختیار برنامه نصب PHP قرار دهید). اسامی این بانکهای اطلاعاتی به قرار زیر است: dBASE?? ، DBM ، FilePro ، Hyperwave ، In formix?? ، InterBase ، Microsoft SQL Server ، mSQL ، MySQL ، ODBC (شامل Access)، Oracle ، Oracle8 ، Postgre SQL و Sybase .

به طور خلاصه، به‌دلیل آنکه PHP در اصل برای برنامه نویسان وب طراحی شده است، تقریباً برای هر نوع مسأله‌ای که برنامه‌نویسان وب با آن مواجه می‌شوند، از مدیریت جلسات کاربران گرفته تا

بهره‌گیری از اسناد XML ، به مجموعه‌ای از توابع و ابزارهای مفید مجهز شده است.

وجود چنین امکانات و تسهیلاتی در مورد PHP به احتمال قوی، این سوال را در ذهن خواننده به‌وجود می‌آورد که آیا این امکانات به بهای از دست دادن ***کارایی***[1] به‌دست آمده‌اند؟ پاسخ منفی است. سرعت اجرای برنامه‌های PHP بر روی وب سرور به‌طور شگفت آوری بالاست. آن‌چنانکه سرعت اجرای آن از برنامه‌های CGI نوشته شده با زبان Perl نیز بیشتر است.

اولین ویرایش کتاب حاضر مقارن با زمانی بود که PHP4 هنوز مراحل پایانی توسعه خود را طی می‌کرد. در حال حاضر این زبان محبوب مراحل تکاملی خود را با موفقیت پشت سر گذاشته و به‌عنوان محیطی امن و پایدار در دنیای برنامه نویسی وب به رشد خود ادامه می‌دهد ضمن اینکه روز به روز نیز بر شمارگان برنامه‌نویسانی که از آن استفاده می‌کنند، افزوده می‌شود.

ویژگی‌های جدید در PHP4

PHP4 چندین‌ویژگی‌جدیدمعرفی‌کرده است که کار برنامه‌نویسان وب را از چندین جهت ساده‌تر و جالب توجه‌تر می‌کند. در زیر به چند نمونه از این ویژگی‌ها اشاره می‌کنیم:

- عبارت foreach جدید که مشابه نمونه خود در زبان Perl است، بیش از گذشته کار پردازش عناصر آرایه‌ها را ساده‌تر کرده است. ما در بسیاری از مثالهای مربوط به آرایه‌ها در این کتاب از این عبارت جدید استفاده خواهیم کرد. در رابطه با پردازش آرایه‌ها نیز هم‌اکنون PHP4 توابعی را معرفی کرده که کار بر روی آرایه‌ها را به آسانی ممکن می‌سازد.
- نوع boolean به عنوان نوع داده‌ای جدید معرفی شده است.
- یکی از ویژگی‌های قابل توجه PHP3 قابلیت نام‌گذاری عناصر فرمها بود، به گونه‌ای که برنامه‌نویس می‌توانست آن عناصر را جزء یک آرایه فرض کند. این بدان معنی است که برنامه‌نویس قادر بود تا اسامی و مقادیر عناصر مذکور را در قالب یک آرایه در اختیار برنامه قرار دهد. این ویژگی هم‌اکنون در PHP4 به آرایه‌های چند بعدی توسعه و تعمیم پیدا کرده است.
- پشتیبانی از مشخصه‌های شئ‌گرایی و برنامه‌نویسی شئ‌گرا همواره یکی از نقاط قوت زبان برنامه‌نویسی PHP بوده است. این ویژگی نیز در PHP4 به‌طور قابل ملاحظه‌ای توسعه پیدا کرده است. برای نمونه در حال حاضر فراخوانی متدهای رونویسی شده کلاس پدر در درون کلاس فرزند، به‌سادگی امکان‌پذیر می‌باشد.

۱- در دنیای برنامه نویسی، واژه کارایی (Performance) اغلب به معنی سرعت است. ـ مترجم

- PHP4 به واسطه بهره‌گیری از کوکی‌ها و رشته‌های پرس و جو امکان پشتیبانی از جلسات کاربران‌رادراختیاربرنامه‌نویسان‌قرار داده است. برنامه‌نویسان اکنون می‌توانند با ثبت متغیرهای مربوط به جلسات مختلف به هنگام ایجاد جلسات، امکان دستیابی به مقادیر آنها در جلسات آینده را از طریق اسامی این متغیرها به‌راحتی فراهم کنند.
- PHP4 شامل عملگر مقایسه‌ای جدیدی با عنوان = = = است که مقایسه نوع داده دو متغیر را به خوبی مقایسه مقادیرشان ممکن می‌سازد.
- آرایه‌های انجمنی جدید در PHP4 که شامل متغیرهای سرور و متغیرهای محیطی می‌باشند، به‌همراه‌متغیری‌که‌اطلاعاتی‌را درمورد فایل‌های بارگذاری شده در خود ذخیره می‌کند، امکانات جالب توجهی را در اختیار برنامه‌نویسان قرار داده است.
- پشتیبانی‌ذاتی PHP4 از Java و XML نکته بارز دیگری است که بدون شک به تعداد کاربران PHP خواهد افزود.

با وجود این‌که ویژگی‌های مذکور به‌همراه سایر ویژگی‌هایی که در این‌جا ذکر نکردیم به‌طور قابل ملاحظه‌ای موجب بهبود در عملکرد زبان برنامه‌نویسی PHP شده‌اند، شاید مهم‌ترین تغییری که در PHP4 صورت گرفته، بهبود عملکرد آن از نظر کارایی باشد که به‌واسطه تغییرات داده شده در کد منبع PHP حاصل شده است.

Zend Engine چیست؟

هنگامی که زبان PHP3 توسعه پیدا کرد، چنین اظهار شد که بخش تجزیه کننده جملات برنامه‌های نوشته شده (که Parser نامیده می‌شود) کاملاً بازنویسی شده‌است. همین داستان در مورد بخشی از زبان PHP4 با عنوان موتور اسکریپت نویسی (scripting engine) نیز تکرار شد. بازنویسی کامل بخش ذکر شده در زبان PHP4 دهها مرتبه اهمیت به‌مراتب بیشتری نسبت به تغییر صورت گرفته در PHP3 دارد.

Zend یک موتور اسکریپت‌نویسی بسیار پیشرفته است که ماجول‌های ویژه PHP بر روی آن سوار می‌شوند. این موتور اسکریپت‌نویسی از نقطه‌نظر کارایی بسیار قابل توجه است.

تغییرات ذکر شده بدون شک موجبات موفقیتهای پی‌درپی PHP را به‌عنوان یک زبان برنامه‌نویسی تضمین خواهند نمود. بیشتر برنامه‌های نوشته شده با استفاده از PHP3 بدون کوچک‌ترین تغییری قابل اجرا و بهره‌برداری می‌باشند، با این تفاوت که سرعت اجرای آنها اکنون به بیش از ۲۰۰ مرتبه افزایش یافته است !

ضمیمه تجاری (غیر رایگان) اضافه شده به موتور اسکریپت‌نویسی Zend ،موجب تسهیلاتی در ترجمه کدهای نوشته شده با PHP خواهد شد. این ضمیمه با آرایه کارایی فوق‌العاده‌اش اگر موجب کنارزدن رقیبان PHP نشود رقابت را برای آنها به زندگی تلخی مبدل خواهد کرد.

Zend با هدف بهبود و افزایش کارایی اسکریپت‌ها طراحی شده است اما قابلیت انعطاف چیز دیگری است که Zend آن‌را به‌خوبی مورد توجه قرار داده و برقراری ارتباط با سرور بهبود پیدا کرده است، لذا ایجاد ماجول‌های PHP جهت بهره‌گیری از طیف گسترده‌تری از سرورها ممکن شده است. برخلاف یک مفسر CGI که محل استقرار آن خارج از وب سرور بوده و با هر بار اجرای اسکریپت مقداردهی می‌شود، Zend به‌عنوان ماجولی از وب سرور به‌همراه سایر ماجول‌های مورد نیاز وب سرور جهت اجرا وارد عمل می‌شود.

بدین ترتیب کارایی به میزان قابل توجهی افزایش پیدا می‌کند چرا که جهت اجرای یک برنامه PHP نیازی به راه‌اندازی موتور اسکریپت‌نویسی Zend نمی‌باشد.

دلایل انتخاب PHP

برای کار با PHP4 به‌عنوان زبان برنامه‌نویسی وب (اسکریپت‌نویسی) دلایل منطقی و محکمی را می‌توان برشمرد. در مورد برخی از پروژه‌ها خواهید دید که فرآیند تولید برنامه نسبت به زمانی که از زبان اسکریپت دیگری استفاده می‌کنید با سرعتی بیش از آنچه که انتظار دارید، پیش می‌رود. از طرف دیگر PHP4 به‌عنوان یک محصول نرم‌افزاری کدباز (Open Source) توسط یک تیم نرم‌افزاری حرفه‌ای به‌خوبی پشتیبانی شده و از سوی جامعه کاربران (برنامه‌نویسان) کاملاً پذیرفته شده است. گذشته از این، PHP بر روی تمامی سیستمهای عامل و اغلب وب سرورها قابل اجرا و بهره‌برداری می‌باشد.

سرعت در توسعه

ازآنجا که PHP امکان جداسازی کد HTML از عناصر اسکریپت‌نویسی را در اختیار برنامه‌نویس قرار می‌دهد، می‌توان کاهش قابل توجهی را در زمان توسعه بسیاری از پروژه‌ها انتظار داشت. به‌عبارت بهتر، برنامه‌نویس PHP قادر است تا مرحله کدنویسی پروژه مربوطه را از مرحله طراحی جدا کند. این گونه تسهیلات علاوه بر ساده کردن بسیاری از کارهای برنامه‌نویسان به‌میزان قابل توجهی از موانعی که بر سر راه یک طراحی موثر و منعطف وجود دارد، می‌کاهد (در دنیای برنامه‌نویسی جداسازی کد برنامه از رابط کاربر دارای اهمیت بسیار زیادی است که در جای خود مورد بحث و بررسی قرار خواهد گرفت ـ مترجم).

ویژگی کد باز (Open Source)

برای بسیاری از مردم اصطلاح کدباز به‌معنی " رایگان " می‌باشد، که البته این به‌خودی خود یک‌مزیت مهم تلقی می‌شود. با این همه پروژه‌های کدبازی که به خوبی هدایت و نگهداری شوند مزایای مهم دیگری را نیز شامل می‌شوند.

یکی از مهم‌ترین مزایای چنین پروژه‌هایی این است که کاربران به مجموعه عظیمی از تجربیات ارزنده و بسیار قابل اعتماد دسترسی دارند. از آنجا که کاربران این‌گونه پروژه‌ها مسائل و مشکلات خود را با یکدیگر در میان می‌گذارند، احتمالاً می‌توانید پاسخ مسائل خود را به آسانی و به سرعت و تنها با کمی همت و جستجو در میان سؤالات از پیش مطرح شده بیابید. در صورتی که چنین اتفاقی نیفتد با ارسال سؤال یا پرسشهای مورد نظرتان به لیستهای پستی مربوطه می‌توانید مطمئن باشید که به‌زودی پاسخ صحیح و قابل اعتمادی دریافت خواهید نمود.

یکی دیگر از ویژگی‌های پروژه‌های کدباز این واقعیت خوشایند است که کاربران همواره از آخرین اشکالات و راه‌حلهای مربوطه آگاهی پیداکرده و از طرف دیگر ویژگی‌های جدیدی نیز مطابق با اعلام نیاز کاربران به پروژه مورد نظر افزوده می‌شود. برای بهره‌بردن از این ویژگی‌های جدید نیازی نیست که کاربران به انتظار انتشار نسخه تجاری بعدی محصول مورد نظر بنشینند، یعنی مسأله‌ای که در مورد سیستمهای تجاری شاهد آن هستیم (که البته این خود نیز با احتمال همراه است).

معمولاً در رابطه با این گونه پروژه‌ها هیچ‌گونه مزیت قطعی در استفاده از سرور یا سیستم عامل مشخصی وجود ندارد. به‌بیان دیگر کاربران آزادند تا مطابق با نیازهای خود یا استفاده‌کنندگان نهایی از محصول تولید شده گزینه‌های مورد نظرشان را انتخاب کنند با این اطمینان که برنامه‌های تولیدشده در هر محیطی قابل اجرا می‌باشد.

کارآیی

PHP4 به‌واسطه موتور پرقدرت و کارآمد خود یعنی Zend توان بسیار بالایی در رقابت با سایر زبانهای اسکریپت‌نویسی بر روی سرور، مانند ASP ، Perl و Java Servlet را دارد. در واقع این زبان اسکریپت‌نویسی به‌طور مداوم در تست مربوط به نمایش ساده " !hello world " رتبه عالی را کسب می‌کند. برای مشاهده نتایج این‌گونه رده‌بندی‌ها به آدرس زیر مراجعه کنید:

http: // www. perlmonth. com / features / benchmarks / benchmarks.html? issue= 4 & id = 9351

هرچند که بعید است زبان PHP مورد استفاده در پروژه‌های برنامه‌نویسی بزرگ چنین رتبه‌بندی را تأیید نماید، اما به‌واسطه بهره‌گیری از ابزارهای بسیار توانمندی مانند Zend Accelerator (قابل دستیابی از طریق آدرس http: // www. zend. com)که به‌منظور افزایش بیشتر کارایی نسبت به بهره‌گیری از PHP تنها ایجاد شده‌اند، می‌توان بر سرعت فوق‌العاده PHP4 گواهی داد.

قابلیت حمل

PHP به‌گونه‌ای طراحی شده که برروی بسیاری از سیستمهای عامل و به‌همراه بسیاری از وب سرورها قابل اجرا بوده و قادر باشد تا از طیف گسترده‌ای از بانکهای اطلاعاتی بهره ببرد. به‌بیان دیگر، سیستمی که توسط PHP برای محیط عامل UNIX طراحی شده است بدون کوچک‌ترین مشکلی قابل انتقال بر روی محیط عامل Windows NT می‌باشد. امروزه تست عملکرد یک پروژه PHP به‌همراه وب سرور PWS یا Personal Web Server پس از توسعه آن تحت محیط عامل Windows و سپس انتقال و نصب سیستم نهایی تحت محیط عامل UNIX و بهره‌برداری از آن به‌همراه وب سرور متداول و کدباز مشهور Apache، امری عادی شده است.

ویژگی‌های جدید ویرایش دوم کتاب

از زمان انتشار ویرایش اول کتاب حاضر، زبان برنامه نویسی PHP موقعیت و جایگاه خود به‌عنوان یکی از بهترین گزینه‌ها جهت توسعه بر روی وب مستحکم‌تر نموده است. زبان PHP را نیز به مانند بسیاری از پروژه‌های کدباز دیگر می‌توان به‌هدفی متحرک با سرعت تحرک زیاد تشبیه نمود. ویرایش جدید کتاب به‌طور گسترده‌ای بررسی و امر بهینه‌سازی مثالها و تمرینها و هم‌چنین متون آموزشی را مورد توجه قرار داده است. هرجا که ویژگی جدیدی معرفی شده مبحث گسترده‌ای به‌منظور شرح و توضیح جامع آن باز شده است. همچنین بخشهای جدیدی به هر ساعت اضافه شده که در برگیرنده مباحث و سرفصلهای مهمی همچون توابع، آرایه‌ها، بهره‌گیری از اشیا و بانکهای اطلاعاتی و بالاخره زمان (شامل تاریخ و ساعت) می‌باشد.

پس از انتشار اولین ویرایش کتاب انبوهی از درخواستها مبنی بر ارسال کد منبع مثالهای کتاب به‌دستمان رسید. با هماهنگی به‌عمل آمده با ناشر (Sams) اکنون خواننده می‌تواند تمامی این کدها را از طریق وب سایت ناشر به آدرس http: // www. samspublishing. com تهیه نماید.

و در نهایت علاوه بر بازبینی و توسعه محتویات ویرایش اول، بخشی در رابطه با XML و بخش دیگری‌نیزدر رابطه با بهره‌گیری از امکان فوق‌العاده‌ای با عنوان Smarty (تکنیک الگوسازی و کتابخانه‌ای بسیار کارآمد جهت ساماندهی پروژه‌های بزرگ PHP) به ویرایش دوم اضافه شده است.

جمع‌بندی

در این ساعت زبان برنامه‌نویسی PHP به طور مقدمه‌وار مورد بررسی قرار گرفت و تاریخچه‌ای درموردزبان‌که‌شامل تکامل آن از مجموعه‌ای از ماکروها تا یک محیط اسکریپت‌نویسی بسیار توانمند است بیان گردید. همچنین مطالب مفیدی درباره PHP4 و موتور اسکریپت‌نویسی Zend و توانمندی

آنها در به کارگیری ویژگی‌های جدید و پیشرفته به منظور احراز کارایی بهتر، عنوان شد. در انتها نیز برخی از ویژگی‌ها و دلایلی که به واسطه آنها می‌توان زبان PHP را به‌عنوان گزینه قابل توجهی در دنیای برنامه‌نویسی وب فرض کرد، بررسی گردید.

امید داریم که مطالب عنوان شده در این ساعت شما را برای استفاده و بهره‌برداری از زبان برنامه‌نویسی PHP ترغیب کرده باشد. در ساعت آینده به بحث و بررسی درباره چگونگی نصب و آماده سازی محیط PHP خواهیم نشست.

پرسش و پاسخ

پرسش: آیا PHP از نقطه‌نظر فراگیری زبان ساده ای محسوب می‌شود؟

پاسخ: پاسخ به‌سادگی این است " بله. " واقعاً یادگیری اساس زبان PHP در بیست و چهار ساعت امری ممکن است. PHP مجموعه‌ای بسیار غنی از توابعی را تدارک دیده است که برای بهره‌گیری از عملکرد آنها در سایر زبانها، برنامه‌نویس نیاز به نوشتن برنامه دارد. PHP همچنین مسائل مربوط به انواع داده‌ها و نیز مسائل مربوط به بهره‌گیری از حافظه را خود کنترل می‌کند. (از این نظر PHP شباهت زیادی به زبان Perl دارد).

با این همه یادگیری دستور زبان و ساختار برنامه‌نویسی توسط یک زبان خاص تنها آغاز سفر است. در نهایت این شما هستید که به واسطه انجام پروژه‌های برنامه‌نویسی گوناگون و اشتباهاتی که مرتکب می‌شوید، تجربه برنامه‌نویسی خود را افزایش می‌دهید. لذا بهتر است تا مطالعه این کتاب را به‌عنوان نقطه شروع خود برای فراگیری زبان PHP فرض کرده و تا جایی که می‌توانید به کسب مهارت بپردازید.

تمرینها

هدف از این بخش دوره مطالب فراگیری شده در قالب آزمون است. پاسخ بخش آزمون بلافاصله بعد از آن آمده است. بخش فعالیتها شامل تمرینهایی است که به‌منظور افزایش قابلیت برنامه‌نویسی خواننده طراحی شده و البته فاقد پاسخ است.

آزمون

۱- درستی یا نادرستی عبارت زیر را مشخص کنید:

" زبان برنامه‌نویسی PHP در اصل جهت استفاده در صنعت بانکداری توسعه یافته است"

۲- هزینه استفاده از زبان PHP چقدر است؟

۳- نام موتور اسکریپت نویسی جدیدی که موجب افزایش توانمندی زبان PHP شده است، چیست؟

۴- ویژگی‌های جدیدی را که در زبان PHP4 معرفی شده است، بیان کنید؟

پاسخ آزمون

۱- این عبارت نادرست است. چرا که زبان PHP در اصل جهت توسعه برنامه‌های وب طراحی شده است.

۲- استفاده از PHP هزینه‌ای ندارد. این محصول رایگان است.

۳- Zend نام موتور اسکریپت‌نویسی جدیدی است که PHP4 بر روی آن سوار است.

۴- علاوه بر موارد دیگر، PHP4 توابعی برای کار بر روی آرایه‌ها، یک ساختار جدید با نام foreach جهت کار بر روی عناصر آرایه، نوع داده‌ای جدید با نام boolean، توابعی جهت تنظیم و بهره‌گیری از جلسات کاربران و متغیرهای سیستمی جدیدی را معرفی کرده است. ضمن اینکه پشتیبانی از شیوه برنامه‌نویسی شئ‌گرا را توسعه داده و هم‌اینک زبان برنامه‌نویسی Java و نیز تکنولوژی XML را مورد پشتیبانی قرار داده است.

فعالیتها

۱- دلایل خود را در رابطه با فراگیری زبان PHP ذکر کنید. ویژگی‌های مطرح شده در این ساعت چگونه به انجام موفقیت‌آمیز پروژه‌هایتان کمک خواهد کرد؟ دو یا سه پروژه‌ای را برای خود تعریف کنید که پس از اتمام مطالعه کتاب تصور می‌کنید به‌خوبی از عهده انجام آنها برخواهید آمد. به‌موازات مطالعه کتاب نکات مربوط به ویژگی‌های زبان PHP و تکنیک‌های برنامه‌نویسی با این زبان را که در توسعه این پروژه‌ها مفید به‌نظر می‌رسند، یادداشت کنید.

محیطهای عامل، سرورها و بانکهای اطلاعاتی پشتیبانی شده توسط PHP

PHP به‌واقع مستقل از محیط عملیاتی خود است. استفاده از PHP امروزه تحت محیطهای عامل ویندوز، بسیاری از محیطهای عامل یونیکس (شامل Linux) و حتی محیط عامل مکینتاش امری عادی شده است. PHP در راستای پشتیبانی از محیطهای ذکر شده لیست بلند بالایی از وب سرورهای گوناگون وبسیار مشهور را نیز مورد حمایت و پشتیبانی قرار داده است.اسامی آنها بدین شرح است: Apache (که خود پروژه‌ای کدباز بوده و به مانند PHP از محیط‌های عامل مستقل است)، Microsoft Internet Information Server، WebSite Pro، iPlanet Web Server و بالاخره Microsoft Personal Web Server، که در این میان وب سرور آخر جهت تست عملکرد برنامه‌های اسکریپت نوشته شده بر روی یک ماشین ویندوز(کامپیوترهای IBM یا سازگار با آن که سیستم عاملی از نوع ویندوز، مانند Windows XP، Windows Me یا Windows 2000 بر روی آن نصب شده باشد) به صورت off line (بدون اتصال به شبکه جهانی اینترنت) مورد استفاده قرار می‌گیرد. قابل ذکر است که نسخه تحت ویندوز وب سرور Apache نیز چند سالی است که در دسترس می‌باشد.

در صورت نیاز می‌توان PHP را به‌عنوان یک برنامه کاربردی مستقل از طریق سطر فرمان (در سیستم‌های ویندوز) یا Shell (در سیستم‌های یونیکس) مورد استفاده قرار داد. هر چند که ما در این کتاب در مورد چگونگی ایجاد برنامه‌های کاربردی وب بحث خواهیم کرد اما خواننده باید همواره متوجه این نکته باشد که زبان PHP4 مانند زبان برنامه‌نویسی Perl یک زبان اسکریپت‌نویسی عمومی (همه منظوره) می‌باشد. بیان این حقیقت که PHP به عنوان یک برنامه CGI قابل اجراست بدین معنی است که هر سروری که اسکریپت‌های CGI را پشتیبانی کند باید قادر به کار با PHP نیز باشد. قابل ذکر است که در این میان چگونگی پیکربندی سرورهایی که برنامه‌های PHP را به‌عنوان اسکریپت‌های CGI اجرا می‌کنند، با یکدیگر متفاوت است.

از طرف دیگر PHP به گونه‌ای طراحی شده است که به‌سادگی توانایی بهره‌گیری از بانکهای اطلاعاتی را داشته باشد. این ویژگی یکی از مهم‌ترین عواملی است که این زبان برنامه‌نویسی را به عنوان گزینه بسیار مناسبی جهت ایجاد برنامه‌های کاربردی وب مطرح می‌کند. زبان PHP در حال حاضر تقریباً تمامی بانکهای اطلاعاتی موجود را یا به‌طور مستقیم ویا از طریق رابطه ODBC (Open Data Base Connectivity) مورد پشتیبانی قرار داده است.

در تمامی کتاب ما از ترکیب بسیار مناسب و کارآمد Linux (به‌عنوان محیط عامل)، Apache (به‌عنوان وب سرور) و MySQL (به‌عنوان سیستم بانک اطلاعاتی، استفاده خواهیم کرد. هر سه محصول فوق از نوع کدباز بوده و بدون کمترین هزینه‌ای از طریق سایت‌های ftp مربوطه قابل دستیابی هستند.

ساعت دوم

نصب PHP بر روی کامپیوتر

پیش از آنکه خود را درگیر جزئیات زبان PHP کنیم، لازم است تا ساعتی را به چگونگی دستیابی و نصب و همچنین پیکربندی موتور PHP بر روی کامپیوتر و تحت سیستم عامل مورد استفاده‌مان اختصاص دهیم. هم‌اکنون نسخه‌های مختلفی از PHP (با عملکرد یکسان) برای طیف گسترده‌ای از محیطهای عامل منتشر شده که به خوبی بر روی بسیاری از سرورها قابل بهره‌برداری است.

در این ساعت با عناوین زیر آشنا خواهید شد:

- **محیطهای عامل، سرورها و بانکهای اطلاعاتی پشتیبانی شده توسط PHP 4**
- **دستیابی به PHP و سایر نرم‌افزارهای کدباز مفید**
- **بررسی یکی از روشهای متداول نصب PHP تحت Linux**
- **بررسی برخی از گزینه‌هایی که ویژگی‌هایی را به PHP اضافه می‌کنند.**
- **بررسی برخی از گزینه‌های مربوط به پیکربندی PHP**
- **درخواست کمک و راهنمایی به‌هنگامی که اوضاع مطابق انتظار به‌پیش نمی‌رود.**

ضمن اینکه نصب آنها نیز بر روی کامپیوترهای متداول PC نسبتاً آسان است.

نحوه دستیابی به PHP وسایر نیازها

کد زبان PHP4 از طریق وب سایت http://www. php. net قابل دستیابی است. از آنجا که این زبان برنامه‌نویسی پروژه‌ای از نوع کدباز (Open Source) است بارگیری (download) و استفاده از آن برای کاربران کمترین هزینه‌ای در بر ندارد.

وب سایت رسمی مربوط به این زبان به آدرس فوق، یک مرجع بسیار ارزنده و عالی برای برنامه‌نویسان PHP است. مستندات کاملی از این زبان نیز به‌همراه نکات ارزشمند و مفیدی که توسط سایر برنامه‌نویسان PHP در اختیار عموم گذاشته شده، از طریق آدرس http: // www. php. net / manual به‌صورت زنده (Online) قابل استفاده است. علاوه بر این امکان بارگیری این مستندات در چندین قالب مختلف نیز جهت مطالعه Offline وجود دارد.

درمورد محیط عامل Linux و چگونگی نصب و راه‌اندازی و بهره‌برداری از آن وب سایت مفیدی موجوداست که‌آدرس‌آن‌به قرار زیر است: http://www.Linux.org/help/beginner/distributions. html در صورت تمایل به بهره‌برداری از محیط عامل Linux بر روی کامپیوترهایی با نوع معماری Power PC می‌توانید اطلاعات مورد لزوم را از طریق آدرس زیر کسب کنید:

http: // www. Linuxppc. org

(این نسخه از Linux به Linux ppc شهرت دارد).

کاربرانی که از محیط عامل Mac OS x استفاده می‌کنند، می‌توانند بدون کوچک‌ترین مشکلی از مزایای PHP بهره‌مند شوند (محیط عامل ذکر شده جدیدترین محیط عملیاتی شرکت Apple است که بر مبنای محیط‌عامل UNIX شکل گرفته است). اطلاعات مربوط به چگونگی نصب PHP تحت این محیط عامل در آدرس زیر قابل دستیابی است:

http: // www. php. net / manual / en / install. macosx. php

هم‌چنین بانک اطلاعاتی MySQL که در سرتاسر کتاب مورد استفاده قرار خواهیم داد از طریق وب سایت رسمی آن به آدرس: http: // www. mysql. com قابل بارگیری می‌باشد. نسخه‌های مختلف تحت محیطهایی چون UNIX، Windows و OS/2 از این بانک اطلاعاتی در دسترس است.

نحوه نصب PHP بر روی محیط‌عامل Linux و تنظیمات وب سرور Apache

در این بخش نگاهی به چگونگی نصب PHP4 به‌همراه وب سرور مشهور Apache بر روی

محیط عملیاتی Linux خواهیم داشت. این روند تقریباً در مورد تمامی نسخه‌های مختلف سیستمهای UNIX یکسان است. هرچند که نصب نسخه‌های از پیش کامپایل شده PHP بر روی کامپیوتر ساده‌تر از روندی است که در این‌جا به آن می‌پردازیم، اما روش مورد استفاده ما برای این کار امکان کنترل بیشتر و بهتری را در اختیارمان قرار می‌دهد.

پیش از هر اقدامی برای نصب PHP ابتدا لازم است تا از اتصال خود به سیستم Linux با عنوان کاربر اصلی (root) اطمینان حاصل کنید. در صورتی‌که چنین مجوزی ندارید، مجبورید تا از مدیر (راهبر) سیستم بخواهید تا عملیات نصب PHP را انجام دهد.

برای نصب PHP به‌همراه وب سرور Apache دو راه پیش رو دارید: روش اول، این است که کد منبع Apache را مجدداً کامپایل کرده و PHP را به آن لینک نمایید. در روش دوم کافی است تا PHP را به‌عنوان یک DSO (یا Dynamic Shared Object) کامپایل کنید. چنان‌چه نسخه Apache مورد استفاده شما از قابلیت DSO پشتیبانی می‌کند، می‌توانید ماجول‌های جدید (مانند PHP) را بدون نیاز به کامپایل مجدد وب سرور (Apache) مورد استفاده قرار دهید. این روش ساده‌ترین راه ممکن برای نصب و بهره‌گیری از PHP می‌باشد. بنابراین ما نیز در این بخش از این روش استفاده خواهیم کرد.

ابتدا، برای کشف این مطلب که آیا نسخه Apache از قابلیت DSO پشتیبانی می‌کند یا خیر لازم است تا برنامه‌ای با عنوان httpd را با استفاده از آرگومان L- اجرا نمایید (دقت کنید که علامت $ ظاهر شده در ابتدای خط اعلان سیستم UNIX یا Linux مورد استفاده است). طریقه انجام این کار به‌صورت زیر است:

```
$ / www / bin / httpd - L
```

با اجرای این برنامه لیستی از ماجول‌های موجود را مشاهده خواهید کرد. در صورتی که ماجولی با عنوان mod_so.c را مابین این ماجول‌ها مشاهده می‌کنید، به این معنی است که نسخه Apache نصب شده بر روی سیستم UNIX شما از قابلیتDSO پشتیبانی می‌کند و لذا قادرید تا بدون کامپایل مجدد Apache به نصب PHP بپردازید (کامپایل مجدد بخشی از زندگی کاربران سیستم‌های UNIX است. در بسیاری از موارد کاربران جهت تحصیل کارایی بهتر و تنظیم سیستم به منظورهای مختلف، مانند بهره‌گیری از آن جهت ارائه یک یا چند سرویس خاص حتی دست به کامپایل مجدد ***Kernel*** [1] سیستم عامل نیز می‌زنند و به‌عبارت دیگر سیستم را tune می‌کنند. از این‌رو تمامی سیستمهای عامل UNIX و همچنین Linux دارای کامپایلر C و C++ هستند. چیزی که در مورد سیستمهای عامل Windows مشاهده نمی‌کنیم ـ مترجم). در صورت عدم مشاهده ماجول ذکر شده لازم است تا دست

۱- قلب هر سیستم عاملی که عملیات اساسی مورد نیاز مانند مدیریت فرآیندها و برنامه‌ها، مدیریت حافظه، مدیریت ورودی خروجی و سایر عملیات حیاتی هر سیستمی را کنترل می‌کند. ـ مترجم

به کامپایل مجدد وب سرور Apache بزنید. برای مشاهده چگونگی انجام این کار به مستندات مربوطه مراجعه کنید.

اگر تاکنون اقدام به بارگیری آخرین نسخه PHP (یعنی نسخه PHP4.0.6 تا زمان انتشار این کتاب) نکرده‌اید، اکنون زمان آن است تا از سایت مربوطه یعنی http: // www. php. net آن‌را تهیه کنید. نسخه‌های تهیه شده برای سیستمهای UNIX در قالب خاصی با عنوان tar آرشیو شده و با استفاده از برنامه gzip فشرده شده‌اند. لذا گام اول پس از دستیابی به PHP4 این است که عکس فرآیند مذکور را به صورتی که در زیر آمده است، انجام دهید:

```
$ tar - xvzf  php - 4.0.6. tar. gz
```

پس از انجام این کار با فرمان cd به فهرست مربوط به PHP4 تغییر مسیر دهید. دستور زیر چگونگی انجام این کار را نشان می‌دهد:

```
$ cd .. / php - 4.0.6
```

در این فهرست اکنون باید اسکریپت خاصی را که جهت پیکربندی PHP طراحی شده و Configure نام دارد، مشاهده نمایید. این اسکریپت خاص از طریق سطر فرمان به صورتی که در ادامه خواهید دید قابل اجرا بوده و از طریق آرگومان‌های گوناگونی که می‌پذیرد، قادر است تا ویژگی‌های PHP نصب شده بر روی سیستم را به‌دقت کنترل نماید. برای درک بهتر مطلب در این بخش، از برخی آرگومان‌هایی که امکانات مفید و جالب توجهی را در اختیار قرار می‌دهند استفاده خواهیم کرد. در صورت تمایل می‌توانید نوع پیکربندی PHP را مطابق نیازهای خود با به‌کاربردن آرگومان‌های مربوطه مشخص کنید. در ادامه این ساعت به بحث در مورد برخی از این گزینه‌های پیکربندی خواهیم نشست. پیکربندی مورد نظر ما با اجرای اسکریپت Configure با استفاده از آرگومان‌های مورد نیاز به‌صورتی که در زیر مشاهده می‌کنید، به‌دست می‌آید:

```
$ . / configure -- with - gdbm = / usr \
                -- with - apxs \
                -- with - mysql \
                -- with - xml \
                -- with - sablot = / usr \
                -- with - expat - dir = / usr / local \
                -- with -xslt - sablot \
                -- with - gd = / usr \
                -- enable - gd - native - ttf \
                -- with - ttf = / usr \
                -- with - dom = / usr \
                -- with - zlib = / usr \
```

آرگومان‌های مورد استفاده در اجرای اسکریپت Configure به‌صورت فوق جهت پشتیبانی از ویژگی‌هایی است که در این کتاب مورد بحث و بررسی قرار می‌گیرند. بیشتر آنها مستلزم این واقعیت است که پیش از نصب PHP بر روی سیستم باید کتابخانه‌های ویژه‌ای را بر روی آن نصب کرده باشید.

چنان‌چه اسکریپت Configure در یافتن این کتابخانه‌ها با شکست مواجه شود، پیکربندی مورد نظرتان به‌طور ناقص انجام خواهد شد.

پس از اجرای موفقیت آمیز اسکریپت Configure اکنون نوبت به اجرای برنامه‌ای با عنوان Make رسیده است. برای اجرای موفقیت آمیز این برنامه وجود یک کامپایلر C بر روی سیستم مورد استفاده‌تان ضروری است (سیستم‌های UNIX خود دارای چنین کامپایلری هستند). اجرای این برنامه با ارسال دو فرمان زیر به صورت توام صورت می‌پذیرد:

```
$ make
$ make install
```

(فرمانهای فوق را از درون فهرست مربوطه به PHP صادر کنید).

با اجرای این دو فرمان فرآیند کامپایل و نصب PHP4 به پایان می‌رسد. اکنون می‌توانید به پیکربندی سرور Apache و بهره‌گیری از PHP بپردازید.

تنظیم برخی از گزینه‌های برنامه Configure

همان‌گونه که ذکر شد آرگومان‌های اسکریپت Configure که از طریق سطر فرمان به این برنامه ارسال می‌شوند، قادرند تا ویژگی‌های مورد انتظارمان را به موتور PHP، یعنی بخشی از آن که برنامه‌های نوشته شده به زبان PHP را اجرا و کنترل می‌کند، اضافه نمایند. درواقع جهت اطلاع از اسامی آرگومان‌های قابل به‌کارگیری می‌توان از خود برنامه configure استفاده نمود. برای انجام این کار کافی است تا از فهرستی که PHP را در آن‌جا نصب کرده‌اید، برنامه مذکور را به‌صورت زیر اجرا کنید:

```
$ ./ configure  -- help
```

از آنجا که لیست تولید شده توسط اجرای فوق شامل اقلام زیادی است، بهتراست تا با استفاده از عملگر تغییر مسیر به صورتی که در اجرای زیر مشاهده می‌کنید نتایج حاصل از اجرا را جهت مطالعه به درون یک فایل متن منتقل نمایید:

```
$./configure  -- help > configoptions. txt
```

با وجودی که نتایج حاصل از اجرای این برنامه کاملاً گویای آرگومان‌های قابل استفاده است اما در ادامه به برخی از آنها توجه خواهیم کرد، به‌ویژه آنهایی که جهت مطالعه ادامه کتاب به آنها نیاز دارید.

آرگومان -- with - gdbm

بهره‌گیری از این آرگومان پشتیبانی از کتابخانه ویژه‌ای تحت عنوان Gnu Database Manager را در پی دارد. وجود این کتابخانه یا هر کتابخانه دیگری که از نوع DBM باشد، جهت استفاده از توابع

DBA مورد بحث در ساعت یازدهم کتاب ضروری است. چنانچه از سیستم Linux استفاده می‌کنید به‌احتمال قوی چنین کتابخانه‌ای در دسترستان است. در غیر این‌صورت لازم است تا مطالب درس یازدهم با عنوان " بهره‌گیری از توابع DBA " را به دقت مطالعه کرده و به‌کار ببندید. در صورتی که کتابخانه DBM مورد استفاده‌تان در موقعیتی غیر استاندارد واقع شده‌است به‌صورتی که در فرمان زیر مشاهد می‌کنید، می‌توانید مسیر آن‌را در اختیار PHP قرار دهید.

```
$ ./ configure -- with - gdbm = / path / to / dir
```

آرگومان -- with - gd

بهره‌گیری‌از این آرگومان پشتیبانی از کتابخانه GD را در پی خواهدداشت. در صورتی که چنین کتابخانه‌ای بر روی سیستم نصب شده باشد اسکریپت‌های PHP قادر خواهند بود تا تصاویر پویایی از نوع GIF یا PNG ایجاد نمایند. در مورد چگونگی ایجاد این‌گونه تصاویر در درس ساعت چهاردهم با عنوان " گرافیک و PHP " به بحث مجدد خواهیم پرداخت. مشابه آرگومان قبل در این مورد نیز امکان تعیین مسیر استقرار کتابخانه توسط برنامه Configure به صورت زیر وجود دارد:

```
$ ./ configure -- with - gd = / path / to / dir
```

چنانچه کامپایل برنامه‌ها با اشکال همراه شود، لازم است تا هنگام استفاده از این آرگومان مسیر کتابخانه GD به‌طور صریح مشخص گردد.

آرگومان -- with - ttf

بهره‌گیری از این آرگومان پشتیبانی از کتابخانه Free Type 1 را در پی دارد. استفاده از این کتابخانه امکان استفاده از انواع قلمها را هنگام ایجاد تصاویر GIF یا PNG در اختیار برنامه‌نویس قرار می‌دهد. جهت فعال سازی این گزینه لازم است تا ابتدا کتابخانه Free Type 1 را بروی سیستم خود نصب کنید. به منظور اطلاع بیشتر در مورد امـکانات این کتابخانه به وب سایت مربوطه به آدرس http:// www. Free type. org مراجعه کنید. مانند آرگومان‌های قبل در صورت بروز مشکل می‌توانید مسیر کتابخانه را صریحاً به‌صورت زیر مشخص نمایید:

```
$ ./ configure -- with - ttf = / path / to / dir
```

آرگومان -- with - mysql

بهره‌گیری از این آرگومان پشتیبانی از بانکهای اطلاعاتی MySQL را در پی خواهد داشت. وجود کتابخانه مربوطه در این مورد نیز الزامی است (این کتابخانه به‌همراه برنامه نصب بانک اطلاعاتی مذکور منتشر می‌شود). در این‌جا نیز هنگام بروز اشکال می‌توان مسیر کتابخانه را به‌طور صریح ذکر نمود:

```
$ ./ configure -- with - mysql = / path / to / dir
```

همان‌گونه که می‌دانید PHP علاوه بر MySQL قادر به پشتیبانی از سایر بانکهای اطلاعاتی نیز می‌باشد.

جدول ۱- ۲، لیست اسامی برخی از آنها را به‌همراه آرگومان مورد نیاز جهت استفاده در برنامه Configure نشان می‌دهد.

جدول ۱- ۲ گزینه‌های پشتیبانی از بانکهای اطلاعاتی مختلف جهت استفاده برنامه Configure

نام بانک اطلاعاتی	گزینه مربوط در برنامه Configure
DBA	-- with - dba
DBM	-- with - dbm
GDBM	-- with - gdbm
Adabas D	--with - adabas
File Pro	-- with - filepro
Msql	-- with - msql
InFormix	-- with - informix
IODBC	-- with - iodbc
OpenLink ODBC	-- with - openlink
Oracle	-- with - oracle
Postgre SQL	-- with - pgsql
Solid	--with - solid
Sybase	-- with - sybase
Sybase - CT	-- with - sybase - ct
Velocis	-- with - velocis
LDAP	-- with - ldap

تنظیم گزینه‌های مربوط به پشتیبانی از XML

در این کتاب برخی از ویژگی‌های XML را مورد بررسی قرار می‌دهیم. توجه داشته باشید که آرگومان with - sabolt -- پشتیبانی از XSLT را میسر می‌کند اما پیش از هر اقدامی لازم است تا موتور Sablotron را که شامل کتابخانه مربوطه است، بر روی سیستم نصب نمایید. دستیابی به این

موتور از طریق وب سایت مربوطه به آدرس http: // www. Gingerall. com امکان‌پذیر است.

جهت پشتیبانی از Expat بهره‌گیری از آرگومان with - xml -- ضروری است.

همچنین جهت بهره‌برداری از توابع XML استاندارد DOM لازم است تا نسخه 2.2.7 یا بالاتر کتابخانه Libxml را که از طریق وب سایت http: // www. Xmlsoft. Org قابل دستیابی است در اختیار داشته باشید. جهت تعیین مسیر این کتابخانه برنامه Configure را به صورت زیر اجرا نمایید:

```
$ ./ configure -- with - dom = / path / to / rib
```

پیکربندی وب سرور Apache

پس از کامپایل موفقیت آمیز PHP و Apache لازم است تا فایل مربوط به پیکربندی Apache با عنوان httpd. Conf را مورد بررسی قرار دهید (این فایل در فهرستی با نام conf از فهرستی که Apache را در آن‌جا نصب کرده‌اید، مستقر است). خطوط زیر را به متن این فایل اضافه کنید:

```
Add Type application / x- httpd - php . php
Add Type application / x- httpd - php - source . Phps
```

این عمل باعث می‌شود تا تمامی فایل‌هایی که دارای پسوند php. هستند جهت بررسی و پردازش به موتور PHP که هم‌اکنون بر روی سیستم نصب شده است، ارسال شوند. ضمن اینکه فایل‌هایی که دارای پسوند phps. هستند، به‌عنوان کد منبع (source code) برنامه‌های PHP در نظر گرفته خواهند شد. بدین ترتیب که کدهای مزبور به‌عنوان یک صفحه HTML در مرورگرهای اینترنت (مانندNetscape و Internet Explorer) قابل مشاهده و بررسی می‌باشند. علاوه بر این مرورگرهای فوق بخشهای مختلف کد منبع را با رنگهای متفاوت مشخص می‌کنند، که این امر باعث سادگی در امر اشکال زدایی خواهد شد.

اگر مایلید تا از پسوند دیگری که برای کاربران شما ملموس‌تر است استفاده کنید، می‌توانید از هر پسوند دلخواهی که مدنظرتان است، بهره ببرید. حتی به‌گونه‌ای که در اینجا مشاهده می‌کنید می‌توانید کلیه فایل‌های با پسوند html. را به‌عنوان یک فایل PHP در مرورگر کاربران خود به نمایش درآورید. کافیست تا خط زیر را به فایل مربوط به پیکربندی وب سرور Apache یعنی httpd. php اضافه کنید:

```
AddType application / x- httpd - php . html
```

با این حال ذکر این نکته حائز اهمیت است که انجام فرآیند مذکور، یعنی تنظیم پیکربندی وب سرور به‌گونه‌ای که فایل‌های Html را به‌عنوان برنامه‌های PHP در نظر بگیرد منجر به کند شدن وب سایت خواهد شد، از آن جهت که تمامی فایل‌های html پیش از ارسال بر روی مرورگر کاربران توسط موتور PHP مورد پردازش قرار می‌گیرند.

چنانچه PHP از پیش بر روی سیستم نصب شده و شما نیز هیچ‌گونه دسترسی به فایل پیکربندی وب سرور Apache ندارید با تغییری که به‌واسطه اضافه کردن گزینه Add Type به فایل htaccess. می‌دهید، می‌توانید تعیین کنید که چه نوع فایل‌هایی توسط موتور PHP مورد پردازش قرار بگیرند. با ایجاد چنین فایلی و اعمال تغییر مذکور به آن فایل‌های موجود در فهرست مربوط به این فایل و نیز فایل‌های موجود در زیر فهرست آن از چنین تغییری متاثر خواهند شد، به این شرط که گزینه‌ای با عنوان AllowOverride برای فهرست مزبور با عنوان Fill Info یا All تنظیم شده باشد.

با وجودی که فایل htaccess . فایل پیش فرض جهت کنترل دسترسی است، به‌راحتی می‌توان چنین شرایطی را تغییر داد. برای این‌کار مقدار گزینه Access File Name در فایل پیکربندی httpd. conf را مورد بازبینی قرار دهید. حتی اگر شما کاربر اصلی (root) سیستم عامل خود نباشید، باز هم می‌توانید محتویات فایل پیکربندی فوق را مورد بررسی و بازخوانی قرار دهید.

در صورتی‌که کاربر اصلی سیستم عامل خود نباشید، روش استفاده از فایل کنترل دسترسی htaccess . یک روش عالی جهت تنظیم دقیق نحوه دسترسی است. با این حال روش اصلی جهت تنظیم رفتارهای PHP دستکاری فایلی با نام php . ini می‌باشد.

فایل php. ini

پس از کامپایل و نصب PHP به‌راحتی و با تغییراتی که در فایل PHP. ini می‌دهید، می‌توانید رفتار مورد نظرتان را از PHP به دست آورید. در سیستمهای UNIX موقعیت پیش فرض این فایل فهرست usr/local/lib / است اما در سیستمهای Windows چنین فایلی را می‌توانید در فهرست خود سیستم عامل پیدا کنید. هم‌اینک باید بتوانید چنین فایلی را در موقعیتهای ذکر شده مشاهده کنید. تنظیمات اولیه این فایل به‌طور پیش فرض انجام شده است. این تنظیمات معرف رفتار پیش فرض PHP می‌باشند. گزینه‌های موجود در فایل php .ini را عموماً می‌توان به دو دسته تقسیم کرد: مقادیر و پرچمها. گزینه‌های نوع اول شکل عمومی " مقدار = نام" دارند که البته هر دو سمت تساوی از گزینه‌ای به گزینه دیگر متفاوت است. شکل عمومی گزینه‌های نوع دوم نیز مشابه همین است با این تفاوت که سمت راست تنها می‌تواند شامل یکی از مقادیر 1، On، Yes یا True برای مقادیر مثبت و 0 ، Off، No و یا False برای مقادیر منفی باشد. کلیه فضاهای خالی در مقادیر این گونه گزینه‌ها صرف‌نظر می‌شوند.

چنان‌چه PHP هم‌اینک بر روی سیستم شما وجود دارد، می‌توانید مقادیر مربوط به گزینه‌های موجود در فایل php. ini را مورد مشاهده و بررسی قرار دهید.

در هر زمان دلخواه که مایل باشید، می‌توانید تغییرات مورد نظرتان را به فایل php. ini اعمال کنید. اما در صورتی که PHP را به‌عنوان یکی از ماجول‌های Apache مورد استفاده قرار می‌دهید، برای

اعمال تغییراتی که بر روی این فایل می‌دهید، لازم است تا وب سرور را Shutdown کرده و آن‌را مجدداً راه‌اندازی کنید.

گزینه Short _ Open _ tag

این گزینه برای تشخیص این مطلب است که آیا می‌توان شروع و پایان کد برنامه‌های PHP را به‌ترتیب با علائم ? › و ‹ ? نشانه گذاری کرد یا خیر. در صورتی که این گزینه غیر فعال باشد یکی از خطوط زیر را در فایل php. ini مشاهده خواهید کرد:

Short _ open _ tag = off
Short _ open _tag = false
Short _ open _tag = no

برای فعال کردن این گزینه کافی‌است یکی از خطوط زیر را جایگزین خط مزبور نمایید:

Short_ open _ tag = on
Short _open _tag = true
Short _ open_ tag = yes

در درس ساعت سوم مطالب بیشتری در مورد نشانه‌های شروع و پایان کد PHP فرا می‌گیرید.

گزینه‌های مربوط به گزارش خطا

جهت تشخیص خطا لازم است گزینه‌های مربوط به گزارش خطا را در فایل php. ini تنظیم کنید. چگونگی انجام این کار چنین است:

display_ errors = On

هم‌چنین می‌توانید سطح گزارش دهی را نیز تعیین کنید. جهت هماهنگی با کتاب حاضر باید گزینه مربوطه را به صورتی که در زیر مشاهده می‌کنید، تنظیم نمایید.

Error_ reporting = E _ ALL & ~ E_ NOTICE

این عمل موجب می‌شود تا تمامی خطاهای برنامه گزارش شده اما در مورد بخش یا بخشهایی از کد که ممکن است مستعد خطا باشند اخطاری به برنامه‌نویس داده نخواهد شد. البته تنظیم مذکور به طور پیش فرض اعمال شده است. لازم به ذکر است که بخشهای مستعد خطا از بهره‌گیری از برخی تکنیکها در PHP جلوگیری به‌عمل می‌آورند.

تنظیم سایر گزینه‌ها

PHP به‌عنوان نتیجه یک درخواست از نوع GET یا POST و یا تنظیم کوکی قادر است تا متغیرهایی را در اختیار برنامه نویس قرار دهد. مشاهده و تغییر این متغیرها از طریق فایل php. ini امکان‌پذیر است.

پیش از نسخه PHP- 4.0.3 گزینه track _ vars موجب می‌شدتا PHP آرایه‌هایی از نوع انجمنی

ایجاد کند. چنین آرایه‌هایی عناصری را به‌عنوان نتیجه یک درخواست HTTP در خود ذخیره می‌کردند. این ویژگی در حال حاضر به‌طور پیش‌فرض فعال است. در صورت استفاده از نسخه‌های قدیمی‌تر PHP و عدم امکان در بهنگام سازی نسخه موجود لازم است تا مقدار این گزینه را برابر با on قرار دهید.

همچنین گزینه register_ globals مشخص می‌کند که آیا مقادیر حاصل از درخواست HTTP باید به‌عنوان متغیرهای سراسری در دسترس برنامه‌نویس قرار بگیرند یا خیر. در مورد بسیاری از برنامه‌های کتاب پاسخ مثبت است. لذا اطمینان حاصل کنید که متغیر مورد بحث به صورت زیر تنظیم شده است:

register _ globals = on

تنظیم گزینه‌های فایل php. ini از طریق برنامه‌نویسی

چنان‌چه PHP رابه‌عنوان یکی از ماجول‌های Apache مورد استفاده قرار می‌دهید و پیکربندی‌تان به‌گونه‌ای است که می‌توانید از فایل دسترسی htaccess. استفاده کنید، قادرید تا گزینه‌های موجود برای تنظیم فایل php. ini را به ازای هر فهرست به‌طور جداگانه انجام دهید.

در داخل فایل دسترسی htaccess. می‌توانید از گزینه php - flag جهت تنظیم پرچمهای موجود در فایل php. ini (گزینه‌هایی که با مقادیر on و off تنظیم می‌شوند) و از گزینه php - Value نیز جهت تنظیم مقادیر موجود در فایل php. ini (گزینه‌هایی که با استفاده از مقادیر عددی یا رشته‌های کاراکتری قابل تنظیم هستند)، بهره‌ببرید. طریقه انجام این کار چنین است:

```
php_ flag  Short _ Open _ t ag  On
php _ Value  include _ path " . : / home / corrder "
```

اما در صورتی‌که به Apache دسترسی ندارید باز هم امکان اعمال تنظیمات وجود دارد. در نسخه PHP-4.0.5 و بالاتر تابعی با عنوان ( ) ini_ set معرفی شده است. این تابع اجازه می‌دهد تا برخی از گزینه‌های فایل php. ini را از درون کد برنامه‌های PHP تنظیم کنید. تابع مذکور جهت اجرا به دو رشته کاراکتری نیاز دارد: نام گزینه مربوطه و مقدار آن:

```
ini _ set(  " include_ path ", ". : / home / corrdev"  ) :
```

در درس ساعت ششم مطالب زیادی در مورد توابع فرا می‌گیرید.

کمک و راهنمایی بیشتر

اینترنت همواره بهترین مرجع جهت دریافت راهنمایی و کمک برای بسیاری از مسائل است. به‌ویژه اگر این مسأله مربوط به یک محصول کامپیوتری کدباز باشد. با این همه پیش از ارسال هرگونه پرسشی اندکی تامل کنید. بدون توجه به چگونگی نصب PHP، پیکربندی، و یا مشکلات برنامه‌نویسی

به احتمال قوی افراد دیگری نیز هستند که با چنین مشکلاتی دست به گریبانند و البته افرادی هم هستند که پاسخ این مشکلات را می‌دانند.

اولین مرجع برای مراجعه کاربران PHP وب سایت رسمی این محصول به آدرس http: // www. php. net است، به‌ویژه مستندات این محصول که دربرگیرنده بسیاری از مسائل و نکات مهم است. آدرس این مستندات به این قرار است:

http: // www. php. net / manual

در صورتی که نمی‌توانید پاسخ سؤال خود را در این مستندات پیدا کنید، امکان جست‌وجوی وب سایت PHP را فراموش نکنید. پاسخ پرسش شما ممکن است در یکی از مستندات منتشر شده جدای از مستندات اصلی و یا اصلاً در فایلی موسوم به FAQ (پرسشهای متداول یا Frequently Asked Questions) باشد. جست‌وجو در وب سایت PHP Builder روش دیگری است که برای یافتن پاسخ می‌توانید به آن امید داشته باشید. آدرس این وب سایت بدین قرار است:

‹http: // www. php builder. Com›

چنان چه باز هم دریافتن پاسخ عاجز ماندید، جست‌وجوی آرشیو پرسشهای لیست پستی http: // www. php. net / search. php را فراموش نکنید. در این آرشیو منابع عظیمی از اطلاعاتی که به واسطه همکاری با بسیاری از جوامع بزرگ PHP به‌دست آمده، گردآوری شده است. به‌همین علت توصیه می‌کنیم که مدتی را صرف جست‌وجو در این آرشیو کنید. به احتمال قوی پاسخ خود را در همین‌جا خواهید یافت.

اگر به اندازه کافی قانع شدید که پاسخ مورد نظرتان به روشهای ذکر شده به‌دست نمی‌آید، بهترین روش ثبت نام در یک از جوامع PHP موجود است.

از طریق آدرس http: // www. php. net / support . php می‌توانید در چندین لیست پستی PHP ثبت نام کنید. هرچند که تجمع کاربران این لیستهای پستی بسیار بالاست ولی مطمئن باشید که مطالب زیادی از آنها فرا خواهید گرفت. اگر در رابطه با برنامه‌نویسی با PHP تصمیم قطعی دارید دست کم در یکی از این لیستها ثبت نام کنید. پس از ثبت نام در لیست پستی مورد نظر می‌توانید مشکل خود را برای مشاهده کلیه اعضای لیست، پست کنید.

جهت ارسال مشکل خود، حتی‌المقدور اطلاعات بیشتری در مورد آن بدهید. مطالب را مختصر و مفید بیان کنید. و از ارسال نامه‌های طویل خودداری کنید. در صورت امکان اقلام زیر را در نامه خود ذکر کنید:

- سیستم عامل مورد استفاده
- نسخه PHP مورد استفاده
- گزینه‌های مورد استفاده در فایل پیکربندی Cofigure

- هرگونه پیغام خروجی که از فرمان Configure یا Make حاصل شده و احتمال می‌دهید که با مشکل مورد نظرتان به نوعی در ارتباط باشد.
- نمونه کامل از برنامه‌ای که موجب بروز خطا شده است.

اما چرا این همه توجه در ارسال یک پرسش به لیست پستی لازم است؟ پیش از همه تقویت مهارت، روحیه تحقیق شما را در وضعیت مطلوبی نگه می‌دارد. محققین خوب و زبده عموماً مشکلات را بسیار سریع‌تر و مؤثرتر از سایرین حل و فصل می‌کنند. پرسیدن یک سؤال ابتدایی از متخصصین معمولاً منجر به دریافت تنها یک یا دو پاسخ می‌شود، که آن‌هم شامل معرفی آرشیوها و بایگانی‌هایی است که می‌توانید پاسخ خود را در آنجا پیدا کنید. به‌عبارت دیگر نقطه‌ای که تحقیق خود در یافتن پاسخ را از آنجا آغاز می‌کنید.

به‌عنوان نکته دوم، به‌خاطر داشته‌باشید که لیستهای پستی مشابه مراکز پشتیبانی سرویس دهی تلفنی نیستند. هیچ‌کس بابت کمک و راهنمایی به شما مبلغی را دریافت نمی‌کند. علیرغم این موضوع شما به منبع بسیار ارزشمندی از اطلاعات که بخشهایی از آن از پایه گذاران خود PHP به‌دست آمده است، دسترسی پیدا می‌کنید. همواره پرسشها و پاسخهای مفید برای استفاده و بهره‌برداری توسط سایر کاربران در آینده بایگانی می‌شوند. ارسال پرسشی که بارها و بارها قبلاً مورد سؤال بوده، بدین ترتیب تنها موجب اتلاف وقت خواهد بود.

البته این واقعیتها نباید موجب شود تا از ارسال پرسش خود به لیستهای پستی دچار وحشت و اضطراب یا نگرانی شوید. توسعه دهندگان PHP همواره از خوش مشرب‌ترین و خوش برخوردترین‌ها هستند. علاوه بر این مطمئن باشید که طرح مسأله و مشکلات در جوامع PHP به سایر کاربرانی که در آینده با چنین مسائلی روبرو می‌شوند، کمک شایانی خواهد کرد.

جمع‌بندی

همان‌گونه که در این ساعت متوجه شدیدPHP یک نرم‌افزار کدباز است، ضمن اینکه هیچ پیش فرضی در مورد نوع وب سرور، سیستم عامل یا بانک اطلاعاتی در رابطه با بهره‌گیری از PHP وجود ندارد.

در درس این ساعت چگونگی دستیابی به نرم‌افزار PHP و نیز سایر نرم‌افزارهای کدبازی را که می‌توانند در میزبانی و سرویس‌دهی وب سایت‌ها به شما کمک کنند، فرا گرفتید. همچنین با نحوه کامپایل کردن PHP به‌عنوان یکی از ماجول‌های وب سرور Apache بر روی Linux (و عموماً UNIX) آشنا شدید. اگر به نرم‌افزار PHP برای سایر محیطهای عامل دسترسی دارید، مطمئن باشید که دستورالعملهای نصب گام به گام و همچنین چگونگی پیکربندی را نیز در اختیار دارید. این درس علاوه بر نصب، چگونگی پیکربندی PHP را نیز جهت دستیابی به ویژگی‌های مختلفی که PHP قادر به

پشتیبانی از آنهاست مورد بحث و بررسی قرار دارد. در این ضمن، شما با نحوه پیکربندی فایل php. Ini و گزینه‌های موجود در آن آشنا شدید. در انتهای درس ما منابعی را که جهت دریافت راهنمایی و پشتیبانی می‌توانید به آنها مراجعه کنید، مورد بررسی قرار دادیم. اکنون باید به اندازه کافی جهت فراگیری خود زبان آمادگی‌های لازم را کسب کرده باشید.

در درس ساعت آینده اولین برنامه اسکریپت خود را خواهیم نوشت. در این راه با گرامرهایی از زبان PHP آشنا می‌شوید که جهت متمایز کردن کد HTML از کد PHP به آنها نیاز دارید.

پرسش و پاسخ

پرسش: در این درس تنها در مورد چگونگی نصب PHP برروی سیستم عامل Linux و بهره‌گیری از وب سرور Apache بحث و بررسی شد. آیا این بدان معنی است که مطالب کتاب درباره سایر سیستمهای عامل و وب سرورها مانند Windows و Internet Information Server کارآیی ندارد؟

پاسخ: خیر. یکی از نقاط قوت PHP در این است که بهره‌گیری از آن تحت محیطهای عامل مختلف امکان‌پذیر است. در صورتی که در مورد نصب PHP بر روی سیستم عامل مورد استفاده‌تان یا در مورد استفاده از وب سرور مورد نظرتان مشکل دارید، مراجعه و بازخوانی مستنداتی را که به‌همراه PHP منتشر می‌شود، از یاد نبرید. در این مستندات چگونگی نصب و پیکربندی PHP در قالب دستورالعملهای گام به گام آمده است. در صورت تداوم مشکل، بخش " درخواست کمک و راهنمایی بیشتر" از درس این ساعت را مجددا مطالعه کنید. منابع Online موجود در آن بخش به احتمال زیاد پاسخ نیاز شما خواهند بود.

تمرینها

هدف از این بخش، دوره مطالب فراگیری شده در قالب آزمون است. پاسخ بخش آزمون بلافاصله بعد از آن آمده است. بخش فعالیتها شامل تمرینهایی است که به منظور افزایش قابلیت برنامه‌نویسی خواننده طراحی شده و البته فاقد پاسخ است.

آزمون

۱- دسترسی به مستندات Online مربوط به PHP از طریق کدام نشانی امکان‌پذیر است؟

۲- در سیستم‌های عامل UNIX (و Linux) چگونه می‌توان به جزئیات مربوط به گزینه‌های پیکربندی دست یافت (منظور گزینه‌هایی است که به‌عنوان آرگومان برنامه Configure مورد استفاده قرار می‌گیرند).

۳- نام فایل پیکربندی وب سرور Apache چیست؟

۴- انجام چه تغییراتی در فایل پیکربندی Apache لازم است تا فایل‌هایی که دارای پسوند php . هستند به‌عنوان برنامه‌های PHP برای وب سرور نامبرده قابل شناسایی باشند.

۵- نام فایل پیکربندی PHP چیست؟

پاسخ آزمون

۱- مستندات PHP4 از طریق نشانی http: // www. php. net قابل دستیابی است.

۲- با فراخوانی (اجرای) برنامه اسکریپت Configure در فهرست PHP و بهره‌گیری از آرگومان help -- به صورت زیر می‌توان گزینه‌های پیکربندی مورد نظر را مشاهده کرد.

۳- فایل پیکربندی مزبور httpd. Conf نام دارد.

۴- با اضافه کردن خط زیر:

Add Type application / x - httpd - php .php

می‌توان اطمینان داشت که وب سرور Apache کلیه فایل‌های با پسوند php. را به‌عنوان برنامه‌های اسکریپت PHP4 تلقی خواهد کرد.

۵- نام این فایل php. ini است.

فعالیتها

۱- PHP را بر روی سیستم مورد استفاده‌تان نصب کنید. اگر در حال حاضر این نرم‌افزار بر روی سیستم شما وجود دارد محتویات فایل پیکربندی php. ini را مورد بازبینی قرار داده و مطابق نیاز آن‌را تغییر دهید.

ساعت سوم

اولین برنامه اسکریپت

پس از نصب پیکربندی PHP اکنون نوبت آن است که این مجموعه را تست نماییم. در درس این ساعت اولین برنامه اسکریپت خود را ایجاد کرده و مدت زمان کوتاهی را صرف تحلیل گرامر و دستور زبان PHP خواهیم نمود. در انتهای درس با دانشی که به‌دست می‌آورید، باید بتوانید اسنادی تهیه کنید که به‌طور توأم شامل کدهای PHP و HTML باشند.

در این ساعت مطالب زیر را فرا می‌گیرید:

- **چگونگی ایجاد، بارگذاری (upload) و اجرای برنامه‌های PHP**
- **نحوه استفاده توأم از کدهای HTML و PHP در یک سند وب**
- **افزایش وضوح برنامه‌ها با استفاده از درج توضیحات**

در ادامه به بررسی این عناوین می‌پردازیم.

نوشتن اولین کد اسکریپت

در این بخش اولین اسکریپت PHP خود را به سرعت توسعه می‌دهیم. برای شروع برنامه ویراستار متن مورد علاقه خود را باز کنید. مشابه اسناد HTML، فایل‌های PHP نیز از متون ساده (Plain text) تشکیل می‌شوند. برای ایجاد چنین فایل‌هایی می‌توانید از هر نوع ویراستار متنی استفاده کنید. ویراستار Notepad در سیستم عامل Windows، ویراستارهای Simple Text و BBEdit در سیستم‌های MacOS و ویراستارهای VI و Emacs در سیستمهای عامل UNIX از متداولترین برنامه‌ها هستند. قابل ذکر است که بیشتر ویراستارهای متداول HTML دست کم به نوعی PHP را نیز پشتیبانی می‌کنند.

> Keith Edmunds لیست مفیدی از ویراستارهای PHP کاربر پسند را در آدرس زیر نگهداری می‌کند:
>
> http: // www. Itworks. Demon. Co. uk / phpeditors. htm

برنامه موجوددرلیست ۱- ۳ را در ویراستار مورد نظر تایپ کرده و آن‌را با نام First. Php بر روی سیستم خود ذخیره کنید.

```
1: <?php
2:     print "Hello Web!";
3: ?>
```

لیست ۱- ۳ اولین برنامه اسکریپت

وجود پسوند php. برای این فایل ضروری است، چرا که در این صورت وب سرور آن‌را به‌عنوان یک فایل برنامه PHP تلقی کرده و برای پردازش آن موتور PHP را فرا می‌خواند. پسوند پیش‌فرض برای اسناد PHP4 به صورت php. مشخص می‌شود. با این وجود چنین پیش‌فرضی را می‌توان با پیکربندی مجدد وب سرور تغییر داد. چگونگی انجام این کار در درس ساعت دوم با عنوان " نصب PHP بر روی کامپیوتر" آمده است. برخی از مدیران سیستمها گاه‌گاهی پیکربندی وب سرورها به را گونه‌ای که با پسوندهای غیر پیش‌فرض نیز قابل استفاده باشند، تغییر می‌دهند. از این‌رو برخی از سرورها انتظار پسوندهایی چون phtml . یا php4 . را دارند.

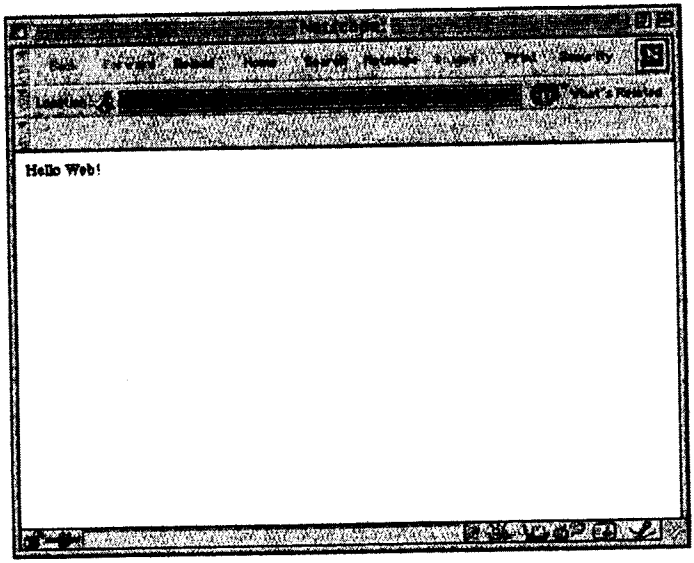
اگر مستقیماً بر روی کامپیوتری که وظیفه سرویس‌دهی را دارد، برنامه‌نویسی نمی‌کنید، به برنامه‌ای از نوع FTP Client مانند برنامه WS - FTP برای سیستمهای نوع Windows یا از برنامه Fetch برای سیستمهای نوع Mac OS جهت بارگذاری (upload) سندی که اخیراً با عنوان first. php ایجاد کرده‌اید، نیاز خواهید داشت.

به‌دلایل تاریخی، سیستمهای عامل مختلف از ترکیبات کاراکتری متفاوتی برای نشان دادن انتهای خطی از یک فایل متن (موسوم به End Of Line یا EOL) استفاده می‌کنند. همواره سعی کنید این عمل را با توجه به وب سروری که در نهایت اسناد PHP ایجاد شده را بر روی آن بارگذاری می‌کنید، انجام دهید. در صورتی‌که این مطلب را رعایت نکنید، امکان زیادی وجود دارد که کد برنامه PHP به‌عنوان یک خط طولانی از رشته‌های کاراکتری توسط موتور PHP پردازش شود. این موضوع معمولاً به‌خودی خود مشکلی پیش نمی‌آورد. اما در مواردی نیز ممکن است شما را با اشکال روبرو کند. ویراستارهای متنی خوب معمولاً به‌کاربر خود اجازه می‌دهند تا این ترکیب کاراکتری را با تعیین سیستم عامل مقصد مشخص کنند.

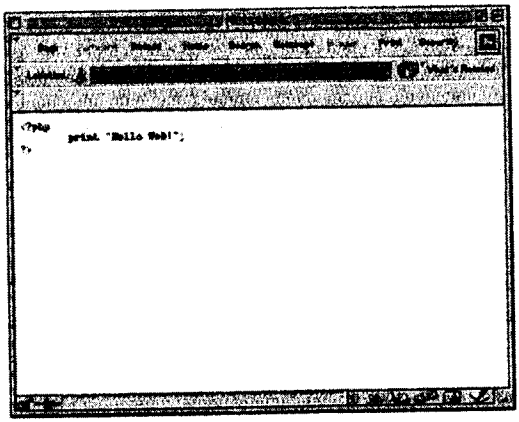
پس از بارگذاری سند در محل مورد نظر بر روی سرور، باید بتوانید از طریق مرورگر اینترنت به آن دسترسی داشته باشید. اگر تمامی مراحل کار را به‌درستی پشت سر گذاشته باشید، هم‌اینک قادر خواهید بود تا محتوای این سند را مشاهده کنید.

شکل ۱- ۳ خروجی برنامه اسکریپت First. Php را نشان می‌دهد.

در صورتی که PHP بر روی سیستم میزبان نصب نشده باشد و یا پسوند فایل برنامه به خوبی توسط وب سرور مورد استفاده تشخیص داده نشود، خروجی شکل ۱- ۳ را به‌درستی مشاهده نخواهید کرد. در چنین مواردی آن‌چه را که مشاهده می‌کنید به‌احتمال زیاد کد منبعی است که در لیست ۱- ۳ پیش‌تر آن‌را مشاهده کردید. شکل ۲- ۳ وضعیتی را نشان می‌دهد که در آن پسوند فایل PHP به‌درستی توسط وب سرور قابل تشخیص نبوده است.



شکل ۱- ۳ خروجی موفقیت آمیز حاصل از اجرای کد منبع لیست ۱- ۳



شکل ۲- ۳ خروجی حاصل از عدم تشخیص پسوند فایل PHP توسط وب سرور

در صورت مشاهده چنین وضعیتی ابتدا پسوند فایلی را که شامل کد برنامه First . Php است و اخیراً آن را بر روی‌کامپیوترتان ذخیره کرده‌اید، مورد بازبینی قرار دهید. همان‌گونه که مشاهده می‌کنید، در شکل ۲- ۳ فرض براین بوده که برنامه‌نویس به اشتباه این برنامه را با عنوان First . Nphp ذخیره کرده است. اگر پسوند فایل درست باشد، لازم است تا از صحت نصب PHP و همچنین این مطلب که وب سرور را برای کار با پسوند مورد استفاده‌تان پیکربندی کرده‌اید، مطمئن شوید. برای یک بحث مفصل درباره نحوه نصب و پیکربندی PHP و وب سرور می‌توانید به مطالب ساعت قبل رجوع کنید.

اکنون که این برنامه را بارگذاری کرده و مورد بررسی قرار دادید، می‌توانیم نگاهی دقیق‌تر به این برنامه بیندازیم و اجزای تشکیل دهنده آن را از نزدیک بررسی کنیم.

نحوه شروع و پایان دادن به بلوکی از برنامه‌های PHP

هنگام نوشتن برنامه‌های PHP لازم است تا مراتب را جهت اجرای دستورالعملهای برنامه نوشته شده به موتور PHP اطلاع دهید. اگر چنین کاری را صورت ندهید، برنامه شما به‌عنوان یک سند HTML به‌صورت یک متن ساده بر روی مرورگر اینترنت مورد استفاده‌تان به نمایش درخواهد آمد. برای اطمینان از این مطلب و جلوگیری از وضعیت فوق از علایم ویژه‌ای (که معمولاً با نام تگ آنها را می‌شناسیم) جهت مشخص کردن ابتدا و انتهای بلوک کد PHP استفاده می‌کنیم. جدول ۱-۳ چهارنوع از این علایم را نشان می‌دهد.

جدول ۱- ۳ علایم شروع و پایان بلوک‌های PHP

نوع علامت	علامت شروع	علامت پایانی
علایم استاندارد	< ? PHP	? >
علایم کوتاه	< ?	? >
علایم ASP	< %	% >
علایم Script	< SCRIPT LANGUAGE = "php">	< / SCRIPT >

از مجموعه علایم موجود در این جدول تنها برای علایم استاندارد و علایم Script تضمین اجرایی برای هر نوع پیکربندی وجود دارد. علایم دو گروه دیگر یعنی مجموعه علایم ASP و علایم کوتاه تنها زمانی قابل استفاده‌اند که گزینه‌های مربوطه در فایل php. ini به‌گونه‌ای مناسب تنظیم شوند. مطالب مربوط به دستکاری گزینه‌های فایل PHP. ini در درس ساعت دوم آمده است.

جهت بهره‌گیری از علایم کوتاه لازم است تا فایل php. ini را در یک ویراستار متن باز کرده و گزینه‌ای با عنوان Short _ open _ tag را به صورت زیر با مقدار " on " تنظیم کنید:

```
Short_ open _ tag = On ;
```

البته این تنظیم به‌طور پیش‌فرض وجود دارد. لذا فایل مزبور را تنها برای غیر فعال کردن علایم کوتاه به‌صورت زیر مورد ویرایش قرار دهید:

```
Short _ Open _tag = Off ;
```

همچنین جهت فعال‌سازی و بهره‌گیری از علایم ASP لازم است تا گزینه‌ای با عنوان asp_ tags را به شکل زیر تنظیم کنید:

```
asp _ tags = on ;
```

پس از اعمال تغییرات مورد نیاز در فایل php. ini می‌توانید از علایم موجود در هریک از این چهارگروه جهت تعیین نقاط آغازین و پایانی بلوک‌های برنامه PHP استفاده کنید. معمولاً اکثر برنامه نویسان PHP دست به انجام چنین کاری می‌زنند. با این حال اگر برنامه نویسی قصد بهره‌گیری از کد XML را در درون کدهای برنامه PHP داشته باشد باید استفاده از علایم کوتاه (< ? ? >) را غیر فعال کرده و با علایم استاندارد (< ? php ? >) کار کند.

> دنباله‌کاراکتری? > به برنامه‌های تجزیه‌کننده اسناد XML (موسوم به XML Parser) وجود دستورالعملهای پردازشی را اطلاع می‌دهد. لذا از این علامت به‌طور متناوب در اسناد XML استفاده می‌شود. اگر بهره‌گیری از کد XML با فعال بودن علایم کوتاه توام باشد، به احتمال قوی موتور PHP دستورالعملهای پردازشی XML را با علایم شروع کد PHP به اشتباه خواهد گرفت. از این رو در صورت استفاده از XML در درون سندتان از غیر فعال بودن علایم کوتاه اطمینان حاصل کنید.

اجازه دهید تا در این قسمت با استفاده از علایم ذکر شده در جدول ۱- ۳ کد برنامه لیست ۱-۳ را به چند روش دیگر باز نویسی کنیم. برای این‌کار می‌توانید از علایم هر یک از چهارگروه ذکر شده استفاده کنید:

روش اول:

```
< ?
print (" Hello Web! ") ;
? >
```

روش دوم:

```
< ? php
print ( " Hello Web! " ) ;
? >
```

روش سوم:

```
< %
print (" Hello Web ! ") ;
% >
```

و روش چهارم:

```
< SCRIPT LANGUAGE = " php " >
print ( " Hello Web ! ") ;
< / SCRIPT >
```

همچنین می‌توان خط ساده‌ای شامل کد PHP را در همان خطی که علائم شروع و پایانی وجود دارند مورد استفاده قرار داد. چگونگی انجام این کار چنین است:

```
< ? Print (" Hello Web ! ") ; ? >
```

اکنون که با چگونگی تعریف بلوکی از کد برنامه PHP آشنا شدید، قصد داریم تا نگاه دقیق‌تری به کد برنامه موجود در لیست ۱- ۳ بیندازیم.

تابع ( ) Print

تابع ( ) Print موجب نمایش داده‌ها در خروجی می‌شود. در بیشتر موارد خروجی این تابع به سادگی بر روی پنجره مرورگر به نمایش در می‌آید. تابع در واقع فرمانی است که موجب انجام یک عمل می‌شود، و معمولاً این عمل تغییری است که بر روی داده‌ها می‌دهد (این داده‌ها در بیشتر موارد همان آرگومان‌هایی هستند که تابع آنها را دریافت می‌کند). داده‌هایی که در قالب آرگومان به تابع ارسال می‌شوند، بعد از نام تابع و در داخل یک جفت پرانتز قرار می‌گیرند. در مورد تابع ( ) Print ، رشته‌ای از کاراکترها را به‌عنوان آرگومان به این تابع ارسال می‌کنیم. به‌خاطر داشته‌باشید که رشته‌های کاراکتری را همیشه در داخل علامت کوتیشن قرار دهید (یعنی جفت علامت " " یا ' ').

> فراخوانی تابع عموماً مستلزم یک جفت پرانتز بعد از نام آن تابع بوده و این امر به این موضوع که آیا تابع دارای آرگومان می‌باشد یا خیر بستگی ندارد. تابع ( )Print در این میان یک استثنا می‌باشد، بدین معنی که رشته کاراکتری که به‌عنوان آرگومان به این تابع ارسال می‌شود، نیازی نیست تا در درون جفت پرانتز محصور شود. بیشتر برنامه‌نویسان از این استثنا بهره می‌برند. ما نیز در مثالهای کتاب از این روش پیروی می‌کنیم.

همان‌گونه که در لیست ۱- ۳ مشاهده می‌کنید، تنها خط موجود در برنامه با علامت سمی‌کولون به پایان رسیده‌است. این علامت پایان تمامی عبارات موجود در برنامه‌های PHP را به موتور PHP اطلاع می‌دهد.

واژه جدید عبارت یا Statement نماینده یک دستورالعمل برای موتور PHP می‌باشد. در واقع عبارت برای موتور PHP مشابه معنی جمله در زبان انسانها است. همان‌گونه که جملات در زبان ما توسط علامت نقطه به پایان می‌رسند، در مورد عبارات نیز باید چنین قاعده‌ای موجود باشد؛ خاتمه

عبارات در زبان PHP معمولاً با علامت سمی‌کولون (;) مشخص می‌شود. البته عباراتی که در داخل عبارات بزرگ‌تر مورد استفاده قرار می‌گیرند و نیز عباراتی که بلوکی از کد را خاتمه می‌دهند (آخرین عبارت موجود در هر کد PHP) از این قاعده مستثنا هستند. با این حال در بیشتر موارد، عدم رعایت این قاعده موجب می‌شود تا موتور PHP به اشتباه افتاده و در نتیجه خطایی رخ دهد.

همان‌گونه که در کد لیست برنامه ۱- ۳ نیز مشاهده می‌کنید، به‌دلیل اینکه کد مزبور تنها شامل یک خط برنامه است، استفاده از علامت سمی‌کولون اختیاری است.

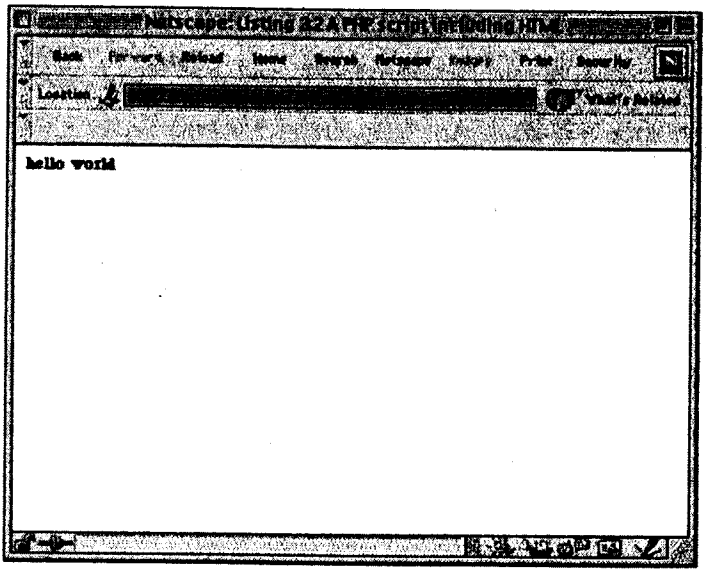
ترکیب کدهای HTML و PHP

برنامه لیست ۱- ۳ به‌طور خالص یک برنامه PHP است. با اضافه کردن کد HTML در خارج از بلوکی که توسط علایم شروع و پایان php ? > و < ? مشخص شده‌است، به‌سادگی می‌توان به سندی دست یافت که شامل ترکیبی از کد PHP و کد HTML باشد. چگونگی انجام این عمل در لیست ۲- ۳ مشخص شده است.

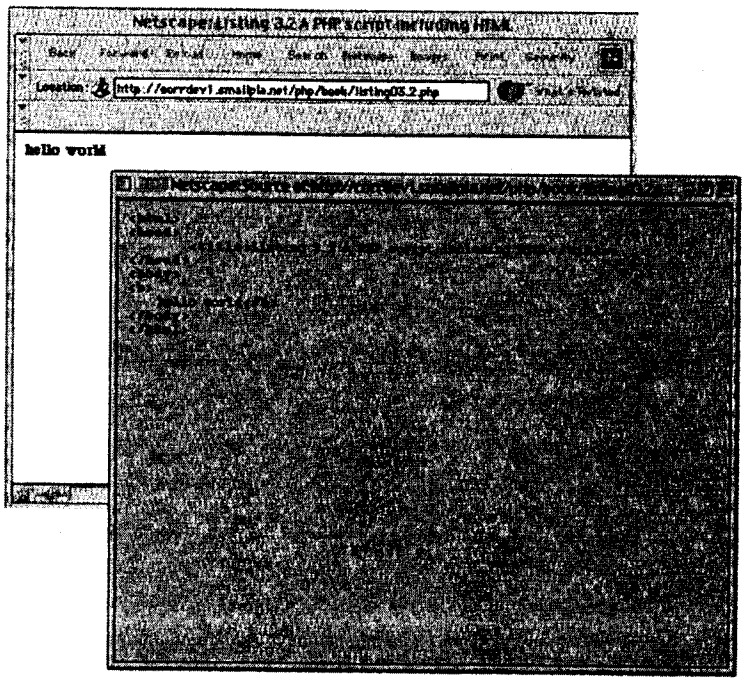
```
 1: <html>
 2: <head>
 3:     <title>Listing 3.2 A PHP script including HTML</title>
 4: </head>
 5: <body>
 6: <b>
 7: <?php
 8:     print "hello world";
 9: ?>
10: </b>
11: </body>
12: </html>
```

لیست ۲- ۳ ترکیب کدهای PHP و HTML

همان‌گونه که مشاهده می‌کنید ترکیب کد HTML در داخل یک سند PHP به سادگی تایپ کد مورد نظر در خارج از محدوده کد PHP امکان پذیر است. موتور PHP کد موجود در خارج از محدوده فوق، یعنی محدوده‌ای که توسط علایم شروع و پایانی php ? > و < ? را نادیده می‌گیرد. چنان‌چه کد لیست ۲- ۳ را در مرورگر اینترنت خود باز کنید، خواهید دید که عبارت hello world به صورت bold به نمایش در می‌آید. شکل ۳- ۳ این وضعیت را نشان می‌دهد. شکل ۴- ۳ نیز کد منبع این سند را که در مرورگر اینترنت باز شده است، به نمایش گذاشته است.



شکل ۳- ۳ خروجی حاصل از لیست شماره ۲- ۳ در یک مروگر اینترنت



شکل ۴- ۳ نمایش کد لیست شماره ۲- ۳ به‌عنوان کد منبع HTML در یک مرورگر اینترنت

به تعداد مورد نیاز می‌توان در یک سند واحد بلوک‌های متعددی از کد PHP را مورد استفاده قرار داد و در صورت نیاز هر سند را با کد HTML ترکیب نمود. هر چند که وجود چندین بلوک کد PHP در یک سند واحد امکان‌پذیر است اما در اینجا خاطر نشان می‌کنیم که ترکیب تمامی آنها تنها یک برنامه اسکریپت را به‌وجود می‌آورد. بدین ترتیب متغیرهای تعریف شده در اولین بلوک معمولاً در بلوک‌های بعدی مورد استفاده قرار می‌گیرند.

بهره‌گیری از توضیحات در برنامه‌های PHP

برنامه‌ای که در زمان حاضر آن‌را می‌نویسید به احتمال بسیار قوی واضح و خوانا می‌باشد اما پس از گذشت مدت زمانی حتی نویسنده آن نیز در درک و فهم برخی قسمتهای آن با مشکل مواجه خواهد شد. از این‌رو بهره‌گیری از توضیحات به‌میزان زیادی در خوانایی برنامه‌هایتان توسط سایر برنامه‌نویسان و حتی خود شما بسیار مفید و مؤثر خواهد بود و در وقت و هزینه زیادی صرفه‌جویی خواهد کرد.

واژه جدید توضیح یا Comment متونی در داخل برنامه‌های اسکریپت هستند که موتور PHP به سادگی آنها را نادیده می‌گیرد. توضیحات نقش ارزنده‌ای در افزایش میزان خوانایی و وضوح برنامه‌ها داشته و موجب صرفه‌جویی در وقت و هزینه می‌شوند.

در زبان PHP استفاده از دو گونه توضیح یک خطی و چند خطی مرسوم است.

توضیح یک خطی با دو علامت متوالی / به صورت // و یا یک علامت # آغاز می‌شود و در انتهای همان خط نیز به پایان می‌رسد (برنامه‌نویس می‌تواند خاتمه این نوع توضیح را با علامت نشانه پایان بلوک برنامه PHP نیز مشخص نماید). به نمونه‌های زیر توجه کنید:

```
// this is a comment
# this is another comment
```

از طرف دیگر توضیحات چندخطی با ترکیب دو علامت * و / به صورت * / آغاز شده و با ترکیب همین دو علامت به صورت / * خاتمه می‌یابند. موتور PHP هر چیزی مابین این دو ترکیب را نادیده می‌گیرد. به توضیح چند خطی زیر توجه کنید:

```
/*
this is a comment
none of this will
be parsed by the
php Engine.
*/
```

با استفاده از یک برنامه نرم‌افزاری ویژه به‌نام PHP Doc می‌توان توضیحات یک برنامه PHP را به اسنادی از نوع فوق پیوند مانند اسناد HTML تبدیل نمود (این برنامه به‌همراه نرم‌افزار PHP توزیع نمی‌شود). این امکان برای پروژه‌های بزرگ برنامه‌نویسی بسیار مفید می‌باشد. جهت دستیابی به این نرم‌افزار به وب سایت مربوطه به نشانی زیر مراجعه کنید:

http: // www. Phpdoc. de

جمع‌بندی

هم‌اکنون باید تمامی ابزارهای مورد نیاز را جهت اجرای برنامه‌های اسکریپت PHP بر روی وب سروری با پیکربندی مناسب در اختیار داشته باشید.

در درس این ساعت اولین برنامه اسکریپت PHP را با هم ایجاد کردیم. طی این فرآیند شما با چگونگی به‌کارگیری و استفاده از یک ویراستار متن جهت ایجاد و نام‌گذاری و ذخیره یک سند PHP آشنا شدید. همچنین چهار مجموعه علایم ویژه‌ای را که جهت مشخص کردن نقاط شروع و پایان بلوک‌های PHP از آنها استفاده می‌کنید را به کار بردید. علاوه بر این چگونگی استفاده از تابع ( )Print جهت ارسال داده‌ها به مرورگر اینترنت و نیز چگونگی ترکیب کد PHP با کد HTML را در یک سند واحد فراگرفتید و در نهایت مطالبی را در مورد نحوه درج توضیحات و میزان مفید بودن آنها در اسناد PHP ،مورد توجه قرار دادید.

دردرس‌ساعت آینده از مهارتهایی که در این درس فراگرفتید، استفاده کرده و برخی از سازه‌های اساسی زبان برنامه‌نویسی PHP شامل متغیرها، انواع داده‌ها و عملگرها را مورد بررسی قرار خواهیم داد.

پرسش و پاسخ

پرسش: کدام جفت علامت شروع و پایان بلوک کد PHP بر سایر علامتها برتری دارد؟

پاسخ: در انتخاب این جفت علامت عوامل متعددی را باید در نظر گرفت. از دیدگاه قابلیت حمل برنامه استفاده از علایم استاندارد (php ? > و < ?) اطمینان خاطر بیشتری می‌دهد. علایم کوتاه به‌طور پیش‌فرض فعال بوده و از نظر اختصار بر سایر علایم برتری دارند، اما به واسطه حضور روزافزون XML در عرصه برنامه‌نویسی وب توصیه می‌کنیم که از این علایم صرف‌نظر کنید.

پرسش: از کدام ویراستارهای متن نباید به‌منظور کد نویسی PHP استفاده کرد؟

پاسخ: در نوشتن برنامه‌های PHP از پردازش‌گرهای متن که متون را جهت ارسال به چاپ‌گر قالب‌بندی می‌کنند (مانند Word) خودداری کنید. حتی در صورتی که برنامه‌های تایپ شده در محیط این پردازشگرها را با عنوان متن ساده (Plain text) ذخیره کنید همواره کاراکترهایی به‌صورت پنهان به بخش‌هایی از متن ساده شما اضافه خواهند شد.

پرسش: بهره‌گیری از توضیحات در برنامه‌های PHP چه‌هنگام ضرورت دارد؟

پاسخ: در این‌جا نیز عوامل متعددی را باید در نظر گرفت. برخی از برنامه‌های کوتاه حتی پس از گذشت مدت زمان طولانی از موعد نوشتن آنها باز هم کاملاً خوانا بوده و درک آنها به‌سادگی صورت می‌گیرد. با این همه برای تمامی برنامه‌ها با هر میزان از حجم و پیچیدگی کد لازم است تا از توضیحات استفاده گردد. این مطلب اغلب در آینده باعث صرفه‌جویی در هزینه و زمان برنامه‌نویس خواهد شد.

تمرینها

هدف از این بخش دوره مطالب فراگیری شده در قالب آزمون است. پاسخ بخش آزمون بلافاصله بعدازآن‌آمده است. بخش فعالیتها شامل تمرینهایی است که به‌منظور افزایش قابلیتهای برنامه‌نویسی خواننده طراحی شده و البته فاقد پاسخ است.

آزمون

۱- آیا کاربران می‌توانند کد منبع برنامه‌های PHP نوشته شده را مشاهده کنند؟
۲- علایم استاندارد جهت تعیین حدود بلوکی از کد PHP کدام است؟
۳- علایم ASP جهت تعیین حدود بلوکی از کد PHP کدام است؟
۴- علایم Script جهت تعیین حدود بلوکی از کد PHP کدام است؟
۵- کدام تابع PHP جهت ارسال خروجی برنامه‌های PHP بر روی صفحه مرورگر اینترنت مورد استفاده قرار می‌گیرد؟

پاسخ آزمون

۱- خیر. کاربران تنها می‌توانند خروجی حاصل از اجرای برنامه‌ها را مشاهده کنند.
۲-

```
< ? php
// your code
? >
```

۳-

```
< %
// your code
```

```
%>
```

۴-

```
< SCRIPT LANGUAGE = " php " >
// your code
< / SCRIPT >
```

۵- تابع print ( )

فعالیتها

۱- فرآیند ایجاد، بارگذاری و اجرای برنامه‌های PHP را مجدداً مورد بازبینی و مطالعه قرار دهید. سعی کنید تا برنامه‌ای را مجدداً جهت نمایش رشته کاراکتری " hello world " بر روی پنجره مرورگر اینترنت توسعه داده و کدهایی از نوع HTML به آن اضافه کنید. همچنین بلوک‌های دیگری از PHP را به این بلوک از برنامه اضافه نمایید. از علایم تشریح شده در این درس جهت تعیین حدود بلوک‌های مذکور استفاده کنید. کدام دسته از علایم به‌طور پیش‌فرض در پیکربندی شما پشتیبانی شده است؟ جهت تأیید پاسخ خود نگاهی به فایل php. ini بیندازید. توضیحاتی را به برنامه خود اضافه نمایید.

« بخش دوم: اجزای زبان PHP »

- ساعت چهارم: اجزای سازنده زبان PHP
- ساعت پنجم: اجزای کنترل برنامه
- ساعت ششم: توابع
- ساعت هفتم: آرایه‌ها
- ساعت هشتم: اشیا

ساعت چهارم

اجزای سازنده زبان PHP

در درس این ساعت آستینها را بالا زده و با اجزای سازنده زبان برنامه‌نویسی PHP از نزدیک دست و پنجه نرم خواهید کرد.

جهت حصول این هدف مطالب زیادی را تحت پوشش و بررسی قرار می‌دهیم. اگر به‌تازگی کار برنامه‌نویسی را آغاز کرده‌اید، گمان خواهید کرد که با انبوهی از اطلاعات بمباران شده‌اید. اما نگران نباشید، چرا که همواره می‌توانید از نقطه شروع مجددا کار خود را آغاز کنید. توصیه می‌کنیم که به جای حفظ و به‌خاطر سپردن مطالب، بیشتر بر روی درک مفاهیم اساسی تکیه کنید.

چنان‌چه برنامه‌نویس باتجربه‌ای هستید، می‌توانید مطالب را به‌سرعت دوره کرده و ویژگی‌های مختص برنامه‌نویسی با PHP را مورد توجه قرار دهید.

در درس این ساعت مطالب زیر را مورد بررسی قرار می‌دهیم:

- **بررسی متغیرها (این بررسی شامل ماهیت متغیرها، لزوم استفاده از آنها و همچنین نحوه بهره‌گیری و استفاده از آنها می‌شود)**
- **چگونگی تعریف و دستیابی به متغیرها**
- **بررسی انواع داده‌ها**
- **بررسی برخی از متداول‌ترین عملگرها**
- **بررسی چگونگی استفاده از عملگرها جهت ایجاد عبارات مختلف**
- **بررسی چگونگی استفاده از ثابتها (مقادیر ثابت)**

در ادامه به بررسی موارد فوق می‌پردازیم.

متغیرها

متغیر ظرف ویژه‌ای است که می‌توان از آن جهت نگهداری یک مقدار استفاده نمود. متغیرها از ارکان اصلی برنامه‌نویسی محسوب می‌شوند. بدون وجود آنها بهره‌گیری از مقادیر مورد نیاز در برنامه به فرآیند بسیار سختی برای برنامه‌نویس مبدل می‌شد. اجازه دهید تا برای روشن شدن مطلب موضوعی را مطرح کنیم. در جمع دو عدد با یکدیگر و چاپ نتیجه حاصل معمولاً استفاده از چنین عبارتی را در نظر می‌آوریم:

```
Print (2 + 4) ;
```

با این وجود عبارت فوق تنها برای کسانی که حاصل جمع دو عدد ۴ و ۲ را می‌دانند مفید خواهد بود. به‌صورت مشابه جهت محاسبه حاصل جمع دو عدد دیگر مثل محاسبه مجموع ۳ و ۵ می‌توان عبارت دیگری مثل عبارت فوق نوشت. واضح است که چنین رویکردی برای برنامه‌نویس بسیار کسل‌کننده است. به‌کمک متغیرها می‌توانیم الگوهایی را جهت عملیات مختلف (مانند محاسبه حاصل جمع دو عدد)، بدون نگرانی در مورد محتوای این متغیرها ایجاد کنیم. هنگام اجرای برنامه‌های اسکریپت متغیرها از طریق مقادیری که کاربران وارد می‌کنند یا از طریق پرس و جوهایی که از یک بانک اطلاعاتی می‌کنیم، مقداردهی می‌شوند.

جایگاه استفاده از متغیرها هنگامی است که بخواهیم از اجرای یک برنامه اسکریپت به اجرای دیگر همان اسکریپت و یا حتی در طول حیات اجرای یک اسکریپت تنها مقادیر متفاوتی از داده‌ها را مورد بهره‌برداری قرار دهیم. متغیرها این امکان را می‌دهند که مقادیر مورد نظرمان را ذخیره کرده و از آنها به‌دلخواه استفاده کنیم.

متغیرها در زبان PHP با انتخاب یک نام دلخواه که در ابتدای آن از علامت دلار ($) استفاده شده است، ایجاد می‌شوند. اسامی متغیرها می‌تواند شامل حروف، اعداد و کاراکتر underscore علامت (_) باشد. دقت کنید که استفاده از فضای خالی و همچنین کاراکترهای غیر الفبا عددی برای نام‌گذاری متغیرها امری غیر مجاز است. همچنین به‌خاطر داشته باشید که اولین کاراکتر نام متغیرها (البته بعد از کاراکتر $) باید یک حرف الفبا یا علامت underscore باشد. اسامی زیر برای نام‌گذاری متغیرهای PHP نمونه‌هایی مجاز هستند:

```
$ a ;
$ a _ Longish _ varaiable _ name;
$ 2453 ;
$ sleepy ZZZZ ;
```

به‌خاطر داشته باشید که علامت سمی‌کولون جهت پایان دادن به عبارات PHP مورد استفاده قرار می‌گیرد. علامت مذکور در قطعه کد بالا جزو اسامی متغیرها نمی‌باشد.

واژه جدید متغیر (Variable) محلی برای نگهداری داده‌هاست. از متغیرها می‌توان جهت نگهداری داده‌هایی از نوع عددی، دنباله‌های کاراکتری، اشیا، آرایه‌ها و یا مقادیر دودویی boolean بهره‌گرفت. محتوای متغیرها می‌تواند در طول اجرای برنامه تغییر کند.

همان‌گونه که مشاهده می‌کنید دست برنامه‌نویس برای انتخاب اسامی متغیرهای برنامه بسیار باز گذاشته شده است. جهت معرفی متغیر در برنامه‌های PHP کافی است از آن در برنامه خود استفاده کنید. برنامه‌نویسان معمولاً فرآیند معرفی و مقدار دهی متغیرها را توأماً در یک عبارت واحد انجام می‌دهند. به دو نمونه زیر توجه کنید:

```
$num 1= 8 ;
$num 2 = 23 ;
```

دو نمونه فوق دو متغیر معرفی شده و با استفاده از عملگر انتساب (=) مقدار دهی شده‌اند. در مورد فرآیند انتساب بعداً در درس همین ساعت به‌طور مفصل در بخش مربوط به " عملگرها و عبارات PHP " بحث خواهیم کرد. پس از مقداردهی متغیرها با مقادیر مورد نظر می‌توان آن متغیرها را دقیقاً معادل همان مقادیر فرض کرده و با آنها به همان ترتیب که با مقادیر رفتار می‌کنیم، برخورد کرد. برای نمونه عبارت زیر را در نظر بگیرید:

```
Print $num 1;
```

عبارت فوق دقیقاً مشابه عبارت زیر است:

```
Print 8 ;
```

تنها به شرطی که متغیر $ num1 شامل مقدار عددی 8 باشد.

انواع داده‌ها

انواع مختلف داده‌ها جهت ذخیره سازی به میزانهای مختلفی از حافظه نیاز داشته و در قالب یک برنامه اسکریپت به‌طور جداگانه و متفاوتی از یکدیگر مورد استفاده قرار می‌گیرند. از این‌رو در برخی از زبانهای برنامه‌نویسی لازم است تا نوع داده‌ای را که متغیرهای برنامه قصد ذخیره و نگهداری آنها را دارند از پیش مشخص کنیم. در مقابل، زبان برنامه‌نویسی PHP4 چنین لزومی را به برنامه‌نویس تحمیل نکرده و بنابراین برنامه‌نویس نیازی به تعیین نوع داده‌هایی که در متغیرهای برنامه ذخیره خواهد کرد، ندارد (به زبانهای برنامه‌نویسی دسته اول tightly typed یا strongly typed و به زبانهای دسته دوم از این نظر loosely typed گفته می‌شود ـ مترجم). این ویژگی نقاط قوت و ضعفی دارد. از یک طرف استفاده از متغیرها در زبان PHP با انعطاف و آزادی عمل زیادی همراه است، بدین معنی که یک متغیر ممکن است در جایی از برنامه برای نگهداری دنباله‌ای از کاراکترها و در جایی دیگر از برنامه برای نگهداری عددی صحیح مورد استفاده قرار بگیرد. از سویی دیگر، همین ویژگی انعطاف و آزادی عمل می‌تواند موجب بروز مشکلاتی در برنامه‌های بزرگ شود بدین صورت که برخلاف انتظار

برنامه‌نویس که تصور می‌کند داده‌ای از نوع رشته کاراکتری در متغیری ذخیره شده است متغیر مورد نظر شامل داده‌ای از نوع عدد صحیح یا از نوع دیگر باشد. برای مثال فرض کنید برنامه شما شامل کدی است که جهت بهره‌گیری از آرایه‌ها طراحی شده و به منظور ذخیره آرایه مورد نظرتان، از متغیری در آن برنامه استفاده کرده‌اید. چنان‌چه این متغیر در عوض شامل یک مقدار عددی باشد هنگامی که برنامه سعی دارد تا عملیات پردازش مخصوص آرایه‌ها را بر روی محتوای این متغیر انجام دهد به احتمال بسیار قوی برنامه‌نویس با مشکلاتی مواجه می‌گردد.

جدول ۱- ۴ شش نوع داده استاندارد قابل استفاده در زبان PHP4 را نشان می‌دهد.

جدول ۱- ۴ انواع داده‌های استاندارد در زبان PHP4

نوع داده	مثال	توضیح
Integer	5	عدد صحیح
Double	3.234	عدد اعشاری
String	" hello "	دنباله کاراکتری
Boolean	true	یکی از مقادیر true یا False
Object		مراجعه شود به درس هشتم
Array		مراجعه شود به درس هفتم

از مجموع این شش نوع داده استاندارد در PHP بررسی دو نوع Array و Object را به‌ترتیب به درس ساعت هفتم و هشتم موکول می‌کنیم و بررسی سایر انواع داده‌ها را در همین درس انجام می‌دهیم.

زبان PHP4 شامل دو نوع داده ویژه دیگر می‌باشد. اسامی این دو نوع داده و شرح مخصوص به هر یک از آنها در جدول ۲- ۴ آمده است.

جدول ۲- ۴ انواع داده‌های ویژه در زبان PHP4

نوع داده	توضیح
Resource	مرجعی به یک منبع خارجی مانند یک بانک اطلاعاتی (خارج از PHP4)
NULL	متغیری که مقداردهی نشده است.

مقادیری با نوع داده Resource معمولاً حاصل اجرای توابعی هستند که با برنامه‌های کاربردی خارجی یا فایل‌ها سر و کار دارند. در درس ساعت دهم، یازدهم و دوازدهم مثال‌ها و برنامه‌هایی را که با این نوع از داده‌ها در ارتباط هستند، بیان خواهیم کرد. نوع داده NULL به متغیرهایی اشاره دارد که مقدار دهی نشده‌اند (بدین معنی که پس از معرفی آنها هیچ مقداری به آنها منسوب نشده است).

با استفاده از تابع سیستمی ویژه‌ای در PHP با نام ( ) gettype می‌توان نوع داده هر متغیری را مورد تشخیص و بررسی قرار داد. چنان‌چه نام متغیری را به‌عنوان ورودی به این تابع بدهید، تابع مزبور رشته‌ای از کاراکترها را که بیانگر نوع داده آن متغیر است به خروجی خواهد داد. همان‌گونه که مشاهده می‌کنید، در لیست برنامه ۱- ۴ پنج نوع داده مختلف به یک متغیر تنها منسوب شده و با استفاده از تابع مذکور نوع داده هر یک از مقادیر مشخص شده است.

دردرس‌روز ششم مطالب مفیدی در مورد چگونگی فراخوانی توابع خواهید آموخت.

```
1: <html>
2: <head>
3: <title>Listing 4.1 Testing the type of a variable</title>
4: </head>
5: <body>
6: <?php
7: $testing; // declare without assigning
8: print gettype( $testing ); // null
9: print "<br>";
10: $testing = 5;
11: print gettype( $testing ); // integer
12: print "<br>";
13: $testing = "five";
14: print gettype( $testing ); // string
15: print("<br>");
16: $testing = 5.0;
17: print gettype( $testing ); // double
18: print("<br>");
19: $testing = true;
20: print gettype( $testing ); // boolean
21: print "<br>";
22: ?>
23: </body>
24: </html>
```

لیست ۱- ۴ تشخیص نوع داده یک متغیر

اجرای این برنامه خروجی زیر را در پی دارد:

```
NULL
integer
string
double
boolean
```

کاملاً واضح است که در خط ۷ از این برنامه متغیری با نام $testing معرفی شده ولی هیچ مقداری به آن منسوب نشده است. بنابراین هنگامی که تابع gettype( ) برای اولین بار در خط ۸ فراخوانی می‌شود، نتیجه مورد انتظار یعنی رشته کاراکتری " NULL " حاصل می‌شود. پس از آن با استفاده از عملگر انتساب (=) مقادیری را پیش از ارسال متغیر $testing به تابع gettype( ) به آن منسوب کرده‌ایم. در خط ۱۰ از برنامه، مقداری از نوع integer به متغیر $testing منسوب شده است که بیان‌گر یک عدد صحیح یا در کل بیان‌گر عددی است که فاقد نقطه اعشار می‌باشد. در خط ۱۳ از برنامه مقداری از نوع String به متغیر $testing منسوب شده است. این مقدار شامل رشته‌ای از چندین کاراکتر است. به‌خاطر داشته باشید، که هنگام استفاده از رشته‌های کاراکتری در برنامه‌های PHP آنها را در داخل یک جفت علامت " " یا ' ' قرار دهید. در خط ۱۶ از برنامه، مقداری از نوع double به متغیر $testing منسوب شده است. این مقدار در حقیقت عددی است که دارای نقطه اعشاری می‌باشد. (به‌عبارت ساده‌تر یک عدد اعشاری) و در نهایت در خط ۱۹ از این برنامه کوتاه مقداری دودویی (یکی

از دو مقدار true یا false) را به متغیر $testing نسبت داده‌ایم. همان‌گونه که مشاهده می‌کنید تابع gettype( ) وجود نوع داده boolean را تأیید کرده است.

> نسخه‌های پیش از PHP4 مقادیری با نوع داده boolean را پشتیبانی نمی‌کردند. هر چندکه از مقدار true در این نسخه استفاده می‌شد، اما درواقع یک مقدار ثابت به‌حساب می‌آمد (مقادیر ثابت نوع خاصی از متغیرها هستند که بعداً در درس همین ساعت راجع به آنها صحبت خواهیم کرد). مقدار این ثابت برابر با عدد 1 بود.
> همچنین انواع داده‌های Resource و NULL نیز از نسخه PHP4 و از آن به بعد در این زبان معرفی شده‌اند.

تغییر نوع داده‌ها با استفاده از تابع Settype( )

در زبان PHP تابعی با عنوان Settype( ) وجود دارد که با استفاده از آن می‌توان نوع داده یک متغیر را تغییر داد. برای انجام یک چنین تغییری کافیست تا نام متغیر مورد نظر را به‌همراه نوع داده دلخواهتان به‌عنوان آرگومان به این تابع ارسال کنید (لازم است تا این دو آرگومان را با استفاده از علامت کاما از یکدیگر جدا نمایید). لیست برنامه ۲- ۴ شامل کدی است که مقدار عدد اعشاری 3.14 را که از نوع داده double است، دریافت کرده و آن‌را به چهار نوع داده مختلف تبدیل می‌کند.

```
 1: <html>
 2: <head>
 3: <title>Listing 4.2 Changing the type of a variable with settype()</title>
 4: </head>
 5: <body>
 6: <?php
 7: $undecided = 3.14;
 8: print gettype( $undecided ); // double
 9: print " is $undecided<br>";  // 3.14
10: settype( $undecided, 'string' );
11: print gettype( $undecided ); // string
12: print " is $undecided<br>";  // 3.14
13: settype( $undecided, 'integer' );
14: print gettype( $undecided ); // integer
15: print " is $undecided<br>";  // 3
16: settype( $undecided, 'double' );
17: print gettype( $undecided ); // double
18: print " is $undecided<br>";  // 3.0
19: settype( $undecided, 'boolean' );
20: print gettype( $undecided ); // boolean
21: print " is $undecided<br>";  // 1
22: ?>
23: </body>
24: </html>
```

لیست ۲- ۴ تغییر نوع داده یک متغیر با استفاده از تابع Settype( )

دنباله کاراکتری " 3.14 " به یک عدد صحیح بخش اعشاری آن برای همیشه از دست رفته است. به‌همین دلیل نیز پس از تبدیل مجدد عدد صحیح حاصل به عدد اعشاری در خط ۱۶ هیچ اثری از بخش اعشاری (14.) متغیر $undecided مشاهده نمی‌شود. در نهایت در خط ۱۹ از برنامه نوع داده متغیر $undecided به نوع boolean تبدیل شده است. هنگام چاپ مقادیر boolean در زبان PHP مقدار true نماینده 1 و مقدار false نماینده هر دنباله کاراکتری خالی خواهد بود. از این‌رو عبارت خط ۲۱ برنامه منجر به چاپ 1 خواهد شد.

تغییر نوع داده‌ها با استفاده از روش Casting

روش Casting بسیار ساده است: با قرار دادن نام نوع داده مورد نظر در درون پرانتز و استفاده از آن در جلوی نام یک متغیر یک کپی از مقدار متغیر مزبور با نوع داده ذکر شده در درون پرانتز ایجاد می‌شود.

تفاوت اصلی مابین تابع ( )Settype و این روش در این حقیقت نهفته است که روش Casting یک کپی ایجاد کرده و نوع داده متغیر اصلی دست‌نخورده باقی می‌ماند. لیست برنامه ۳- ۴ بیانگر این حقیقت است.

```
 1: <html>
 2: <head>
 3: <title>Listing 4.3 Casting a variable</title>
 4: </head>
 5: <body>
 6: <?php
 7: $undecided = 3.14;
 8: $holder = ( double ) $undecided;
 9: print gettype( $holder ) ; // double
10: print " is $holder<br>";   // 3.14
11: $holder = ( string ) $undecided;
12: print gettype( $holder );  // string
13: print " is $holder<br>";   // 3.14
14: $holder = ( integer ) $undecided;
15: print gettype( $holder );  // integer
16: print " is $holder<br>";   // 3
17: $holder = ( double ) $undecided;
18: print gettype( $holder );  // double
19: print " is $holder<br>";   // 3.14
20: $holder = ( boolean ) $undecided;
21: print gettype( $holder );  // boolean
22: print " is $holder<br>";   // 1
23: print "<hr>";
24: print "original variable type: ";
25: print gettype( $undecided ); // double
26: ?>
27: </body>
28: </html>
```

لیست ۳- ۴ استفاده از روش Casting

همان‌گونه که مشاهده‌می‌کنید ما هرگز نوع داده متغیر $undecided را در این برنامه تغییر ندادیم، از این‌رو نوع این متغیر تا انتهای برنامه از نوع double باقی‌مانده است. این مطلب در خط ۲۵ از برنامه با استفاده از تابع ( )gettype جهت نمایش نوع داده متغیر مورد بحث نشان داده شده است.

در حقیقت ما با استفاده از روش Casting از مقدار ذخیره شده در متغیر $undecided ابتدا یک کپی تهیه کرده و سپس نوع داده آن کپی را به‌گونه دلخواه (با ذکر نام نوع داده مورد نظر در درون پرانتز) تغییر داده‌ایم. ابتدا در خط ۸ از برنامه و سپس در خطوط ۱۱، ۱۴، ۱۷ و ۲۰ حاصل این عملیات تغییر را در متغیری به‌نام $holder ذخیره کرده‌ایم. از آنجا که فرآیند تغییر نوع داده بر روی کپی متغیر $undecided انجام می‌شود، اطلاعات موجود در این متغیر به‌صورتی که در خطوط ۱۳ و ۱۹ از برنامه مشاهده می‌کنید، دست‌نخورده باقی می‌ماند.

اکنون که می‌توانیم نوع داده محتوای متغیرها را از نوعی به نوع دیگر تغییر دهیم (با استفاده از تابع ( )Settype یا روش Casting)، باید از فواید چنین تغییری نیز آگاه باشیم. در حقیقت چنین تغییراتی در برنامه‌های PHP اغلب روی نمی‌دهند چراکه PHP به‌صورت خودکار بسته به موقعیت، تغییرات مورد نیاز را بر روی نوع داده متغیرها اعمال می‌کند. با این حال باید همواره به‌خاطر داشته باشید که عملیات Casting خودکار که توسط PHP انجام می‌شود، نتایج موقتی داشته و در صورت لزوم برنامه‌نویس باید حاصل عملیات تغییر را به‌صورت دائمی در یک متغیر ذخیره کند.

اعداد تایپ شده توسط کاربران در فرمهای HTML همواره در اسکریپت‌های PHP به‌عنوان یک دنباله کاراکتری مورد دستیابی قرار می‌گیرند. از این‌رو اگر شما سعی کنید تا دو دنباله کاراکتری را که شامل اعدادی هستند با یکدیگر جمع کنید، PHP پیش از انجام هر عملیاتی برای محاسبه مجموع ابتدا آنها را به نوع داده عددی تبدیل خواهد کرد. بنابراین حاصل جمع زیر:

"30 cm" + " 40 cm "

به‌سادگی عدد صحیح 70 را تولید می‌کند. بدین ترتیب مشاهده می‌کنیم که در اعمال روش Casting به‌دنباله‌های کاراکتری، PHP کاراکترهای غیر عددی را نادیده می‌گیرد. با این حال ممکن است خود شما شخصاً علاقمند باشید تا کاراکترهای غیر عددی را هنگام انجام عملیات عددی بر روی ورودی‌هایی که از طریق فرمهای ورودی HTML دریافت می‌کنید، حذف نمائید. فرض کنید در قالب یک فرم HTML از کاربران خود خواسته‌اید تا یک مقدار عددی را وارد کنند. همین فرآیند را می‌توانیم با معرفی یک متغیر و انتساب مقداری به آن به صورت زیر شبیه‌سازی کنیم:

$test = "30 cm" ;

همان‌گونه که مشاهده می‌کنید کاربر مورد نظر ما به‌طور ناشیانه‌ای علاوه بر مقدار عددی خواسته شده، واحدی را نیز برای آن ذکر کرده است. به‌روش زیر می‌توان مطمئن شد که این ورودی عاری از هر کاراکتر غیر عددی بوده و تنها بخش عددی آن باقی می‌ماند. بدین ترتیب کاربرد دیگری از روش Casting را مشاهده می‌کنید:

```
$test = (integer) $test ;
Print " Your imaginary box has a width of $test centimeters " ;
```

در مورد انجام خودکار روش Casting توسط PHP در مواقع ضروری در درس ساعت شانزدهم با عنوان " بهره‌گیری از انواع داده‌ها " مطالب بیشتری را عنوان خواهیم کرد.

اهمیت تشخیص نوع داده‌ها

با توجه به مطالب عنوان شده درباره انواع داده‌ها این سؤال پیش می‌آید که اهمیت و فایده اطلاع از نوع داده یک متغیر چیست؟ هنگام برنامه‌نویسی در اغلب موارد با مواردی مواجه می‌شویم که مجبوریم تا داده‌هایی را از یک منبع دیگر در برنامه خود مورد استفاده قرار دهیم. برای مثال در درس ساعت ششم با چگونگی ایجاد توابع در برنامه‌هایتان آشنا می‌شوید. توابع قادرند تا اطلاعاتی را از برنامه‌ای که آنها را فراخوانده در قالب آرگومان‌هایی دریافت کنند. در مورد توابع همواره بهتر است ابتدا نوع داده‌ای که در قالب آرگومان به تابع ارسال شده است را با نوع داده‌ای که تابع انتظار دریافت آن را دارد، مقایسه کرده و از یکی بودن آنها اطمینان حاصل شود. برای مثال تابعی که انتظار دریافت آرگومانی از نوع resource را دارد، با دنباله‌ای که به‌عنوان آرگومان به آن ارسال شده است، کار نخواهد کرد.

عملگرها و عبارات PHP

تا به اینجا با چگونگی مقداردهی متغیرها آشنا شدید. همچنین نحوه تغییر نوع داده متغیرها را در قالب برنامه‌های PHP کوتاهی تجربه کردید. اما به یاد داشته باشید که چنانچه نتوانید عملیات مورد نظرتان را بر روی داده‌ها انجام دهید، استفاده از زبانهای برنامه‌نویسی و نوشتن برنامه‌ها هیچ فایده‌ای ندارد. عملگرها علایمی هستند که استفاده از یک یا چند متغیر را جهت تولید یک مقدار جدید ممکن می‌سازند. معمولاً به مقداری که به واسطه یک عملگر، عملیاتی بر روی آن صورت می‌گیرد، عملوند گفته می‌شود.

واژه جدید عملگر یا Operator علامت یا یکسری از علایمی است که تاثیر آن بر روی یک یا چند مقدار ذخیره شده در متغیرها معمولاً موجب تولید یک مقدار جدید می‌گردد.

واژه جدید عملوند مقداری است که عملگر بر روی آن اعمال می‌شود. معمولاً در بیشتر مواقع شاهد آن هستیم که یک عملگر به‌همراه دو عملوند مورد استفاده قرار می‌گیرد.

اجازه دهید تا در این‌جا مثالی که نشانگر ترکیب دو عملوند و یک عملگر برای رسیدن به یک عملوند جدید است را بررسی کنیم ؛ در نمونه زیر:

```
4 + 5
```

اعداد 4 و 5 عملوند هستند. عملگر محاسبه مجموع (+) بر روی این دو عملگر تأثیر گذاشته و موجب تولید مقدار عددی 9 می‌شود. جایگاه عملگرها تقریباً همیشه مابین دو عملوند است. البته به‌زودی با عملگرهایی آشنا می‌شوید که از این قاعده عمومی مستثنی هستند. این عملگرها را در درس همین ساعت بررسی می‌کنیم.

ترکیب عملوندها با یک عملگر جهت تولید یک نتیجه جدید معمولاً یک عبارت نامیده می‌شود. هرچند که بیشتر عملگرها پایه و اساس عبارات را تشکیل می‌دهند اما لزوماً برای داشتن یک عبارت به‌وجود عملگر نیاز نمی‌باشد. در حقیقت در زبان PHP عبارت هر چیزی است که بتوان از آن به جای یک مقدار استفاده کرد. با این تعریف مقادیر عددی ثابت مانند 654، متغیرها مثل $user و فراخوانی توابع مانند ( )gettype نوعی عبارت محسوب می‌شوند. از این‌رو می‌توان چنین استنباط کرد که عبارت (5 + 4) از دو عبارت کوچک‌تر 4 و 5 و عملگر محاسبه مجموع (+) تشکیل شده است. هنگامی که عبارتی منجر به تولید یک مقدار می‌شود، چنین بیان می‌شود که آن عبارت معادل با آن مقدار می‌باشد. بدین ترتیب هنگامی که تمامی عبارات جزئی یک عبارت بزرگ‌تر را به حساب آوریم، می‌توانیم آن عبارت را به منزله کدی فرض کنیم که می‌توان از آن به‌جای مقدار واقعی عبارت استفاده کرد.

واژه جدید عبارت یا expression به هر ترکیبی از توابع، مقادیر و عملگرها گفته می‌شود که بتوان معادل یک مقدار مشخص فرض کرد. به‌عنوان یک قاعده کلی، عبارت به هر چیزی گفته می‌شود که بتوان از آن به‌جای یک مقدار مشخص در برنامه استفاده کرد.

اکنون که با اصول و قواعد بازی عملگرها و عبارات آشنا شدیم وقت آن‌است که نگاه دقیق‌تر و جزئی‌تری به عملگرهای متداول در زبان برنامه‌نویسی PHP بیندازیم.

عملگرهای نسبت دهی

تاکنون از عملگرهای نسبت‌دهی هنگام مقداردهی متغیرها در برنامه استفاده کرده‌اید. این عملگر به‌سادگی همان عملگر = است که تا بدین جا در چندین برنامه آن‌را مورد استفاده قرار داده‌اید. عملگر نسبت‌دهی همواره مقدار موجود در سمت راست خود را به آن‌چه که در سمت چپ وجود دارد، نسبت می‌دهد. از آن‌جا که از واژه عملگر نسبت‌دهی استفاده کرده‌ایم، مناسب است که عوامل سمت چپ و راست آن‌را عملوند بنامیم. به نمونه زیر توجه کنید:

```
$name = " matt " ;
```

با اجرای خط فوق در یک برنامه PHP متغیر $name شامل مقدار " matt " که یک دنباله کاراکتری است، خواهد بود. جالب است بدانید که ساختار مذکور یک عبارت است. در نگاه اول ممکن است چنین به نظر برسد که عملگر نسبت دهی به‌سادگی مقدار متغیر $name را بدون تولید یک مقدار

جدید تغییر می‌دهد، اما به واقع عبارتی که از عملگر نسبت دهی استفاده می‌کند همیشه معادل با کپی مقداری است که عملوند سمت راست تساوی آن‌را نگه می‌دارد. از این جهت عبارت زیر:

```
Print ( $name = " matt ") ;
```

علاوه بر اینکه دنباله کاراکتری " matt " را به متغیر $name نسبت می‌دهد، این دنباله کاراکتری را نیز بر روی پنجره مرورگر اینترنت چاپ می‌کند.

عملگرهای ریاضی

عملگرهای ریاضی دقیقاً همان چیزی را انجام می‌دهند که شما انتظارش را دارید. جدول ۳- ۴ این دسته از عملگرها را با جزئیات و عملکرد نمونه نشان می‌دهد. عملگر جمع دو عملوند راست و چپ خود را با یکدیگر جمع می‌کند. عملگر تفریق عملوند سمت راست خود را از عملوند سمت چپ کم می‌کند. عملگر تقسیم عملوند سمت چپ خود را به عملوند سمت راست تقسیم می‌کند. عملگر ضرب حاصل‌ضرب دو عملوند سمت چپ و راست خود را با ضرب کردن آنها در یکدیگر محاسبه می‌کند.

و بالاخره عملگر باقیمانده تقسیم، باقیمانده حاصل از تقسیم عملوند سمت چپ خود را به عملوند سمت راست محاسبه می‌کند.

جدول ۳- ۴ عملگرهای ریاضی

نام عملگر	نشانه مربوطه در PHP	مثال کاربردی	نتیجه عملیات
جمع	+	10 + 3	13
تفریق	-	10 - 3	7
تقسیم	/	10 / 3	3.3333333
ضرب	*	10 * 3	30
باقی‌مانده	%	10 % 3	1

عملگر ترکیب

عملگر ترکیب به‌سادگی یک علامت نقطه می‌باشد. با فرض این‌که هر دو عملوند این عملگر از نوع داده دنباله کاراکتری باشند، این عملگر عملوند سمت راست خود را به عملوند سمت چپ خود متصل می‌کند. بنابراین عملیات زیر:

```
" hello " . " world "
```

حاصل زیر را به‌دنبال خواهد داشت:

```
" hello world "
```

بدون توجه به نوع داده عملوندهای این عملگر، عملیات به‌گونه‌ای است که عملوندها و همچنین حاصل عملیات از نوع دنباله کاراکتری فرض می‌شوند. در سراسر این کتاب هنگامی که نیازمند ترکیب نتایج حاصل از یک عبارت با دنباله‌ای از کاراکترها باشیم به‌طور مکرر از عملگر ترکیب استفاده خواهیم کرد. به‌نمونه زیر که ترکیب دو دنباله کاراکتری با حاصل یک عملیات تقسیم است، توجه کنید:

```
$centimeters = 212 ;
print " the width is ". ( $centimeters / 100). " meters " ;
```

عملگرهای نسبت دهی مرکب

با وجودی که در حقیقت تنها یک عملگر نسبت‌دهی وجود دارد، زبان برنامه‌نویسی PHP4 تعدادی عملگر ترکیبی معرفی کرده است که علاوه بر تغییر عملوند سمت چپ نتیجه را نیز باز می‌گرداند. به‌عنوان یک قاعده کلی، عملگرها عملوندهای خود را بدون تغییر مقادیر آنها مورد استفاده قرار می‌دهند. اما عملگرهای نسبت دهی این قاعده را زیر پا می‌گذارند. یک عملگر نسبت‌دهی مرکب ترکیبی است از یک عملگر استاندارد که به‌دنبال آن از علامت نسبت دهی (=) استفاده شده باشد. عملگرهای نسبت دهی مرکب به‌واسطه کاهش دو عملیات به یک عملیات در وقت شما صرفه‌جویی کرده و ضمناً با این کار احتمال خطا و اشتباه را کاهش می‌دهند.

برای مثال دو عبارت زیر را در نظر بگیرید:

```
$X = 4 ;
$X = $ X + 4 ; // $X now equals 8 .
```

به صورت زیر می‌توان عملیات عبارت دوم را با استفاده از عملگر نسبت دهی تقلیل داد:

```
$X = 4 ;
$X + = 4 ; // $X now equals 8.
```

به ازای هر یک از عملگرهای ریاضی که پیشتر در جدول ۳- ۴ مشاهده کردید و همچنین به ازای عملگر ترکیب یک عملگر نسبت دهی مرکب وجود دارد. جدول ۴- ۴ برخی از متداول‌ترین آنها را نشان می‌دهد.

همان‌گونه که مشاهده می‌کنید، در هر یک از مثالهای این جدول مقدار متغیر $X با استفاده از مقدار موجود در سمت راست عملگر نسبت دهی مرکب دستخوش تغییر شده است.

جدول ۴- ۴ برخی از عملگرهای نسبت دهی مرکب

عملگر نسبت دهی مرکب	مثال	عملیات معادل
+=	$X += 5	$X = $X + 5
-=	$X -= 5	$X = $X - 5
/=	$X /= 5	$X = $X / 5
*=	$X *= 5	$X = $X * 5
%=	$X %= 5	$X = $X % 5
.=	$X .= " five "	$X = $X . " five "

عملگرهای مقایسه‌ای

عملگرهای مقایسه‌ای عمل بررسی و ارزیابی عملوندها را انجام می‌دهند. چنان‌چه نتیجه این ارزیابی موفقیت آمیز باشد، این عملگرها مقدار true و در غیر این‌صورت مقدار false را باز می‌گردانند. از این نوع فرآیندهای مقایسه و ارزیابی در ساختارهای کنترلی برنامه، همچون ساختارهای if و while استفاده‌های زیادی می‌شوند. در درس ساعت پنجم در مورد این ساختارهای کنترلی مفصل بحث خواهیم کرد.

برای نمونه، به‌منظور تشخیص این موضوع که آیا مقدار ذخیره شده در متغیر $X کوچک‌تر از عدد 5 است می‌توان از عملگر " کوچک‌تر است از " با علامت < استفاده کرد:

```
$X < 5
```

چنانچه متغیر $X شامل عدد 3 باشد. نتیجه این ارزیابی مقدار true و اگر این متغیر شامل عدد 7 باشد ارزیابی مذکور معادل با مقدار false خواهد بود.

اسامی، علایم و جزئیات مربوط به عملگرهای مقایسه‌ای در جدول ۵- ۴ آمده است.

جدول ۵- ۴ عملگرهای مقایسه‌ای در زبان PHP

نام عملگر	علامت	شرط ارزیابی به‌صورت true	مثال(بافرض این‌که $X برابر با 4 باشد)	نتیجه مثال
تساوی	= =	عملوندهای چپ و راست برابر	$X = = 5	false
عدم تساوی	! =	عملوندهای چپ و راست‌نابرابر	$X ! = 5	true
معادل	= = =	عملوندهای چپ و راست برابر و هم‌نوع	$X = = = 5	false
بزرگ‌تر از	>	عملوند چپ بزرگ‌تر از راست	$X > 4	false
کوچک‌تر از	<	عملوند چپ کوچک‌تر از راست	$X < 4	false
بزرگ‌تریامساوی	> =	عملوند چپ بزرگ‌تر یا مساوی راست	$X > = 4	true
کوچک‌تر یا مساوی	< =	عملوند چپ کوچک‌تر یامساوی راست	$X < = 4	true

از عملگرهای ذکر شده در این جدول در بیشتر مواقع برای مقایسه مقادیر عددی صحیح یا اعشاری استفاده می‌شود. با این همه، عملگر تساوی برای مقایسه دنباله‌های کاراکتری نیز مورد استفاده زیادی دارد.

ایجاد عبارتهای مقایسه‌ای پیچیده با استفاده از عملگرهای منطقی

عملگرهای منطقی فرآیند مقایسه مابین مقادیر boolean را انجام می‌دهند. برای مثال عملگر منطقی or که با دو کاراکتر pipe متوالی به‌صورت || و یا به‌سادگی با واژه or مشخص می‌شود تنها در صورتی به شکل true ارزیابی می‌شود که دست کم یکی از عملوندهای سمت چپ و راست آن true ارزیابی شوند. بدین ترتیب عبارت مقایسه‌ای زیر:

true || False

برابر با مقدار true ارزیابی می‌گردد.

همچنین عملگر منطقی and که با دو کاراکتر متوالی ampersand به‌صورت && و یا به‌سادگی با واژه and مشخص می‌شود، تنها زمانی به‌صورت true ارزیابی خواهد شد که هر دو عملوند سمت چپ

و راست آن به صورت true ارزیابی شوند. از این‌رو عبارت مقایسه‌ای true && false به‌صورت false ارزیابی خواهد شد.

علی‌رغم دو مثال فوق بعید است که کسی برای مقایسه ثابتهای نوع داده boolean ، یعنی true و false از عملگرهای منطقی استفاده کند. معمولاً از این نوع عملگرها برای ارزیابی دو یا چند عبارتی که معادل این مقادیر ثابت هستند، استفاده می‌شود. برای مثال عبارت زیر:

($X > 2) && ($X < 15)

در صورتی که متغیر $X مقداری بزرگ‌تر از 2 و کوچک‌تر از 15 داشته باشد، معادل با true ارزیابی خواهد شد. وجود پرانتزها به این علت است که وضوح عبارت مذکور بیشتر شده و خوانایی آن افزایش یابد. جدول ۶- ۴ اسامی عملگرهای منطقی را به‌همراه جزئیات مربوطه نشان می‌دهد.

جدول ۶- ۴ عملگرهای منطقی

نام عملگر	علامت	شرایط ارزیابی به‌صورت true	مثال	نتیجه مثال
Or	||	عملوند چپ یا راست true	true || false	true
Or	or	عملوند چپ یا راست true	true or false	true
Xor	xor	تنها یکی از عملوندهای چپ یا راست true	true xor true	false
And	&&	عملوندهای چپ و راست true	true && false	false
And	and	عملوندهای چپ و راست true	true and false	false
Not	!	عملوند غیر true	! true	false

ممکن است با مشاهده جدول فوق این سؤال به ذهن برسد که چرا از عملگرهای منطقی and و or دو نسخه متفاوت موجود است؟ پاسخ در حقیقتی با عنوان تقدم عملگرها نهفته است. در این‌باره بعداً به‌طور مفصل در درس همین ساعت به بحث و بررسی خواهیم پرداخت.

افزایش و کاهش خودکار مقدار یک متغیر از نوع عددی صحیح

هنگام برنامه‌نویسی با PHP اغلب به مواقعی برمی‌خوریم که نیاز است تا مقدار یک متغیر صحیح را افزایش یا کاهش دهیم. معمولاً هنگامی نیاز به این کار داریم که قصد شمارش تعداد دفعات گردش در طول یک حلقه را در برنامه داشته باشیم. در حال حاضر با دو روش انجام این کار آشنا

هستید. در روش اول می‌توانیم مقدار عدد صحیح ذخیره شده در متغیر $X را به کمک عملگر ریاضی جمع به صورت زیر افزایش دهیم:

```
$X = $X + 1 ; // $X is incremented
```

همچنین می‌توانیم این کار را با استفاده از عملگر نسبت دهی مرکب = + به شکل زیر نیز انجام دهیم:

```
$X + = 1 ; // $X is incremented
```

در هر دو صورت عدد صحیح حاصل که اکنون به اندازه یک واحد نسبت به قبل افزایش یافته، در متغیر $X ذخیره شده است. به‌دلیل اینکه عبارات این‌چنین در برنامه‌نویسی متداول هستند، در زبان PHP عملگرهای ویژه‌ای پیش بینی شده است که امکان افزایش یا کاهش مقدار متغیرهای عددی صحیح به اندازه یک واحد و سپس نسبت دهی نتیجه حاصل به همان متغیر را به‌سادگی در اختیار برنامه‌نویس قرار می‌دهند. این عملگرها با نامهای پس‌افزایش (post -increment) و پس‌کاهش (post - decrement) در بین برنامه‌نویسان متداول هستند. عملگر پس‌افزایش با دو علامت جمع متوالی که به‌دنبال نام متغیر قرار می‌گیرد، مشخص می‌شود. به نمونه زیر توجه کنید:

```
$X+ + ; // $ X is incremented
```

در این نمونه مقدار متغیر $X به اندازه یک واحد افزایش می‌یابد. با قراردادن دو علامت منهای متوالی پس از نام متغیر عملگر پس‌کاهش حاصل می‌آید. به‌نمونه زیر توجه کنید:

```
$X - - ; // $X is decremented
```

چنان‌چه عملگر پس‌افزایش یا پس‌کاهش را به‌همراه یک عملگر مقایسه‌ای مورد استفاده قرار دهید، این نکته را همواره به‌خاطر داشته باشید: مقدار عملوند مربوطه تنها پس از انجام عمل مقایسه تغییر خواهد کرد. به مثال زیر توجه کنید:

```
$X = 3 ;
$X + + < 4 ; // true
```

در این مثال متغیر $X هنگامی که با مقدار ثابت عددی 4 مقایسه می‌شود، کماکان شامل عدد 3 است. از این‌رو عبارت آخر مقدار true را باز می‌گرداند. پس از انجام مقایسه و حصول نتیجه فوق مقدار موجود در متغیر $X به‌واسطه عملگر پس‌افزایش به اندازه یک واحد افزایش می‌یابد.

در برخی شرایط ممکن است عکس این عملکرد مورد نیاز باشد، یعنی بخواهیم پیش از انجام عمل مقایسه متغیر مقدار آن‌را افزایش یا کاهش دهیم. برای انجام این منظور نیز PHP عملگرهایی را پیش‌بینی کرده‌است که با عناوین پیش‌افزایش (pre- increment) و پیش‌کاهش (pre- decrement) شناخته می‌شوند. تاثیر این دو عملگر تشابه زیادی با دو عملگر قبل یعنی عملگرهای پس افزایش و پس کاهش دارد. ظاهر آنها نیز شبیه به هم است، بدین ترتیب که عملگر پیش‌افزایش متشکل از دو علامت جمع متوالی است که پیش از نام متغیر قرار می‌گیرد. به‌طور مشابه، عملگر پیش‌کاهش نیز متشکل از دو علامت منهای متوالی است. که پیش از نام متغیر قرار می‌گیرد.

به دو نمونه زیر توجه کنید:

```
+ + $X ; // $X is incremented
- - $X ; // $X is decremented
```

اگر این عملگرها به‌عنوان بخشی از یک عبارت مورد استفاده قرار بگیرند، فرآیند افزایش یا کاهش مقدار متغیر مربوطه پیش از انجام هرگونه عملیاتی بر روی این متغیر صورت خواهد گرفت. به مثال زیر توجه کنید:

```
$X = 3 ;
+ + $X < 4 ; // false
```

در مثال فوق مقدار متغیر $X درست پیش از انجام مقایسه آن با عدد 4 به اندازه یک واحد افزایش می‌یابد و از آن‌جا که عدد 4 در شرایط کوچک‌تر یا مساوی با خودش صدق نمی‌کند، عبارت شرطی فوق مقدار false را باز می‌گرداند.

تقدم عملگرها

موتور PHP هنگام مواجهه با یک عبارت عملگرهای به‌کار رفته در آن‌را معمولاً به‌ترتیب از سمت چپ به راست مورد ارزیابی قرار می‌دهد. در مورد عباراتی که شامل بیش از یک عملگر هستند البته جزئیاتی وجود دارد که لازم است تا برنامه‌نویس به‌دقت آنها را در نظر داشته باشد. برای نمونه عبارت بسیار ساده زیر را در نظر بگیرید:

```
4 + 5
```

مفهوم این عبارت کاملاً روشن بوده و امکان اشتباه وجود ندارد، چرا که در این مورد PHP به‌سادگی دو عدد 4 و 5 را با یکدیگر جمع می‌کند. اما در مورد عبارت زیر نیز آیا همان میزان وضوح وجود دارد؟

```
4 + 5 * 2
```

عبارت فوق پای مشکلی را به پیش می‌کشد. آیا این عبارت بدان معنی است که ابتدا مجموع دو عدد 4 و 5 محاسبه شده و حاصل جمع در عدد 2 ضرب شود؟ بدین ترتیب عدد 18 به‌دست می‌آید. یا اینکه هدف این عبارت حاصل‌ضرب دو عدد 5 و 2 در یکدیگر و جمع نتیجه به دست آمده با عدد 4 است؟ در این حالت عدد 14 حاصل می‌شود. مشاهده می‌کنید که نتایج نهایی کاملاً متفاوتند. اگر به‌جای موتور PHP اقدام به محاسبه چنین عبارتی می‌کردید و جملات آن‌را از چپ به راست می‌خواندید، روش اول را درست تصور می‌کردید. با این حال آگاه باشید که PHP قوانین اولویت‌بندی خاصی برای اعمال عملگرها به عملوندها دارد. از آنجا که اولویت عملگر ضرب نسبت به عملگر جمع بالاتر است در مورد عبارت فوق پاسخ روش دوم یعنی 14 صحیح است. چرا که ابتدا عمل ضرب 5 و 2 در یکدیگر و سپس عمل جمع حاصل با عدد 4 صورت می‌گیرد.

با این همه در صورت تمایل می‌توانیم با استفاده به‌جا از پرانتزها PHP را مجبور کنیم تا این قوانین سفت و سخت اولویت‌بندی را نادیده بگیرد. به عبارت زیر توجه کنید:

```
(4 + 5) * 2
```

در مورد عبارت فوق وجود پرانتزها باعث می‌شود تا ابتدا حاصل جمع دو عدد 4 و 5 محاسبه شده و در نهایت در عدد 2 ضرب شود. بدین ترتیب پاسخ 18 به دست می‌آید.

بدون توجه به قوانین اولویت‌بندی عملگرها در یک عبارت پیچیده، بهره‌گیری از پرانتزها همواره باعث می‌شود که برنامه از وضوح و خوانایی بیشتری برخوردار شده و احتمال وقوع اشتباه بدین ترتیب کاهش یابد. جدول ۷-۴، لیست عملگرهای بررسی شده در درس این ساعت را بر حسب ترتیب نزولی اولویتها از بالا به پایین نشان می‌دهد (اقلام بالاتر در جدول از اولویت بیشتری برخوردارند).

جدول ۷- ۴ ترتیب اولویت عملگرهای منتخب در PHP

عملگر(ها)
+ + - - (cast)
/ * %
+ -
< < = > = >
== === !=
&&
||
= + = - = /= *= %= .=
and
xor
Or

همان‌گونه که مشاهده می‌کنید، عملگر or نسبت به عملگر || و همچنین عملگرand نسبت به عملگر && از اولویت پائین‌تری برخوردار هستند. از این‌رو می‌توان از عملگرهای منطقی با اولویت پایین‌تر جهت تغییر در چگونگی عملکرد عبارتهای منطقی پیچیده استفاده نمود. این روش البته لزوماً ایده مناسبی نمی‌باشد. به دو عبارت زیر توجه کنید. هر دو عبارت ارزش یکسانی داشته و به یک صورت ارزیابی می‌شوند. با این حال عبارت دوم از خوانایی بالاتری برخوردار است:

```
$X and $y || $z
$x && ( $y || $z)
```

وجود قانون اولویت‌بندی تنها دلیل معرفی دو عملگر && و and جهت یک عمل واحد در زبان PHP است. همین گفته در مورد عملگرهای || و or صدق می‌کند. با این همه، در بیشتر موارد بهره‌گیری از پرانتزها می‌تواند به افزایش وضوح و خوانایی برنامه‌ها کمک کرده و از میزان اشکلات آن بکاهد. این گفته در مورد عبارت پیچیده‌تری که از عملگرهای بیشتری بهره می‌برند، بیشتر صدق می‌کند. در سرتاسر این کتاب ما استفاده از عملگرهای && و || را بر استفاده از and و or ترجیح داده‌ایم.

مقادیر ثابت

متغیرها روش قابل انعطاف و مؤثری برای ذخیره داده‌ها محسوب می‌شوند چراکه به‌هنگام نیاز می‌توان مقادیر و نوع داده ذخیره شده در آنها را تغییر داد. با این همه، اگر مایل باشیم تا با مقداری در برنامه کارکنیم که بنا به دلایلی آن مقدار در طول اجرای برنامه تغییر نکند، می‌توانیم به‌جای استفاده از متغیرها از ثابتها در برنامه استفاده کنیم. برای بهره‌گیری از مقادیر ثابت ابتدا باید آنها را تعریف کنیم. برای تعریف مقادیر ثابت در برنامه‌های PHP لازم است تا از تابع سیستمی ویژه‌ای با نام define( ) استفاده کنیم. پس از تعریف مقدار ثابت مورد نظر توجه داشته باشید که نمی‌توان آن‌را در سرتاسربرنامه تغییر داد. استفاده از تابع define( ) بسیار ساده است. کافی‌است تا نام ثابت مورد نظر و مقداری را که می‌خواهیم به آن نسبت دهیم در قالب دو آرگومان به این تابع ارسال کنیم. در زبان PHP معمولاً آرگومان‌های تابع را در درون پرانتز و در جلوی نام آن می‌نویسیم و آنها را با علامت کاما از یکدیگر جدا می‌کنیم. به نمونه زیر در این رابطه توجه کنید:

```
Define ( " CONSTANT_ NAME ", 42) ;
```

دقت کنید که تنها مقادیر عددی و دنباله‌های کاراکتری را می‌توانیم به ثابتها منسوب کنیم. چنین متداول است که برنامه‌نویسان از حروف انگلیسی بزرگ برای نام‌گذاری اسامی ثابتها استفاده می‌کنند. مقادیر ثابت تنها از طریق اسامی آنها قابل دستیابی بوده و استفاده از علامت دلار ($) ضرورتی ندارد. لیست برنامه ۴- ۴ چگونگی تعریف و دستیابی به یک ثابت را نشان می‌دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 4.4 Defining a constant</title>
 4: </head>
 5: <body>
 6: <?php
 7: define( "USER", "Gerald" );
 8: print "Welcome ".USER;
 9: ?>
10: </body>
11: </html>
```

لیست ۴- ۴ تعریف و دستیابی به مقادیر ثابت

توجه کنید که در خط ۸ از این برنامه کوتاه PHP از عملگر ترکیب جهت ضمیمه کردن محتویات ثابت USER به انتهای دنباله کاراکتری " Welcome " استفاده کرده‌ایم. این بدان دلیل است که موتور PHP هیچ راهی برای تفاوت گذاشتن بین یک ثابت و دنباله کاراکتری موجود مابین جفت علامت " " را ندارد.

تابع ( )define قادر است تا آرگومان سومی را نیز بپذیرد. استفاده از این آرگومان برای برنامه‌نویس اختیاری است، بدین معنی که برنامه‌نویس تنها با مشخص کردن دو آرگومان اول قادر است تا ثابت مورد نظر خود را تعریف کند. به‌کمک آرگومان سوم که باید مقداری از نوع boolean باشد، برنامه‌نویس می‌تواند لزوم این مطلب را که موتور PHP باید مابین حروف بزرگ و کوچک نام ثابت تفاوت قائل شود یا خیر مشخص نماید. طبق پیش‌فرض چنین تفاوتی در مورد اسامی ثابتها وجود دارد، اما برنامه‌نویس با ارسال مقدار true به‌عنوان سومین آرگومان به تابع ( )define بر لزوم حذف چنین تفاوتی تاکید نماید. بنابراین اگر از عبارت زیر برای تعریف ثابتی با نام USER در برنامه استفاده کنیم:

```
define( " USER " ", Gerald " , true ) ;
```

آن‌گاه می‌توانیم بدون نگرانی از هریک از سه عبارت زیر برای نمایش مقدار این ثابت استفاده نماییم:

```
Print User ;
Print usEr ;
Print USER ;
```

چرا که هر سه عبارت فوق از دیدگاه موتور PHP یکسان تلقی می‌شوند. این ویژگی می‌تواند خوانایی برنامه شما توسط سایر برنامه‌نویسان را اندکی افزایش دهد چراکه در این صورت نیازی نیست تا هنگام دستیابی به مقدار این‌گونه ثابتها ملاحظات ویژه‌ای درباره بزرگی و کوچکی حروف درنظر گرفته شود. اما از طرف دیگر، این حقیقت که آیا برنامه نسبت به کوچک یا بزرگ بودن اسامی ثابتها حساس است یا خیر به جای کاهش اشتباه می‌تواند موجب افزایش اشتباه برنامه‌نویسان در این زمینه شود. به‌عنوان یک قاعده کلی، چنان‌چه یک دلیل محکم و قابل استناد ندارید، بهتر است ترتیبی دهید که برنامه‌تان نسبت به حروف بزرگ و کوچک اسامی ثابتها حساس باشد. علاوه بر این بهتر است ثابتها

را تمامی با حروف بزرگ انگلیسی نام‌گذاری کرده و در صورت استفاده از بیش از یک کلمه برای نام‌گذاری از علامت underscore مابین آنها استفاده نمایید. (مانند MY _ CONSTANT).

ثابتهای سیستمی

PHPبه‌خودی‌خودشامل چندین ثابت سیستمی است. برای مثال، ثابت سیستمی _ _ FILE _ _ همواره شامل نام فایلی است که موتور PHP هم‌اکنون در حال پردازش آن است. همچنین ثابت سیستمی _ _ LINE _ _ شامل شماره خطی از برنامه PHP مربوطه است که این ثابت در آن‌جا مورد استفاده قرار گرفته است. از این گونه ثابتها معمولاً جهت تولید پیغامهای خطا استفاده می‌شود. همچنین ثابت سیستمی PHP_ VERSION نسخه‌ای از PHP را مشخص می‌کند که درحال پردازش برنامه مورد نظر است. از این ثابت سیستمی در مواقعی استفاده می‌شود که بخواهیم اجرای برنامه‌های PHP را محدود به نسخه ویژه‌ای از این زبان برنامه‌نویسی نماییم.

جمع‌بندی

در درس این ساعت بعضی از ویژگی‌های ابتدایی زبان برنامه‌نویسی PHP را با هم مورد بررسی قرار دادیم. در این درس با چگونگی معرفی متغیرها و نحوه مقداردهی آنها را با استفاده از عملگر نسبت دهی (=) فراگرفتید. همچنین با برخی از عملگرها و نیز چگونگی ترکیب آنها با یکدیگر و ایجاد عبارات پیچیده‌تر آشنا شدید. در انتهای درس نیز چگونگی تعریف مقادیر ثابت با استفاده از تابع ( )define و نحوه دستیابی به آنها را فراگرفتید.

اکنون که مهارت و دانش خود را در مورد برخی از اصول و مبانی زبان برنامه‌نویسی PHP افزایش دادید، در درس ساعت آینده می‌توانیم بحث خود در مورد برنامه‌های PHP را جدی‌تر بررسی کنیم. در درس ساعت آینده با چگونگی برنامه‌نویسی جهت تصمیم‌گیری و انجام امور تکراری آشنا می‌شوید. مطالب درس حاضر، یعنی مطالب مربوط به متغیرها، عبارات و عملگرها کمک زیادی برای درس آینده شما خواهند بود.

پرسش و پاسخ

پرسش: اطلاع از نوع داده ذخیره شده در یک متغیر چه فایده‌ای دارد؟

پاسخ: در اغلب مواقع نوع داده یک متغیر نشان‌دهنده کاری است که می‌توانید با آن متغیر انجام دهید. برای مثال ممکن است بخواهید پیش از انجام هرگونه عملیات ریاضی و محاسباتی بر روی متغیر مورد نظرتان از اینکه این متغیر شامل داده‌ای از نوع عددی (عدد صحیح یا اعشاری) است،

اطمینان حاصل کنید.

در درس روز شانزدهم مواردی از این قبیل را مشاهده خواهید کرد.

پرسش: آیا به هر ترتیب دلخواهی می‌توان متغیرها را نام‌گذاری کرد؟

پاسخ: همیشه هدف اصلی شما هنگام نام‌گذاری متغیرها باید تسهیل و ساده‌سازی امر خوانایی برنامه و درک آن باشد. برای نمونه، متغیری با نام abc 123245 $ در مورد نقش و جایگاه این متغیر در برنامه هیچ‌گونه اطلاعاتی در اختیار خواننده آن قرار نداده و همین امر می‌تواند منشأ بروز اشتباه باشد. همواره نام متغیرهای خود را کوتاه اما توصیف‌کننده انتخاب کنید، به‌گونه‌ای که در عین ایجاز بیانگر وظیفه‌ای باشد که در برنامه برای آن درنظر گرفته‌اید.

متغیری با نام $f به احتمال قوی حتی برنامه‌نویس خود را پس از گذشت یک یا چند ماه در مورد وظیفه‌اش به اشتباه خواهد انداخت. در مقابل متغیری با نام $filename کاملاً توصیفی است.

پرسش: آیا به‌خاطر سپردن تقدم عملگرها الزامی است؟

پاسخ: هیچ‌دلیلی برای عدم انجام این کار در دست نیست، اما تجربه نشان داده که با این کار نتایج مفیدتری حاصل می‌شود. با به‌کارگیری‌پرانتزها در عبارات PHP می‌توانید بر میزان خوانایی برنامه خود اضافه کنید و ضمناً تقدمی را که مد نظرتان است بر روی عملگرهای مورد استفاده اعمال نمایید.

تمرینها

هدف از این بخش دوره مطالب فراگیری شده در قالب آزمون است. پاسخ بخش آزمون بلافاصله بعد از آن آمده است. بخش فعالیتها شامل تمرینهایی است که به‌منظور افزایش قابلیت برنامه نویسی خواننده طراحی شده و البته فاقد پاسخ است.

آزمون

۱- کدام‌یک از موارد زیر برای نام‌گذاری متغیرها معتبر هستند؟

```
$a - value _ submitted _ by _ a _ user
$ 666666 xyz
$ xyz 666666
$ _ _ _ _ counter _ _ _ _
$ the first
$ file - name
```

۲- خروجی حاصل از اجرای قطعه کد زیر چیست؟

```
$num = 33 ;
(boolean) $num ;
print $num ;
```

۳- خروجی حاصل از اجرای عبارت زیر چیست؟

```
Print gettype (" 4 ") ;
```

۴- خروجی حاصل از اجرای قطعه کد زیر چیست؟

```
$ test _ val = 5.4566 ;
settype ( $test _ val, " integer ") ;
print $test _ val ;
```

۵- کدام‌یک از موارد زیر شامل عبارت نمی‌باشند؟

```
4 ;
gettype (44) ;
5/12 ;
```

۶- کدام‌یک از موارد ذکر شده در پرسش قبل شامل یک عملگر هستند؟

۷- عبارت زیر چه‌مقداری را برمی‌گرداند؟

```
5 < 2
```

نوع داده مقدار بازگشتی را ذکر نمایید.

پاسخ آزمون

۱- استفاده از نام $ 666666 xyz برای یک متغیر غیر مجاز است چرا که اسامی متغیرها باید با یک حرف یا علامت (کاراکتر) ویژه‌ای با عنوان underscore آغاز شود. همچنین نام $the first نیز برای این منظور غیر مجاز است چرا که شامل فضای خالی است. نام $ file - name نیز چنین است زیرا عموماً استفاده از کاراکترهای غیر الفبایی و غیر عددی مانند علامت خط فاصله برای نام‌گذاری متغیرها معتبر نیست.

۲- خروجی حاصل از اجرای این قطعه کد عدد صحیح 33 است. در این‌جا فرآیند casting جهت تبدیل نوع داده متغیر $num به boolean موجب تبدیل کپی این مقدار می‌شود نه خود آن. بدین ترتیب نوع داده متغیر $num دست‌نخورده باقی می‌ماند.

۳- خروجی این عبارت رشته کاراکتری " String " است.

۴- خروجی حاصل از اجرای این قطعه کد عدد 5 است. هنگام تبدیل یک عدد اعشاری به یک عدد صحیح اطلاعات بخش اعشاری عدد از دست می‌رود.

۵- از آنجا که همه موارد معادل یک مقدار می‌باشند، لذا همگی عبارت محسوب می‌شوند.

۶- عبارت 5/12 شامل عملگر تقسیم است.

۷- این عبارت معادل false و نوع داده آن boolean است.

فعالیتها

۱- برنامه‌ای بنویسید که دست‌کم شامل پنج نوع متغیر مختلف باشد. به هر یک از این متغیرها مقداری با نوع داده مختلف منسوب نمایید. در نهایت با به‌کاربردن تابع

gettype( ) نوع داده هریک از آنها را بر روی پنجره مرورگر اینترنت خود نمایش دهید.

۲- مقادیری را به دو متغیر نسبت دهید. با استفاده از عملگرهای مقایسه‌ای که در این درس با آنها آشنا شدید، موارد زیر را تشخیص دهید:

- بررسی مساوی بودن مقادیر متغیرها
- بررسی کوچک‌تر بودن مقدار متغیر اول از متغیر دوم
- بررسی بزرگ‌تر بودن مقدار متغیر اول از متغیر دوم
- بررسی بزرگ‌تر یا مساوی بودن مقدار متغیر اول با متغیر دوم

نتیجه هر بررسی را بر روی پنجره مرورگر نمایش دهید.

مقادیر نسبت داده شده به متغیرها را تغییر داده و برنامه را مجددا اجرا کنید.

ساعت پنجم

اجزای کنترل برنامه

تمام برنامه‌هایی که درساعت قبل نوشتیم همگی در یک جهت اجرا می‌شدند، بدین معنی که دستورالعملها و عباراتی که برنامه‌های PHP را تشکیل می‌دادند در اجراهای مختلف برنامه با یک ترتیب ثابت به اجرا در می‌آمدند. این وضعیت همان‌گونه که پیداست هیچ‌گونه انعطافی را به‌همراه ندارد.

در درس این ساعت قصد داریم شما را با ساختارهایی آشنا کنیم که امکان تطبیق برنامه‌ها در وضعیتها و شرایط مختلف را در اختیارمان قرار می‌دهند. در این درس با موارد زیر آشنا خواهید شد:

- **چگونگی استفاده از عبارت if برای اجرای بخشی از برنامه تحت شرایطی که حاصل یک عبارت منطقی برابر با مقدار true ارزیابی شود.**
- **چگونگی اجرای بخشهای مختلف یک برنامه تحت شرایطی که حاصل یک عبارت منطقی if برابر با مقدار false ارزیابی شود.**
- **چگونگی استفاده از عبارت switch جهت اجرای بخشی از یک برنامه بر مبنای مقدار بازگشتی حاصل از بررسی یک عبارت منطقی**
- **چگونگی تکرار اجرای بخشی از یک برنامه با استفاده از عبارت while**
- **چگونگی تکرار اجرای بخشی از یک برنامه با استفاده از عبارت for**
- **چگونگی متوقف کردن حلقه تکرار**
- **چگونگی استفاده تودرتو از حلقه‌های تکرار**
- **چگونگی استفاده از علایم شروع و پایان PHP در داخل ساختارهای کنترله برنامه**

در ادامه به بررسی موارد فوق می‌پردازیم:

تغییر در مسیر اجرای برنامه

در اغلب برنامه‌ها شاهد ارزیابی شرایط مختلف و تغییر رفتار برنامه براساس نتایج حاصل از این ارزیابی‌ها هستیم.امکان‌تصمیم‌گیری‌درتغییر مسیر اجرای برنامه باعث می‌شود تا در مورد اسناد PHP به صفحاتی پویا با قابلیت تغییر خروجی مطابق با شرایط مختلف برسیم. مانند بیشتر زبانهای‌برنامه‌نویسی، در زبان PHP4 نیز عبارتی با عنوان if موجود است که چنین امکانی را برای برنامه‌نویس فراهم می‌کند.

عبارت if

عبارت if روشی برای کنترل اجرای عبارتی است که در پی آن می‌آید (چنین عبارتی می‌تواند ساده یا متشکل از چندین عبارتی باشد که در درون جفت علامت { } واقع می‌شوند). عبارت if ابتدا به ارزیابی عبارتی که در درون جفت پرانتز به‌دنبال آن می‌آید، می‌پردازد چنان‌چه حاصل این ارزیابی برابر با مقدار true باشد، دستورالعمل یا دستورالعملهای بعد از آن اجرا خواهند شد و در غیر این‌صورت از اجرای آنها صرف‌نظر به‌عمل می‌آید. این فرآیند امکان تصمیم‌گیری بر مبنای هر تعداد دلخواهی از عوامل را در اختیار برنامه‌نویس قرار می‌دهد. شکل عمومی ساختار تصمیم‌گیری if به‌صورت زیر است:

```
if (expression) {
  // code to execute if the expression evaluates to true
}
```

برنامه‌لیست۱- ۵چگونگی استفاده از این ساختار را نشان می‌دهد. در این برنامه چنان‌چه متغیر تحت بررسی شامل دنباله کاراکتری " happy " باشد، بلوکی از کد که به‌دنبال آن می‌آید، اجرا می‌شود.

```
 1: <html>
 2: <head>
 3: <title>Listing 5.1</title>
 4: </head>
 5: <body>
 6: <?php
 7: $mood = "happy";
 8: if ( $mood == "happy" ) {
 9:     print "Hooray, I'm in a good mood";
10: }
11: ?>
12: </body>
13: </html>
```

لیست ۱- ۵ استفاده از عبارت if

همان‌گونه که مشاهده می‌کنید در این برنامه از عملگر مقایسه‌ای = = جهت مقایسه مقدار متغیر $ mood با دنباله کاراکتری " happy " استفاده شده‌است. اگر این دو باهم برابر باشند این بررسی به صورت true ارزیابی شده و بنابر این بلوکی از کد PHP که به‌دنبال عبارت if آمده است، اجرا می‌گردد. هرچند که در این برنامه بلوک کد مذکور در درون جفت علامت { } محصور شده‌است، اما

این کار ضرورتی نداشته و تنها در صورتی استفاده از این جفت علامت الزامی است که اجرای بیش از یک دستورالعمل مد نظر باشد. از این‌رو به‌جای ساختار فوق می‌توان از قطعه کد زیر نیز استفاده کرد:

```
if ( $mood = = " happy ")
Print " Hooray, I'm in a good mood ";
```

چنان‌چه مقدار ذخیره شده در متغیر $mood را به دنباله کاراکتری دیگری غیر از " happy " مثلاً " sad " تغییر دهید و برنامه را مجددا اجرا نمایید، عبارت منطقی داخل پرانتز به‌صورت false ارزیابی شده و از بلوک کدی که به‌دنبال آن می‌آید، صرف‌نظر خواهد شد. در این صورت شرایط به‌گونه‌ای است که گویی اصلاً برنامه‌ای در کار نبوده است.

استفاده از بخش else در عبارت if

هنگام کار با عبارت if اغلب مایلیم تا در صورتی که عبارت منطقی داخل پرانتز که بعد از if واقع می‌شود به‌صورت true ارزیابی نشود (به‌عبارت دیگر به‌صورت false ارزیابی شود)، بلوک دیگری از کد اجرا گردد. با اضافه کردن بخش else به عبارت if چنین فرآیندی امکان‌پذیر خواهد بود. شکل استفاده از if به‌همراه بخش else به‌صورت زیر است:

```
if ( expression) {
  // code to execute if the expression evaluates to true
} else {
  // code to execute in all other cases
}
```

برنامه PHP موجود در لیست ۲- ۵ برنامه لیست ۱- ۵ را با اضافه کردن بخش else به ساختار تصمیم‌گیری if توسعه داده است. چنان‌چه متغیر $mood معادل دنباله کاراکتری" happy " ارزیابی نشود دستورالعمل دیگری از برنامه اجرا خواهد شد.

```
 1: <html>
 2: <head>
 3: <title>Listing 5.2</title>
 4: </head>
 5: <body>
 6: <?php
 7: $mood = "sad";
 8: if ( $mood == "happy" ) {
 9:     print "Hooray, I'm in a good mood";
10: } else {
11:     print "Not happy but $mood";
12: }
13: ?>
14: </body>
15: </html>
```

لیست ۲- ۵ ساختار تصمیم‌گیری if به‌همراه بخش else

دقت کنید که در این برنامه ابتدا متغیر $mood با دنباله کاراکتری " sad " مقداردهی شده که البته معادل دنباله کاراکتری " happy " نمی‌باشد. از این‌رو عبارت منطقی‌بخش if (عبارت موجود در پرانتز) معادل با مقدارfalse ارزیابی شده و بنابراین بلوک کد مربوط به این بخش نادیده گرفته می‌شود. این بدان معنی است که بلوک کد مربوط به بخش else اجرا می‌گردد (به‌طور کلی اجرا نشدن کد مربوط به بخش if به معنی اجرای کد بخش else است و بالعکس). بنابراین پیغام " Not happy but sad " به نمایش در می‌آید.

بهره‌گیری از بخش else ساختار if به برنامه‌نویس اجازه می‌دهد تا با آزادی بیشتری فرآیند تصمیم‌گیری را پیاده‌سازی کند. PHP4 قادر است تا چندین عبارت منطقی را پشت سر هم یکی پس از دیگری مورد ارزیابی قرار دهد، بدین ترتیب که برنامه‌نویس باید چندین ساختار if / else را پشت سر هم مورد استفاده قرار دهد. قسمت بعد روش دیگری را برای انجام این کار پیشنهاد می‌کند.

استفاده از بخش elseif در عبارت if

به منظور ارزیابی چندین عبارت منطقی پیش از آرایه بلوکی از کد جهت اجرا می‌توان از ساختار ویژه‌ای در زبان PHP با عنوان else _ elseif _if استفاده کرد. شکل عمومی این ساختار تصمیم‌گیری به‌صورت زیر است:

```
if (expression) {
   // code to execute if the expression evaluates to true
} elseif (another expression) {
   // code to execute if the previous expression failed
   // and this one evaluates to true
} else {
   // code to execute in all other cases
}
```

چنان‌چه اولین عبارت بعد از if معادل با true ارزیابی نشود از بلوک کد مربوط به آن صرف‌نظر خواهد شد. تحت این شرایط عبارت دوم بعد از elseif مورد ارزیابی قرار می‌گیرد. باردیگر، اگر چنان‌چه این عبارت منطقی معادل true ارزیابی شود، بلوک کد واقع بعد از آن به اجرا درخواهد آمد. در غیر این‌صورت (اگر عبارت منطقی مورد بحث false ارزیابی شود) بلوک کد مربوط به بخش آخر، یعنی بخش else اجرا می‌شود. استفاده از هر تعداد بخش elseif به‌طور متوالی بلامانع است. ضمناً اگر برنامه‌نویس نیازی به اجرای دستورالعملهای بخش else نداشته باشد، به‌سادگی می‌تواند این بخش را حذف نماید. (در این صورت ساختار حاصل به شکل if - elseif خواهد بود).

> توجه‌کنید که می‌توان بخش elseif را به‌طور جداگانه به صورت else if نوشت. این امری است که به سلیقه برنامه‌نویس ارتباط دارد.

برنامه لیست ۳- ۵ مشابه برنامه قبلی است با این فرق که از بخش elseif استفاده شده است.

```
 1: <html>
 2: <head>
 3: <title>Listing 5.3</title>
 4: </head>
 5: <body>
 6: <?php
 7: $mood = "sad";
 8: if ( $mood == "happy" ) {
 9:     print "Hooray, I'm in a good mood";
10: } elseif ( $mood == "sad" ) {
11:     print "Awww. Don't be down!";
12: } else {
13:     print "Neither happy nor sad but $mood";
14: }
15: ?>
16: </body>
17: </html>
```

لیست ۳- ۵ بهره‌گیری از ساختار if به‌همراه بخشهای else و elseif

همان‌گونه که مشاهده می‌کنید در این برنامه نیز مانند برنامه قبل متغیر $mood با دنباله کاراکتری " sad " مقداردهی شده است. از آنجا که این دنباله کاراکتری با دنباله کاراکتری"happy" معادل نیست اولین بلوک کد که مربوط به بخش if است، نادیده گرفته می‌شود. سپس در بخش elseif بررسی معادل بودن دنباله کاراکتری " sad " با محتوای متغیر $mood انجام می‌شود. حاصل این ارزیابی true است چرا که هر دو مقدار فوق معادلند. از این‌رو بلوک کد مربوط به بخش elseif اجرا می‌گردد. دستورالعملهای پیش‌فرض این ساختار در خطوط ۱۲ تا ۱۴ برنامه مشاهده می‌شوند. (منظور از دستورالعملهای پیش‌فرض بلوک کد مربوط به بخش else در ساختار تصمیم‌گیری if- elseif- else است). چنان‌چه هیچ یک از عبارات شرطی این ساختار معادل true ارزیابی نشوند، بلوک مربوط به else اجرا شده و به‌سادگی پیغامی شامل مقدار متغیر $mood را بر روی صفحه مرورگر نشان می‌دهد.

استفاده از ساختار تصمیم‌گیری Switch

عبارت switch روش دیگری است برای تغییر مسیر روند اجرای برنامه بر مبنای عبارت شرطی که در این ساختار مورد ارزیابی قرار می‌گیرد. تفاوتهای کلیدی مابین دو ساختار تصمیم‌گیری if و switch وجود دارد. با استفاده از ساختار تصمیم‌گیری if به‌همراه بخش elseif می‌توان چندین عبارت شرطی را مورد ارزیابی قرار داد. این در حالی است که در ساختار تصمیم‌گیری switch تنها یک عبارت شرطی مورد ارزیابی قرارگرفته و بر مبنای نتیجه این ارزیابی بلوک کدهای مختلفی به اجرا درخواهند آمد. البته این به شرطی است که عبارت مورد بررسی معادل با یکی از انواع داده‌های عددی (integer یا double)، دنباله کاراکتری و یا نوع داده boolean باشد(نتیجه حاصل از ارزیابی که به‌عنوان بخشی از

عبارت if مورد استفاده قرار می‌گیرد، همان‌گونه که در قسمت قبل مشاهده کردید، یکی از مقادیر true یا false (یعنی مقداری از نوع داده boolean) است). حاصل عبارت شرطی ساختار تصمیم‌گیری switch را با هر تعداد دلخواهی از مقادیر می‌توان مقایسه کرد. شکل عمومی استفاده از این ساختار چنین است:

```
switch
case result 1 :
   // execute this if expression results in result 1
   break ;
case result 2 :
   // execute this if expression results in result 2
   break ;
default :
   // execute this if no break statement
   // has been encountered hitherto
}
(expression) {
```

معمولاً عبارت شرطی ساختار تصمیم‌گیری switch به‌سادگی یک متغیر است. در درون بلوک کد ساختار switch می‌توان چندین عبارت case را مشاهده کرد. هر یک از این عبارات مقداری را با عبارت شرطی ساختار switch مقایسه می‌کنند. چنان‌چه حاصل مقایسه‌ای در این بین مثبت ارزیابی شود، بلوک کد بعد از آن عبارت case اجرا خواهد شد (بلوک کد هر عبارت case شامل دستورالعملهایی است که مابین این عبارت case و عبارت case بعدی واقع می‌شود ـ مترجم). دستورالعمل break موجب پایان اجرای عملکرد کل ساختار switch می‌گردد. در صورت حذف این دستورالعمل عبارت case بعدی به ناچار مورد ارزیابی قرار خواهد گرفت و در نهایت، در صورت استفاده از بخش اختیاری default در ساختار switch بلوک کد مربوط به آن اجرا می‌شود (استفاده از break الزامی نیست).

> همواره سعی کنید تا استفاده از دستورالعمل break در هر یک از عبارات case مربوط به ساختار تصمیم‌گیری switch را در دستور کار خود قرار دهید. بدون استفاده از این دستورالعمل روند اجرای برنامه در سایر عبارات case و در نهایت بخش default این ساختار را ادامه پیدا خواهد کرد. در بیشتر موارد این رفتار آن چیزی نیست که مورد انتظار ما باشد چرا که بسیاری از بررسی‌ها و ارزیابی‌ها بیهوده صورت خواهند گرفت.

برنامه موجود در لیست ۴- ۵ با استفاده از ساختار تصمیم‌گیری switch نسخه دیگری از برنامه لیست قبل را که توسط ساختار if- elseif- else پیاده‌سازی شده بود، نشان می‌دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 5.4</title>
 4: </head>
 5: <body>
 6: <?php
 7: $mood = "sad";
 8: switch ( $mood ) {
 9:     case "happy":
10:         print "Hooray, I'm in a good mood";
11:         break;
12:     case "sad":
13:         print "Awww. Don't be down!";
14:         break;
15:     default:
16:         print "Neither happy nor sad but $mood";
17: }
18: ?>
19: </body>
20: </html>
```

لیست ۴- ۵ استفاده از ساختار switch

در لیست فوق بار دیگر متغیر $mood با دنباله کاراکتری " sad " مقداردهی شده است. این مقدار در ساختار تصمیم switch جهت مقایسه با سایر مقادیر مورد استفاده قرار می‌گیرد. اولین عبارت case بررسی معادل بودن مقدار متغیر مذکور با دنباله کاراکتری " happy " را انجام می‌دهد. از آنجا که حاصل این ارزیابی معادل false است، لذا اجرای case دوم با بررسی معادل بودن متغیر مورد بحث با دنباله کاراکتری " sad " در دستور کار برنامه قرار می‌گیرد. به‌دلیل اینکه این دو مقدار معادل یکدیگر ارزیابی می‌شوند، بلوک کد مربوط به این عبارت case به اجرا در می‌آید. دستورالعمل break در انتهای این بلوک کد منجر به خروج از ساختار switch و خاتمه دادن به فرآیند مربوطه می‌گردد.

استفاده از عملگر؟

عملگر؟ به سادگی عملکردی مشابه عبارت if را در اختیار برنامه‌نویس قرار می دهد، اما با این تفاوت که یکی از مقادیری را که بعد از این عملگر واقع شده و با علامت کولون (:) از یکدیگر جدا شده‌اند، باز می‌گرداند. اینکه حاصل استفاده از این عملگر کدام یک از این مقادیر است، به ارزیابی عبارتی بستگی دارد که پیش از عملگر؟ در درون پرانتز واقع می‌شود.

شکل عمومی استفاده از این عملگر به‌صورت زیر است:

```
(expression) ? returned _ if _ expression _ is _ true :
               returned_ if _ expression _ is _ false ;
```

چنان‌چه حاصل ارزیابی عبارت واقع پیش از عملگر؟ مقدار true باشد مقدار (یا عبارت) اول پس از آن به‌عنوان نتیجه باز می‌گردد و در غیر این‌صورت دومین مقدار (یا عبارت) پس از آن به‌عنوان

نتیجه حاصل خواهد شد. برنامه موجود در لیست ۵- ۵ از این عملگر جهت تنظیم مقدار یک متغیر بر مبنای مقدار ذخیره شده در متغیر $mood بهره می‌برد.

```
1: <html>
2: <head>
3: <title>Listing 5.5</title>
4: </head>
5: <body>
6: <?php
7: $mood = "sad";
8: $text = ( $mood=="happy" )?"Hooray, I'm in a good mood":"Not happy but
➥$mood";
9: print "$text";
10: ?>
11: </body>
12: </html>
```

لیست ۵- ۵ استفاده از عملگر ?

به‌مانند کدهای قبلی در این‌جا نیز متغیر $mood با دنباله کاراکتری " sad "مقداردهی شده است. در خط ۸ از برنامه مقدار این متغیر با دنباله کاراکتری " happy " مورد مقایسه قرار گرفته‌است. از آنجا که حاصل این مقایسه چیزی جز مقدار false نیست، عبارت " Not happy but $mood " که یک دنباله کاراکتری است به‌عنوان نتیجه حاصل می‌شود.

عملگر? از نقطه نظر خوانایی مشکل می‌نماید اما در مواقعی که تنها با دو امکان (و نه بیشتر) مواجه باشیم و بخواهیم از کوتاه نویسی کد بهره‌مند شویم، مفید واقع می‌شود.

حلقه‌های تکرار

تا به اینجا امکان و قابلیت تصمیم‌گیری برنامه و اجرای بخشهای مختلف کد بر مبنای نتیجه تصمیم‌گیری به‌اندازه کافی بحث و بررسی کردیم. علاوه بر تصمیم‌گیری، برنامه می‌تواند قطعه‌ای از کد را به دفعات مکرر پشت سر هم اجرا کند. ساختارهایی که چنین امکانی را در اختیار برنامه‌نویس قرار می‌دهند به ساختارهای تکرار شهرت دارند. ساختارهای تکرار جهت اجرای تکراری وظایف در برنامه طراحی شده‌اند. تقریباً در تمامی موارد ساختارهای تکرار تا زمانی که شرایط خاصی حاصل شود یا برنامه‌نویس به‌طور صریح فرآیند تکرار را متوقف کند، به کار خود ادامه می‌دهند.

ساختار تکرار while

شکل عمومی این ساختار تکرار مشابه ساختار تصمیم if است:

```
While (expression) {
    // do something
}
```

به‌شرطی که عبارت مورد ارزیابی در این ساختار معادل مقدار true باشد بلوک کد بعد از آن بارها و بارها به‌طور مکرر اجرا می‌شود. معمولاً به‌هر بار اجرای بلوک کد موجود در این‌گونه ساختارها یک تکرار گفته می‌شود. در داخل بلوک مورد بحث، معمولاً شرایط در نهایت به‌گونه‌ای تغییر می‌کند که بر روی عبارت مورد ارزیابی تأثیر می‌گذارد. این تأثیر به‌گونه‌ای است که باعث می‌شود تا اجرای مکرر دستورالعملهای حلقه متوقف شود. چه در غیر این‌صورت اجرای حلقه تا بی‌نهایت ادامه خواهد داشت. برنامه موجود در لیست ۶- ۵ با استفاده از ساختار تکرار while ضرایب عدد صحیح ۲ را تا محاسبه عدد ۲۴ بر روی صفحه نشان می‌دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 5.6</title>
 4: </head>
 5: <body>
 6: <?php
 7: $counter = 1;
 8: while ( $counter <= 12 ) {
 9:     print "$counter times 2 is ".($counter*2)."<br>";
10:     $counter++;
11: }
12: ?>
13: </body>
14: </html>
```

لیست ۶- ۵ استفاده از حلقه تکرار while

همان‌گونه که مشاهده می‌کنید در این برنامه ابتدا متغیر $counter در خط ۷ مقداردهی شده است. در خط ۸ از این برنامه مقدار متغیر مذکور در قالب یک عبارت منطقی با عدد 12 مورد مقایسه قرار می‌گیرد. به‌شرطی که عدد صحیح ذخیره شده در متغیر $counter کوچک‌تر یا برابر با عدد 12 باشد حلقه به اجرای دستورالعملهای خود ادامه خواهد داد. در داخل بلوک کد while مقدار ذخیره شده در متغیر $counter در هر بار اجرای حلقه، ضرب در ۲ شده و نتیجه حاصل بر روی صفحه به نمایش درمی‌آید. سپس در خط ۱۰ مقدار این متغیر به اندازه یک واحد افزایش می‌یابد. عمل فوق در این بین از اهمیت فوق‌العاده‌ای برخوردار است. چنان‌چه تغییر مقدار ذخیره شده در این متغیر فراموش شود عبارت مورد ارزیابی در بخش while هرگز نتیجه false را در پی نخواهد داشت. این بدان معنی است که حلقه تا ابد اجرا خواهد شد.

ساختار تکرار do ... while

ساختار تکرار do ... while مشابه ساختار تکرار while است با این تفاوت که عنوان آن اندکی تغییر یافته است. تفاوت اساسی مابین این دو ساختار در این است که در ساختار do ... while ارزیابی شرایط تکرار حلقه بعد از بلوک کد مربوطه انجام می‌شود. شکل عمومی این ساختار به‌صورت زیر است:

```
do {
   // code to be executed
} while (expression) ;
```

وجود علامت سمی‌کولون (;) بعد از عبارت مورد ارزیابی در ساختار تکرار do ... while الزامی است.

این ساختار احتمالاً هنگامی مفید واقع می‌شود که حتی به‌ازای ارزیابی عبارت شرطی به‌صورت false نیز بخواهیم دست کم یک مرتبه بلوک کد مربوطه را اجرا نماییم. برنامه موجود در لیست ۷-۵، استفاده از این ساختار را نشان می‌دهد. واضح است که بلوک کد داخل این ساختار دست کم یک مرتبه اجرا خواهد شد.

```
1: <html>
2: <head>
3: <title>Listing 5.7</title>
4: </head>
5: <body>
6: <?php
7: $num = 1;
8: do {
9:     print "Execution number: $num<br>\n";
10:    $num++;
11: } while ( $num > 200 && $num < 400 );
12: ?>
13: </body>
14: </html>
```

لیست ۷- ۵ استفاده از ساختار تکرار do ... while

همان‌گونه که مشاهده می‌کنید، ساختار do ... while مقدار ذخیره شده در متغیر $num را برای بررسی اینکه آیا این مقدار بزرگ‌تر از 200 و کوچک‌تر از 400 است، مورد ارزیابی قرار می‌دهد. در سطر ۷ از این برنامه متغیر $num ابتدا با عدد 1 مقداردهی شده است. از آنجا که این مقدار در شرایط مذکور صدق نمی‌کند، عبارت شرطی بخش while معادل false ارزیابی می‌شود. اما به‌دلیل اینکه این ارزیابی بعد از بلوک کد ساختار فوق انجام می‌شود، دستورالعملهای این بلوک بدون هیچ پیش‌شرطی اجرا می‌شوند. بنابراین خروجی برنامه لیست فوق چاپ یک پیغام بر روی صفحه است.

ساختار تکرار for

ساختار تکرار for بسیار شبیه به ساختار تکرار while است. اگر انجام فرآیندی توسط while امکان‌پذیر نباشد، توسط ساختار for نیز نمی‌توان آن‌را پیاده سازی کرد. با این حال ساختار for در اغلب موارد همان قابلیت و کارایی while را البته به‌صورت واضح‌تر و خواناتری در اختیار برنامه‌نویس

قرار می‌دهد. پیشتر همان‌گونه که در برنامه لیست ۶- ۵ مشاهده کردید، متغیری را در خارج از ساختار while مقداردهی کردیم. این بدان دلیل بود که ساختار مذکور بتواند مقدار آن متغیر را در قالب یک عبارت شرطی مورد ارزیابی قرار دهد. در درون ساختار فوق مقدار این متغیر به‌ازای هر بار اجرای بلوک کد مربوطه یک واحد افزایش پیدا می‌کرد. با استفاده از ساختار for می‌توانیم تمامی فرآیند مورد بحث، یعنی مقداردهی اولیه متغیر، ارزیابی عبارت شرطی و در نهایت اضافه کردن مقدار متغیر فوق را تنها در قالب یک خط به‌سادگی انجام دهیم. این روش امکان خلاصه نویسی بیشتری را در اختیار برنامه‌نویس قرار می‌دهد. ضمن اینکه احتمال ایجاد حلقه‌های بی‌نهایت را نیز بسیار کاهش می‌دهد چراکه در این روش بعید است برنامه‌نویس افزایش متغیر شمارنده (کنترل کننده) حلقه را فراموش کند. شکل عمومی این ساختار به‌صورت زیر است:

```
for (initialization expression ; test expression ; modification expression) {
   // code to be executed
}
```

همان‌گونه که مشاهده می‌کنید، عبارات موجود در پرانتز جلوی for با استفاده از علامت سمی‌کولون از یکدیگر تفکیک شده‌اند.

معمولاً اولین عبارت داخل پرانتز متغیر شمارنده‌ای را مقداردهی می‌کند. عبارت دوم یک عبارت شرطی است که شرایط توقف حلقه را مشخص می‌کند. و عبارت سوم مقدار متغیر شمارنده را افزایش می‌دهد. برنامه موجود در لیست ۸- ۵ با استفاده از ساختار for نسخه دیگری از برنامه لیست ۶-۵ که با ساختار while نوشته شده و ضرایب عدد 2 را تا عدد 24 به دست می‌دهد را در اختیارتان قرار می‌دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 5.8</title>
 4: </head>
 5: <body>
 6: <?php
 7: for ( $counter=1; $counter<=12; $counter++ ) {
 8:     print "$counter times 2 is ".($counter*2)."<br>";
 9: }
10: ?>
11: </body>
12: </html>
```

لیست ۸- ۵ استفاده از ساختار for

نتیجه حاصل از اجرای برنامه‌های موجود در دو لیست ۶- ۵ و ۸- ۵ دقیقاً مشابهند. با این همه، همان‌گونه که از ظاهر برنامه لیست ۸- ۵ نیز مشاهده می‌کنید، ساختار for امکان خلاصه‌نویسی بیشتری را در اختیار برنامه‌نویس قرار می‌دهد. از آنجا که فرآیند مقداردهی اولیه و افزایش مقدار متغیر $counter در ابتدای این ساختار انجام می‌شود، منطق ساختار تکرار مورد نظر در نگاه نخست بسیار

واضح و خوانا می‌نماید. در خط ۷ از این برنامه، در درون پرانتز جلوی for، اولین عبارت متغیر $counter را با عدد 1 مقداردهی اولیه می‌کند. عبارت دوم مقایسه‌ای را جهت ارزیابی این‌که متغیر $counter مقداری برابر یا کوچک‌تر از عدد 12 دارد، انجام می‌دهد و در نهایت آخرین عبارت مقدار متغیر $counter را یک واحد افزایش می‌دهد.

هنگامی که اجرای برنامه به ساختار تکرار for می‌رسد، ابتدا متغیر $counter مقداردهی شده و سپس عبارت شرطی کنترل کننده حلقه مورد ارزیابی قرار می‌گیرد. چنان‌چه این عبارت معادل با مقدار true ارزیابی شود، بلوک کد داخل این ساختار به اجرا در می‌آید. پس از اجرای این بلوک کد مقدار متغیر $counter یک واحد افزایش یافته و عبارت شرطی کنترل کننده حلقه مجدداً مورد ارزیابی قرار می‌گیرد. این فرآیند تا جایی ادامه پیدا می‌کند که عبارت شرطی مذکور معادل با مقدار false ارزیابی شود.

خروج از حلقه با استفاده از دستورالعمل break

هر دو ساختار حلقه while و for شامل عبارتی شرطی هستند که نتیجه ارزیابی آن شرایط خروج از حلقه را مشخص می‌کند. با این وجود دستورالعمل دیگری با عنوان break قادر است تا اجرای حلقه‌ها را در نقطه دلخواه پایان دهد (معمولاً این دستورالعمل به‌عنوان نتیجه شرط دیگری به اجرا در می‌آید). این دستورالعمل می‌تواند حفاظتی در برابر خطاهای احتمالی باشد. برنامه موجود در لیست ۹- ۵ شامل یک ساختار تکرار for است که به‌طور مکرر عدد صحیح بزرگی را به متغیری که مقدار آن دائماً افزایش پیدا می‌کند، تقسیم کرده و نتیجه حاصل را بر روی صفحه نشان می‌دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 5.9</title>
 4: </head>
 5: <body>
 6: <?php
 7: for ( $counter=1; $counter <= 10; $counter++ ) {
 8:     $temp = 4000/$counter;
 9:     print "4000 divided by $counter is... $temp<br>";
10: }
11: ?>
12: </body>
13: </html>
```

لیست ۹- ۵ استفاده از ساختار for جهت تقسیم عدد صحیح 4000 به متغیری که مقدار آن در درون حلقه افزایش می‌یابد.

همان‌گونه که مشاهده می‌کنید در خط ۷ از این برنامه متغیر $counter با عدد 1 مقداردهی شده است. بخش عبارت شرطی ساختار for مقدار این متغیر را جهت بررسی اینکه کوچک‌تر یا برابر مقدار عددی 10 است، ارزیابی می‌کند. در بلوک کد این ساختار عدد 4000 بر مقدار متغیر $counter

تقسیم شده و در نهایت نتیجه حاصل از این تقسیم بر روی صفحه به نمایش در می‌آید.

در نگاه اول این برنامه کاملاً کارآمد و بدون نقص به‌نظر می‌رسد. اما اگر مقدار اولیه متغیر $counter از طریق ورودی دریافت شود، وضعیت باز هم به همین ترتیب خواهد بود؟ مقدار ورودی ممکن است یک عدد منفی یا از آن‌هم بدتر یک دنباله کاراکتری باشد. تغییر مقدار اولیه متغیر $counter از عدد 1 به 4 - موجب می‌شود که به ازای پنجمین اجرای حلقه با یک تقسیم بر صفر که عملاًغیرممکن‌و تعریف نشده است، مواجه شویم. این وضعیت، به وضوح زنگ خطری را برای برنامه‌نویس به‌صدا در می‌آورد. لیست ۱۰- ۵ شامل برنامه‌ای است که به‌واسطه بهره‌گیری از دستورالعمل break در صورتی که متغیر $counter شامل مقدار صفر باشد، به اجرای حلقه پیش از انجام هر گونه عملیاتی پایان می‌دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 5.10</title>
 4: </head>
 5: <body>
 6: <?php
 7: $counter = -4;
 8: for ( ; $counter <= 10; $counter++ ) {
 9:     if ( $counter == 0 )
10:         break;
11:     $temp = 4000/$counter;
12:     print "4000 divided by $counter is... $temp<br>";
13: }
14: ?>
15: </body>
16: </html>
```

لیست ۱۰- ۵ استفاده از دستورالعمل break

به‌خاطر داشته باشید که فرآیند تقسیم بر صفر موجب خطای مخرب و خطرناکی در PHP نبوده و برنامه تنها با نمایش یک پیغام اخطار به اجرای خود ادامه می‌دهد.

همان‌گونه که مشاهده می‌کنید در خط ۹ از این برنامه مقدار متغیر $counter در قالب یک عبارت if مورد بررسی قرارگرفته است. اگر مقدار متغیر مذکور برابر با صفر باشد، دستورالعمل break اجرا شده و به عملیات بلوک کد داخل حلقه پایان داده می‌شود، به‌گونه‌ای که برنامه به اجرای دستورالعملهای واقع پس از ساختار for می‌پردازد.

توجه کنید که عمل مقداردهی اولیه متغیر $counter در خط ۷، یعنی خارج از ساختار تکرار for انجام شده است. این‌کار به این خاطر است که خواننده بتواند تصوری از مقداردهی متغیر $counter از طریق ورودی یا از طریق مراجعه برنامه به یک بانک اطلاعاتی داشته باشد.

به‌خاطر داشته باشید که حذف هر سه عبارت مقداردهی اولیه شمارنده، شرط بررسی ادامه حلقه و افزایش مقدار متغیر امکان‌پذیر است اما وجود سمی‌کولون‌ها در هر صورت الزامی می‌باشد.

صرف‌نظر از اجرای حلقه با استفاده از دستورالعمل continue

دستورالعمل continue بلافاصله به اجرای حلقه جاری خاتمه داده ولی موجب خروج از ساختار حلقه و خاتمه دادن به کار آن نمی‌شود، بدین معنی که اجرای حلقه بعدی بلافاصله آغاز می‌گردد. استفاده از دستورالعمل break در برنامه لیست ۱۰- ۵ شاید اندکی جسورانه به‌نظر برسد ضمن اینکه در برخی از کاربردها به چنین خروجی از ساختار حلقه نیاز نباشد. به کمک دستورالعمل continue در برنامه لیست ۱۱- ۵ می‌توان بدون خاتمه دادن به کار ساختار حلقه و خروج کلی از آن از وقوع رخداد ناخواسته تقسیم بر صفر نیز جلوگیری به‌عمل آورد.

```
 1: <html>
 2: <head>
 3: <title>Listing 5.11</title>
 4: </head>
 5: <body>
 6: <?php
 7: $counter = -4,
 8: for ( ;  $counter <= 10; $counter++ ) {
 9:     if ( $counter == 0 )
10:         continue;
11:     $temp = 4000/$counter;
12:     print "4000 divided by $counter is... $temp<br>";
13: }
14: ?>
15: </body>
16: </html>
```

لیست ۱۱- ۵ استفاده از دستورالعمل continue

همان‌گونه که مشاهده می‌کنید در خط ۱۰ از این برنامه دستورالعمل continue جایگزین دستورالعمل break در برنامه قبلی شده است. اکنون اگر متغیر $counter برابر با صفر باشد، از اجرای مابقی‌دستورالعملهای‌حلقه جاری خودداری به‌عمل آمده و حلقه بعدی بلافاصله کار خود را آغاز می‌کند.

توجه به این نکته ضروری است که علی‌رغم کارایی ممکن، دستورالعملهای break و continueمی‌توانندازمیزان‌وضوح و خوانایی برنامه بکاهند. از آنجا که این دو دستورالعمل معمولاً پیچیدگی منطق حلقه‌های تکرار را افزایش می‌دهند. بهتر است هنگام استفاده از آنها دقت بیشتری به خرج دهید.

حلقه‌های تودرتو

هر ساختار حلقه‌ای می‌تواند شامل ساختار یا ساختارهای حلقه دیگری باشد. از این ترکیب به ویژه‌می‌توان‌هنگام کار باجداولی از HTML که به‌صورت پویا در درون برنامه‌های PHP ساخته می‌شوند، استفاده کرد. برنامه لیست ۱۲- ۵ چگونگی استفاده تودرتو از دو حلقه for را جهت نمایش یک جدول ضرب معمولی بر روی صفحه نمایش نشان می‌دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 5.12</title>
 4: </head>
 5: <body>
 6: <?php
 7: print "<table border=\"1\">\n";
 8: for ( $y=1; $y<=12; $y++ ) {
 9:     print "<tr>\n";
10:     for ( $x=1; $x<=12; $x++ ) {
11:         print "\t<td>";
12:         print ($x*$y);
13:         print "</td>\n";
14:     }
15:     print "</tr>\n";
16: }
17: print "</table>";
18: ?>
19: </body>
20: </html>
```

لیست ۱۲- ۵ استفاده از حلقه‌های تودرتو

پیش از بررسی عملکرد این دو حلقه تودرتوی for ابتدا اجازه دهید تا نگاهی سریع به خط ۷ از برنامه لیست ۱۲- ۵ بیندازیم. این خط را بار دیگر در اینجا تکرار می‌کنیم:

```
print " < table border = \" 1 \" >\ n " ;
```

همان‌گونه که مشاهده می‌کنید در درون رشته کاراکتری فوق، پیش از هر علامت کوتیشن از علامت \ استفاده کرده‌ایم. این امر ضروری است چراکه در این حالت موتور PHP علامت کوتیشن را به‌عنوان نقطه شروع یا پایان دنباله کاراکتری تلقی نکرده و متوجه این نکته می‌شود که منظور برنامه‌نویس از این کار به کارگیری علامت کوتیشن در درون دنباله کاراکتری بوده‌است. چنان‌چه در این‌گونه مواقع از به‌کارگیری علامت \ قبل از علامت کوتیشن خودداری کنیم، چنین عبارتی برای موتور PHP مفهوم واقعی خود را نداشته و موتور PHP به‌سادگی آن‌را رشته‌ای کاراکتری تلقی می‌کند که به‌دنبال آن یک عدد (در این جا عدد 1) و به‌دنبال آن نیز رشته کاراکتری دیگری واقع شده است. این وضعیت موجب تولید خطا در برنامه خواهد شد. به استفاده از علامت \ پیش از علامت کوتیشن (و برخی علایم دیگر) معمولاً تکنیک escaping یا گریز گفته می‌شود. در درس هفتم با عنوان " آرایه‌ها " باز هم در این مورد بحث می‌کنیم.

و اما در مورد برنامه لیست ۱۲- ۵، همان‌گونه که مشاهده می‌کنید حلقه for خارجی در خط ۸ متغیری با نام $y را با مقدار اولیه 1 معرفی می‌کند. عبارت شرطی این ساختار شامل بررسی کوچک‌تر یا مساوی بودن عدد ذخیره شده در این متغیر با عدد 12 است و طبق معمول عبارت دیگری نیز در این ساختار عهده‌دار افزایش مقدار این متغیر است. به‌ازای هربار اجرای حلقه، بلوک کد داخل حلقه در خط ۹، المانی از زبان نشانه‌گذاری HTML را با عنوان TR جهت نمایش سطری از جدول چاپ کرده و ساختار حلقه for دیگری را نیز در خط ۱۰ تعریف می‌کند. در ساختار حلقه درونی متغیری با عنوان $x مقداردهی شده و شرایطی مشابه شرایط ساختار حلقه خارجی به این متغیر نیز اعمال می‌گردد. به ازای هربار اجرای حلقه داخلی المان دیگری از زبان HTML با عنوان TD جهت نمایش ستونی از جدول در خط ۱۱ مورد استفاده قرار گرفته و در خط ۱۲ نیز حاصل‌ضرب مقادیر دو متغیر $x و $y محاسبه و به‌نمایش در می‌آید. در خط ۱۳ از برنامه، دستورالعمل لازم جهت بستن هر سلول جدول اجرا می‌شود. پس از تکمیل اجرای حلقه داخلی، حلقه خارجی در خط ۱۵ دستورالعمل بستن سطر مورد نظر از جدول را اجرا می‌کند. بدین ترتیب فرآیند فوق آماده است تا مجدداً از ابتدا به اجرا درآید. پس از تکمیل حلقه خارجی نتیجه حاصل جدول ضربی از اعداد است که به‌خوبی جهت نمایش قالب‌بندی شده‌است. قالب بندی جدول با اجرای دستورالعمل خط ۱۷ تکمیل می‌شود.

بلوک‌های کد و نمایش خروجی در مرورگر اینترنت

همان‌گونه که در درس ساعت سوم نیز عنوان شد، به‌هنگام نیاز می‌توانیم با استفاده از علایم ویژه‌ای از حالت HTML به PHP (و بالعکس) سوئیچ کنیم. در درس این ساعت متوجه شدید که با توجه به نتایج حاصل از تصمیم‌گیری با استفاده از ساختارهای کنترلی مانند if و switch می‌توانیم خروجی‌های مختلفی را در اختیار کاربران قرار دهیم. در این بخش قصد داریم تا این دو مقوله را با یکدیگر ترکیب کنیم.

برنامه اسکریپتی را درنظر بگیرید که تنها در صورت ارزیابی متغیر ویژه‌ای به‌صورت true جدولی از مقادیر را به‌عنوان خروجی نمایش دهد. برنامه لیست ۱۳- ۵ چگونگی ساخت جدول ساده‌ای را با استفاده از علایم نشانه‌گذاری HTML و بهره‌گیری از عبارت شرطی if به‌همراه آن‌را نشان می‌دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 5.13</title>
 4: </head>
 5: <body>
 6: <?php
 7: $display_prices = true;
 8: if ( $display_prices ) {
 9:     print "<table border=\"1\">";
10:     print "<tr><td colspan=\"3\">";
11:     print "todays prices in dollars";
12:     print "</td></tr>";
13:     print "<td>14</td><td>32</td><td>71</td>";
14:     print "</tr></table>";
15: }
16: ?>
17: </body>
18: </html>
```

لیست ۱۳- ۵ بلوکی از کد PHP شامل چندین عبارت ( )print

در صورتی که مقدار متغیر $dis play _ prices در خط ۷ از این برنامه معادل مقدار true ارزیابی شود، جدول مورد نظر در خروجی چاپ خواهد شد. به‌دلیل افزایش خوانایی برنامه در این لیست از چندین عبارت ( )print جهت تولید خروجی استفاده شده است (باردیگر خاطر نشان می‌کنیم که جهت استفاده از علامت کوتیشن در درون دنباله‌های کاراکتری از روش escaping بهره‌گرفته‌ایم). این‌گونه استفاده از عبارت ( )print هیچ مشکلی در پی ندارد، اما برای کاهش میزان تایپ و استفاده از خلاصه نویسی می‌توانیم به کمک علایم شروع و پایان کد PHP به‌راحتی به حالت HTML سوئیچ کرده و تنها دستورالعملهای نشانه‌گذاری HTML را تایپ کنیم. در برنامه لیست ۱۴-۵ چنین کاری را انجام داده‌ایم.

```
 1: <html>
 2: <head>
 3: <title>Listing 5.14</title>
 4: </head>
 5: <body>
 6: <?php
 7: $display_prices = true;
 8: if ( $display_prices ) {
 9: ?>
10:     <table border="1">
11:     <tr><td colspan="3">todays prices in dollars</td></tr>
12:     <td>14</td><td>32</td><td>71</td>
13:     </tr></table>
14: <?php
15: }
16: ?>
17: </body>
18: </html>
```

لیست ۱۴- ۵ استفاده به‌جا از علایم شروع و پایان کد PHP جهت سوئیچ به حالت HTML

نکته مهم و قابل توجه در این برنامه این است که فرآیند انتقال از کد PHP به HTML تنها در صورتی انجام می‌شود که عبارت شرطی مورد بررسی در خط ۹ معادل true ارزیابی شود. این روش جالب ما را از به‌کارگیری روش escaping جهت استفاده از علامت کوتیشن و درج کد HTML در قالب عبارت ( )print معاف می‌کند. با این حال روش مذکور در دراز مدت و به‌ویژه در مورد برنامه‌های طولانی می‌تواند موجب کاهش میزان خوانی برنامه شود.

جمع‌بندی

در درس این ساعت مطالب مفیدی درباره ساختارهای کنترل برنامه و روشهای مفید دیگر جهت ایجاد برنامه‌های منعطف و پویا فراگرفتید. بیشتر این ساختارهای کنترل را به‌طور مکرر در بخشهای باقیمانده از این کتاب مورد استفاده و بهره‌برداری قرار خواهیم داد.

چگونگی تعریف عبارت if و نیز چگونگی اجرای بخشهای مختلف برنامه بر مبنای نتایج حاصل از بررسی عبارت شرطی بخشهای elseif و else از عبارت if مطلبی بود که درس این ساعت را با آن آغاز کردیم. در ادامه درس چگونگی استفاده از ساختار تصمیم‌گیری switch را جهت تغییر مسیر اجرای برنامه به‌واسطه بررسی عبارات مختلف در بخشهای case مربوط به این ساختار تشریح کردیم. سپس به بحث و بررسی درباره ساختارهای تکرار و استفاده از حلقه در برنامه پرداخته و در این میان توجه خود را به ساختارهای تکرار while و for معطوف نمودیم. همچنین به بحث در مورد نحوه استفاده از دستورالعملهای break و continue پرداخته و اهمیت آنها را در مورد خاتمه اجرای ساختار حلقه و خروج از آن و نیز صرف‌نظر از اجرای حلقه جاری و اجرای حلقه بعدی، متذکر شدیم. بهره‌گیری تودرتو از حلقه‌های تکرار در درون یکدیگر و بررسی یک برنامه نمونه در این زمینه مطلب دیگری بود که در این درس به آن پرداختیم و در انتها نیز به روش مفید و جالبی برای انتقال از کد PHP به HTML و بالعکس با استفاده از علایم شروع و پایان کد PHP اشاره کرده و برنامه‌ای را در این زمینه بررسی کردیم.

هم‌اینک با مطالبی که تا اینجای کتاب فراگرفتید باید اطلاعات کافی جهت نوشتن اسکریپت‌های مورد نظرتان را داشته باشید. برنامه‌هایی که می‌نویسید اکنون باید بتوانند تصمیم‌گیری کرده و بخشهایی از کد را به‌طور تکراری اجرا کنند.

در درس ساعت بعدی ابزارهای دیگری را که موجب افزایش توان شما در برنامه‌نویسی جهت ایجاد برنامه‌های کارآمدتر می‌شوند، مورد بحث و بررسی قرار می‌دهیم. توابع امکاناتی نظیر ساماندهی برنامه ها، جلوگیری از دوباره کاری و مهم‌تر از همه امکان استفاده مجدد از کدهای نوشته شده را در اختیارتان قرار می‌دهند.

پرسش و پاسخ

پرسش: آیا عبارات شرطی ساختارهای کنترل لزوماً باید حاصلی از نوع داده boolean داشته باشند؟

پاسخ: نهایتاً بله، اما توجه داشته باشید که در زمینه بررسی عبارات شرطی، مقدار صفر، متغیرهای تعریف نشده، و همچنین دنباله کاراکتری تهی (" ") معادل با مقدار false ارزیابی می‌شوند. این بدان معنی است که سایر مقادیر به‌صورت true ارزیابی می‌شوند.

پرسش: آیا قرار دادن بلوک کد داخل ساختارهای کنترلی در درون جفت علامت { } الزامی است؟

پاسخ: در صورتی که بلوک کد مورد استفاده از ساختارهای کنترلی شامل بیش از یک دستورالعمل باشد، پاسخ مثبت است. در غیر این‌صورت نیازی به استفاده از جفت علامت { } نیست.

پرسش: آیا در درس این ساعت کلیه ساختارهای تکرار مورد بررسی قرار گرفت؟

پاسخ: در درس ساعت هفتم با عنوان " آرایه‌ها " با ساختار تکرار دیگری موسوم به Foreach آشنا خواهید شد. این ساختار جهت پردازش عناصر آرایه مورد استفاده قرار می‌گیرد.

تمرینها

هدف از این بخش دوره مطالب فراگیری شده در قالب آزمون است. پاسخ بخش آزمون بلافاصله بعد از آن آمده است. بخش فعالیتها شامل تمرینهایی است که به‌منظور افزایش قابلیت برنامه‌نویسی خواننده طراحی شده و البته فاقد پاسخ است.

آزمون

۱- با استفاده از عبارت if قطعه کدی بنویسید که دنباله کاراکتری " Youth message " را در صورتی که متغیر $age شامل عدد صحیحی بین ۱۸ و ۳۵ باشد، بر روی صفحه نمایش دهد. در صورتی که متغیر$age شامل هر مقدار دیگری باشد قطعه کد مذکور باید دنباله کاراکتری " Generic messag " را بر روی صفحه نمایش دهد.

۲- قطعه کد تمرین قبل را به‌گونه‌ای توسعه دهید که در صورتی که مقدار متغیر $age بین دو عدد صحیح 1 و 17 واقع باشد، دنباله کاراکتری " Child message " را بر روی صفحه نمایش دهد.

۳- با استفاده از یک ساختار تکرار while قطعه کدی بنویسید که اعداد فرد مابین دو عدد 1 و 49 را بر روی صفحه نمایش دهد.

۴- تمرین قبل را مجدداً تکرار کنید. این بار به‌جای استفاده از ساختار تکرار while از ساختار for استفاده کرده و نتیجه را با آنچه که قبلاً به‌دست آوردید، مقایسه نمایید.

پاسخ آزمون

-۱

```
$age = 22 ;
if ( $age > = 18 &&  $age < = 35)
   print " Youth message < BR >\ n " ;
else
   print " Generic message < BR >\ n " ;
```

-۲

```
$age = 12 ;
if ( $age > = 18 && $age < = 35)
   print " Youth message < BR >\ n " ;
elseif ( $age > = 1 &&  $age < = 17)
   print " child message < BR >\ n " ;
else
   print " Generic message < BR >\ n " ;
```

-۳

```
$num = 1 ;
while ( $num < = 49)
   {
   print " $num < BR >\ n";
   $num + = 2 ;
}
```

-۴

```
For ( $num = 1 ;  $num < = 49 ; $num + = 2)
   Print  " $num <BR >\ n " ;
```

فعالیتها

۱- گرامر مربوط به ساختارهای کنترل مورد بررسی در این درس را مجدداً بازبینی کنید. درباره اینکه این ساختارها چگونه می‌توانند به برنامه‌نویسی شما کمک کنند، فکر کنید. شاید در حال حاضر ایده پیاده سازی برخی از برنامه‌ها را با استفاده از این ساختارها داشته باشید. شاید بخواهید بر مبنای ورودی کاربر عملیات مختلفی را انجام داده و یا با استفاده از حلقه‌های تکرار یک جدول HTML را در مرورگر اینترنت به نمایش درآورید. هم‌اکنون شروع به ساخت این‌گونه ساختارهای کنترلی

کنید که در آینده از آنها استفاده خواهید کرد. با استفاده از متغیرهای موقت سعی کنید تا ورودی کاربر یا مراجعات برنامه به بانک اطلاعاتی را شبیه سازی کنید.

۲- مبحث مربوط به عملگر؟ را مجدداً مورد بررسی قرار دهید. چه تفاوتهایی بین این عملگر و ساختارهای کنترلی مشابه پیدا می‌کنید؟ کدام‌یک را مفیدتر می‌یابید؟

ساعت ششم

توابع

توابع قلب هر برنامه خوب سازماندهی شده‌ای را تشکیل می‌دهند. استفاده از توابع ضمن افزایش خوانایی برنامه تسهیلاتی را نیز از دیدگاه استفاده مجدد کد در اختیار برنامه‌نویس قرار می‌دهد. هیچ پروژه برنامه‌نویسی بزرگی را بدون وجود توابع نمی‌توان مدیریت و سازماندهی کرد.

در درس این ساعت جزئیات مربوط به توابع را مورد بررسی قرار داده و روشهایی را به نمایش خواهیم گذاشت که به‌واسطه آنها می‌توان از بسیاری دوباره کاری‌ها در برنامه به‌طور مؤثری جلوگیری به‌عمل آورد. در این ساعت به‌طور مفصل در مورد مطالب زیر به بحث خواهیم نشست:

- **چگونگی تعریف و فراخوانی توابع**
- **چگونگی ارسال مقادیر به توابع دریافت نتایج بازگشتی**
- **چگونگی فراخوانی پویای یک تابع با استفاده از دنباله کاراکتری ذخیره‌شده در یک متغیر**
- **چگونگی دستیابی به متغیرهای سراسری از درون یک تابع**
- **چگونگی تخصیص حافظه به یک تابع**
- **چگونگی ارسال داده‌ها به تابع از طریق مرجع**
- **چگونگی ایجاد توابع بی‌نام**
- **چگونگی بررسی وجود یک تابع پیش از اقدام به فراخوانی آن**

در ادامه به بررسی هر یک از این موارد می‌پردازیم.

مفهوم تابع

تابع را می‌توان به منزله یک ماشین تصور کرد. هر ماشینی قادر است مواد خام داده شده به آن را بگیرد و به منظور خاصی مثلاً تولید یک محصول، عملیات یا فرآیندهایی را بر روی آن به اجرا درآورد. به‌طور مشابه، تابع نیز داده‌ها و مقادیری را از طریق ورودی دریافت کرده و پس از انجام پردازش بر روی آن فرآیندی را به‌عنوان نتیجه انجام داده (مثل چاپ نتایج بر روی صفحه مرورگر اینترنت) یا مقداری را به‌عنوان نتیجه باز می‌گرداند (توابع ممکن است هر دو عمل فوق را به‌طور توأم انجام دهند).

چنان‌چه قصد پختن تنها یک کیک را داشته باشید، به‌احتمال قوی خودتان اقدام به این کار می‌کنید. اما در صورتی که قصد آماده‌کردن هزاران کیک را داشته باشید، به فکر تهیه یک ماشین کیک‌پزی خواهید افتاد. به‌طور مشابه هنگامی که تصمیم به ایجاد تابعی می‌گیرید مهمترین عاملی که به احتمال زیاد شما را وادار به چنین تصمیمی می‌کند جلوگیری از انجام عملیات تکراری یا به عبارت بهتر جلوگیری از دوباره کاری است.

تابع بلوک مستقلی از کد است که می‌توان آن‌را در برنامه‌های اسکریپت فراخوانی کرد. هنگامی که تابع فراخوانی می‌شود کد درون آن تابع به اجرا در می‌آید. می‌توان مقادیری را به‌عنوان ورودی به تابع ارسال کرد. توابع می‌توانند بر روی مقادیر دریافتی پردازش و عملیات ویژه‌ای را که طراح آنها، مشخص کرده، انجام دهند. پس از اتمام عملیات پردازش، تابع مقداری را به عنوان نتیجه به کدی که آن‌را فراخوانی کرده باز می‌گرداند.

فراخوانی توابع

عموماً توابع را می‌توان به دو دسته بزرگ تقسیم کرد: توابع سیستمی (توابعی که از پیش در PHP وجود دارند) و توابعی که برنامه‌نویس اقدام به تعریف آنها می‌کند. PHP4 صدها تابع سیستمی دارد. اگر به‌خاطر داشته باشید اولین برنامه‌ای که در کتاب نوشتیم شامل فراخوانی تابعی به صورت زیر بود:

```
Print (" Hello Web ") ;
```

در آن برنامه، تابع ( ) print را فراخوانی کرده و دنباله کاراکتری " Hello Web " را جهت پردازش به آن ارسال نمودیم. فرآیند پردازش این تابع به‌سادگی نمایش مقدار ورودی بر روی صفحه است. فراخوانی تابع چیزی نیست جز ذکر نام آن (در این‌جا print) و به‌دنبال آن یک جفت پرانتز. اگر نیاز به ارسال اطلاعاتی به تابع باشد، می‌توان این اطلاعات را در درون پرانتز فوق ذکر نمود. اطلاعاتی که بدین طریق به تابع ارسال می‌شود، آرگومان نام دارد. برخی از توابع جهت عملیات خود به بیش از

یک آرگومان نیاز دارند. در این صورت لازم است، تا اسامی آرگومان‌ها را با استفاده از علامت کاما از یکدیگر جدا کنیم. شکل عمومی زیر بیانگر همین واقعیت است :

```
Some_ function ( $an_ argument, $another _ argument) ;
```

تابع ( ) print نمونه‌ای از آن دسته از توابع است که مقداری را به‌عنوان نتیجه باز می‌گردانند. بیشتر توابع چنین هستند، یعنی به محض کامل شدن عملیات، اطلاعاتی را دراختیارمان قرار می‌دهند که گویای موفقیت یا عدم موفقیت در انجام عملیات می‌باشد. تابع ( ) print مقداری از نوع boolean را به تابع فراخوان باز می‌گرداند.

تابع ( ) print از دیدگاه فراخوانی یک استثنا محسوب می‌شود، بدین معنی که فراخوانی این تابع نیازی به استفاده از جفت پرانتز ندارد. دو فراخوانی زیر را در نظر بگیرید:

```
print ( " Hello Web " ) ;
print " Hello Web" ;
```

تأثیرفراخوانی هر دو تابع فوق یکسان بوده و موجب چاپ دنباله کاراکتری "Hello Web" بر روی صفحه می‌شود. قرار دادن آرگومان یا آرگومان‌های تمامی توابع هنگام فراخوانی در درون جفت پرانتز الزامی است و تنها تابع ( ) print در این میان از این قاعده مستثنی است.

برای مثال تابع ( ) abs یک مقدار عددی را به عنوان آرگومان پذیرفته و قدر مطلق آن را به‌عنوان حاصل عملیات باز می‌گرداند. برنامه لیست ۱- ۶ این عملیات را نشان می‌دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 6.1</title>
 4: </head>
 5: <body>
 6: <?php
 7: $num = -321;
 8: $newnum = abs( $num );
 9: print $newnum;
10: // prints "321"
11: ?>
12: </body>
13: </html>
```

لیست ۱- ۶ استفاده از تابع سیستمی ( ) abs

همان‌گونه که مشاهده می‌کنید ابتدا مقدار عددی 321 - به متغیر $num منسوب شده است. سپس این متغیر به‌عنوان آرگومان تابع ( ) abs به این تابع ارسال شده است. تابع نامبرده محاسبات لازم را بر روی این مقدار انجام داده و نتیجه را باز گردانده است. در خط ۸ از برنامه نتیجه مذکور به متغیر جدیدی با نام $newnum منسوب شده و در خط بعدی این مقدار با استفاده از تابع سیستمی

print بر روی صفحه به نمایش درآمده است. در حقیقت می‌توانستیم از به‌کارگیری متغیر جدید $newnum نیز صرف‌نظر کنیم، بدین ترتیب که مقدار عددی مورد بررسی را به تابع ( ) abs ارسال کرده و سپس خود تابع مذکور را در تابع سیستمی ( ) print به‌عنوان آرگومان مورد بهره‌برداری قرار دهیم. فراخوانی زیر این وضعیت را نشان می‌دهد:

```
print ( abs ( - 321)) ;
```

با این همه در این برنامه ترجیح دادیم تا به‌دلیل روشن‌تر شدن عملیات و افزایش وضوح و خوانایی برنامه از متغیرهای موقت $num و $newnum استفاده کنیم. گاهی اوقات می‌توان با تقسیم یک برنامه به قطعات کوچک‌تر یا چندین عبارت ساده به میزان خوانایی آن اضافه نمود.

استفاده از توابع غیر سیستمی (توابع تعریف شده توسط برنامه نویس) نیز به سادگی فراخوانی توابع سیستمی امکان‌پذیر می‌باشد.

تعریف تابع

توابع را می‌توان با استفاده از عبارت Function در زبان PHP تعریف کرد. شکل عمومی تعریف یک تابع با دو آرگومان فرضی به صورت زیر است:

```
Function some _ function ($argument 1, $argument 2) {
   // function code here
}
```

در این تعریف نام تابع به‌دنبال واژه function آمده و بعد از آن نیز جفت پرانتزی که شامل آرگومان‌های تابع است، ذکر می‌شوند. در صورتی که تابع مورد نظر نیازمند بیش از یک آرگومان باشد، لازم است تا اسامی آنها با استفاده از علامت کاما از یکدیگر جدا شوند. مقادیر آرگومان‌ها هنگام فراخوانی تابع مشخص شده و از این نظر تعداد مقادیر ارسالی به تابع باید با تعداد آرگومان‌ها برابر باشد. چنان‌چه تابع فاقد آرگومان باشد، باز هم استفاده از جفت پرانتز در جلوی نام تابع هنگام تعریف آن ضروری است.

برنامه موجود در لیست ۲- ۶ چگونگی تعریف یک تابع و فراخوانی آن‌را نشان می‌دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 6.2</title>
 4: </head>
 5: <body>
 6: <?php
 7: function bighello() {
 8:      print "<h1>HELLO!</h1>";
 9: }
10: bighello();
11: ?>
12: </body>
13: </html>
```

لیست ۲- ۶ تعریف یک تابع

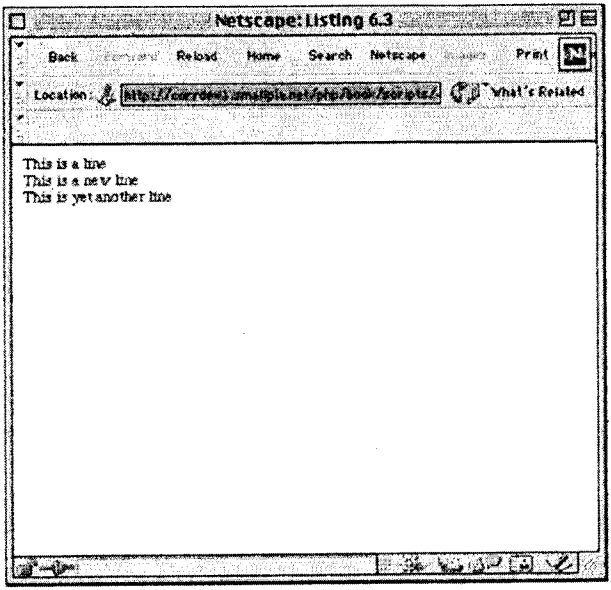
برنامه موجود در این لیست به‌سادگی دنباله کاراکتری " HELLO " را در قالب المان ‹ H1 › زبان نشانه‌گذاری HTML نمایش می‌دهد. تابع ( ) bighello جهت اجرا به هیچ آرگومانی نیاز ندارد. از این‌رو داخل جفت پرانتز هیچ نوع اطلاعاتی مشاهده نمی‌کنید. این تابع به‌خوبی کار می‌کند ولی نکته اینجاست که آنچنان کاربرد مفیدی ندارد. برنامه موجود در لیست ۳- ۶ تابعی تعریف می‌کند که مستلزم دریافت یک آرگومان است. این تابع نسبت به تابع برنامه قبل عملکرد مفیدتری را به نمایش می‌گذارد.

```
 1: <html>
 2: <head>
 3: <title>Listing 6.3</title>
 4: </head>
 5: <body>
 6: <?php
 7: function printBR( $txt ) {
 8:      print ("$txt<br>\n");
 9: }
10: printBR("This is a line");
11: printBR("This is a new line");
12: printBR("This is yet another line");
13: ?>
14: </body>
15: </html>
```

لیست ۳- ۶ تعریف تابعی که جهت اجرا به یک آرگومان نیاز دارد

خروجی حاصل از اجرای این برنامه در شکل ۱- ۶ به نمایش گذاشته شده‌است. همان‌گونه که مشاهده می‌کنید، در خط ۷ از برنامه تابع ( ) printBR به‌گونه‌ای تعریف شده که انتظار دریافت آرگومانی (از نوع دنباله کاراکتری) را دارد. از این‌رو ما نام $txt را به‌عنوان این آرگومان انتخاب کرده و در تعریف تابع ( ) printBR آن‌را در درون جفت پرانتز قرار داده‌ایم. بدین ترتیب هر آنچه که هنگام فراخوانی به تابع مذکور ارسال شود در متغیر $txt ذخیره خواهد شد. در داخل بدنه تابع مذکور در خط

۸ مقدار این متغیر چاپ شده و المان ‹ br › و به‌دنبال آن نیز کاراکتر خط جدید (جهت درج یک خط جدید در خروجی) به انتهای آن اضافه شده است.



شکل ۱- ۶ نمایش خروجی حاصل از اجرای تابع ( ) printBR

اکنون با در دست داشتن تعریف یک چنین تابعی هنگام نیاز به چاپ یک خط خروجی بر روی مرورگر اینترنت (مانند خطوط ۱۰، ۱۱ و ۱۲ در این برنامه)، کافی است تا به‌جای تابع سیستمی print ( ) از تابع جدید ( ) printBR استفاده کنیم، چرا که این تابع ما را از زحمت تایپ تکراری المان ‹ br › معاف خواهد کرد.

بازگشت نتایج از توابع تعریف شده توسط برنامه‌نویس PHP

در برنامه نمونه قبل با استفاده از تابع ( ) printBR توانستیم یک رشته کاراکتری را به‌طور مکرر بر روی پنجره مرورگر اینترنت نمایش دهیم. گاهی اوقات لازم است تا به منظور خاصی خروجی حاصل از عملیات تابع را جهت پردازش بیشتر از تابع دریافت کنیم. اگر تابع مورد استفاده‌تان تابعی است که دنباله کاراکتری دریافتی به‌عنوان آرگومان را به نوعی تغییر شکل می‌دهد، شاید علاقمند باشید که خروجی آن‌را جهت پردازش بیشتر در قالب آرگومان به تابع دیگری ارسال نمایید. جهت بازگشت نتایج از توابع از عبارت return به‌همراه مقدار بازگشتی مورد نظر استفاده می‌کنیم. عبارت return موجب توقف اجرای تابع شده و مقدار بازگشتی را به برنامه فراخواننده تابع باز می‌گرداند.

برنامه لیست ۴- ۶ شامل تابعی است که مجموع دو عدد را پس از محاسبه باز می‌گرداند.

```
 1: <html>
 2: <head>
 3: <title>Listing 6.4</title>
 4: </head>
 5: <body>
 6: <?php
 7: function addNums( $firstnum, $secondnum ) {
 8:      $result = $firstnum + $secondnum;
 9:      return $result;
10: }
11: print addNums(3,5);
12: // will print "8"
13: ?>
14: </body>
15: </html>
```

لیست ۴- ۶ تابعی که مقداری را به برنامه فراخواننده باز می‌گرداند

خروجی حاصل از اجرای برنامه لیست ۴- ۶ نمایش عدد ۸ در خروجی است. توجه کنید که در خط ۷ از این برنامه تابع ( ) addNums باید با دو آرگومان عددی فراخوانی شود (در خط ۱۱ دو عدد ۳ و ۵ به همین منظور در فراخوانی تابع نامبرده مورد استفاده قرار گرفته‌اند). این دو عدد به ترتیب در متغیرهای $firstnum و $secondnum ذخیره می‌شوند. مطابق انتظار، تابع ( ) addNums این دو عدد را با یکدیگر جمع کرده و نتیجه حاصل از این عملیات را در متغیر دیگری با نام $result ذخیره می‌کند.

در مجموع عبارت return می‌تواند تنها یک مقدار را به برنامه فراخواننده بازگرداند (همچنین ممکن است این عبارت هیچ مقداری را باز نگرداند). عبارت return بسیار با انعطاف است به‌گونه‌ای که قادر است به روشهای مختلفی نتیجه حاصل از عملیات را بازگرداند. ساده‌ترین روشها این است که به سادگی نتیجه نهایی محاسبات تابع را به‌صورت زیر بازگرداند:

```
return 4 ;
```

در روش دیگر return می‌تواند حاصل یک عبارت دیگر را به‌صورت زیر بازگرداند:

```
return ( $a / $b) ;
```

همچنین عبارت return می‌تواند نتیجه حاصل از فراخوانی تابع دیگری را به‌صورت زیر باز گرداند:

```
return ( another _function ( $an _ argument) ) ;
```

فراخوانی توابع به‌صورت پویا

در زبان PHP می‌توان نام تابعی را در قالب یک دنباله کاراکتری به متغیری منسوب کرده و سپس از این متغیر دقیقاً به جای نام تابع جهت فراخوانی بهره گرفت. برنامه موجود در لیست ۵- ۶ مثال ساده‌ای را جهت نمایش این قابلیت زبان PHP به‌خوبی نشان می‌دهد.

```
<html>
```

```
<head>
<title>Listing 6.5</title>
</head>
<body>
?>php
function sayHello(){
print "hello<br>";
}
$ function_holder = "sayHello"';
$function_holder();
?>
</body>
</html>
```

لیست ۵- ۶ فراخوانی پویای یک تابع

همان‌گونه که مشاهده می‌کنید در خط ۱۰ از این برنامه دنباله کاراکتری معادل با نام تابع ( ) sayHello یعنی " sayHello " به متغیر $function _ older منسوب شده است. با این عمل می‌توان جهت فراخوانی تابع ()SayHello نام این متغیر را به‌همراه جفت پرانتزی در جلوی آن مورد استفاده قرار دارد. خط ۱۱ از این برنامه به‌وضوح چنین فرآیندی را نشان می‌دهد.

اماواقعاًانجام چنین کاری چه فایده‌ای در پی دارد؟ در این مثال، همان‌گونه که مشاهده می‌کنید ظاهراً کار اضافه‌ای را جهت نسبت دهی دنباله کاراکتری " sayHello " به متغیر $function _ holder انجام داده‌ایم. فراخوانی پویای یک تابع هنگامی مفید است که بخواهیم بر مبنای شرایط گوناگون روند اجرای برنامه را دستخوش تغییر نماییم. برای مثال ممکن است بخواهیم بر مبنای مقداری از یک پارامتر موجود در رشته جستجوی یک آدرس URL برنامه ما رفتارهای متفاوتی از خود نشان دهد. برای تحصیل این هدف می‌توانیم مقدار پارامتر مورد نظرمان را از رشته جستجو استخراج کرده و بر مبنای آن یکی از چند تابع موجود را فراخوانی کنیم.

برخی از توابع سیستمی PHP نیز از این ویژگی بهره‌برداری می‌کنند. برای مثال تابع سیستمی ( ) array _ walk از یک دنباله کاراکتری جهت فراخوانی تابعی به‌ازای هر یک از عناصر موجود در آرایه استفاده می‌کند. بعداً در درس مربوطه جزئیات مربوط به این تابع و مثالی را در رابطه با آن بررسی می‌کنیم.

حوزه تعریف متغیرها در یک برنامه PHP

متغیرهای معرفی شده در داخل یک تابع حکم متغیرهای محلی را برای آن تابع دارند. به‌عبارت دیگر از این‌گونه متغیرها نمی‌توان در خارج از تابع مورد نظر یا در سایر توابع استفاده کرد. در مورد پروژه‌های برنامه‌نویسی بزرگ‌تر که معمولاً از تعداد زیادی تابع استفاده می‌شود، این وضعیت از رونویسی مقادیر متغیرهای همنام که در توابع جداگانه تعریف شده‌اند، جلوگیری به‌عمل آورده و امکان خطا را کاهش می‌دهد.

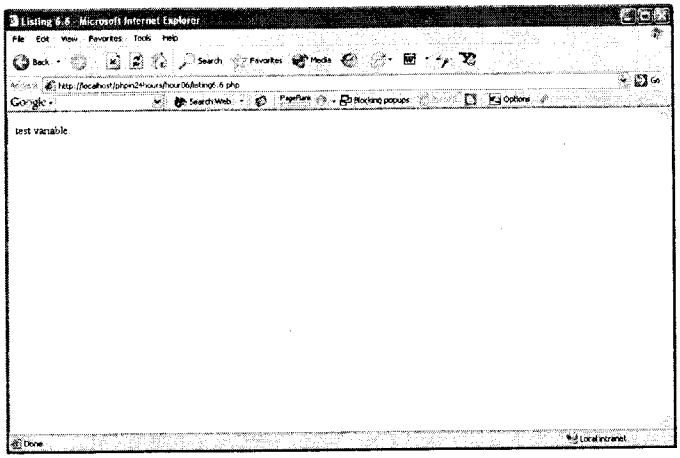
برنامه موجود در لیست ۶- ۶ متغیری را در درون یک تابع معرفی کرده و سپس سعی می‌کند تا مقدار آن متغیر را در خارج از تابع جهت نمایش مورد دستیابی قرار دهد.

```
<html>
<head>
<title>Listing 6.6</title>
</head>
<body>
<?php
function test () }
$ testvariable = "this is a test variable";
}
print "test variable: $testvariable<br>";
?>
</body>
</html>
```

لیست ۶- ۶ نمایشی از قانون حوزه متغیرها: متغیر معرفی شده در درون یک تابع در خارج از آن تابع غیر قابل دستیابی است.

خروجی حاصل از این برنامه را می‌توانید در شکل ۲- ۶ مشاهده کنید. همان‌گونه که از این شکل نیز بر می‌آید مقدار متغیر $testvariable هرگز مورد دستیابی و نمایش قرار نگرفته است. این بدان جهت است که متغیر نامبرده در خارج از تابع ( ) test موجود نمی‌باشد. توجه کنید که تلاش جهت دستیابی به یک چنین متغیری (خط ۱۰ از برنامه) هرگز موجب بروز خطا نمی‌شود.

به‌طور مشابه متغیرهای معرفی شده در خارج از تابع را به خودی خود نمی‌توان در درون آن تابع مورد دستیابی و استفاده قرار داد.



شکل ۲- ۶ خروجی حاصل از برنامه لیست ۶- ۶

دستیابی به متغیرها با استفاده از واژه کلیدی global

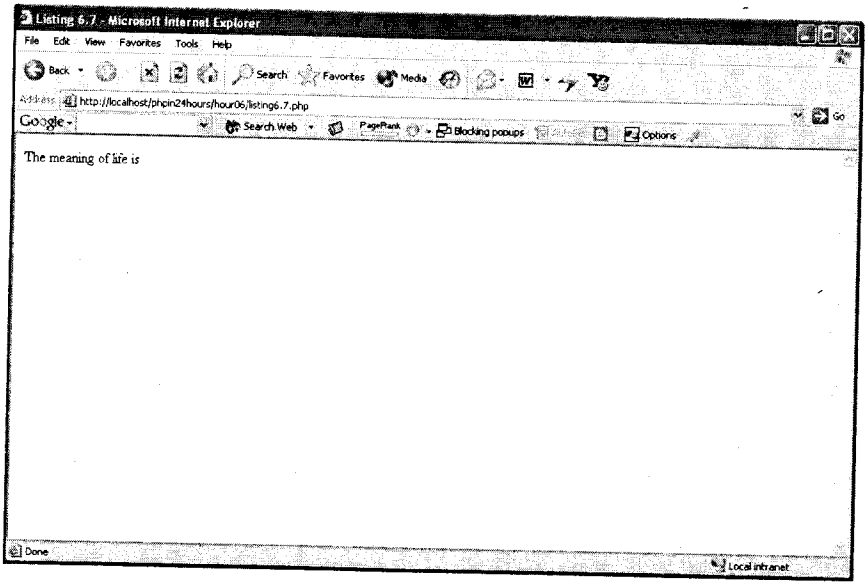
به‌طور پیش‌فرض، متغیرهای معرفی شده در خارج از یک تابع را نمی‌توان در درون آن تابع مورد استفاده و دستیابی قرار داد. این بدان معنی است که در صورت استفاده از یک چنین متغیری در درون تابع موتور PHP آن متغیر را به عنوان موجودیتی مستقل از متغیر همنامی که خارج از تابع معرفی شده است، فرض می‌کند. برنامه موجود در لیست ۷- ۶ این وضعیت را به‌خوبی نشان داده است.

```
<html>
<head>
<title>Listing 6.7</title>
</head>
<body>
<?php
$life = 42;
()}function meaningOfLife
print "The meaning of life is $life<br>";
}
() ;meaningOfLife
?>
</body>
</html>
```

لیست ۷- ۶ متغیرهای تعریف شده در خارج از تابع را به‌طور پیش‌فرض نمی‌توان در درون آن تابع مورد استفاده و دستیابی قرار داد

خروجی حاصل از اجرای این برنامه در شکل ۳- ۶ به نمایش درآمده است. همان‌گونه که ممکن است متوجه شده‌باشید، تابع ( ) meaningOFLIFE هیچ‌گونه دسترسی به متغیر $life که در

خط ۷ از برنامه و خارج از تابع مزبور معرفی شده‌است، ندارد. چنین به‌نظر می‌رسد که محتوای متغیر $life هنگام تلاش تابع فوق جهت دستیابی به آن تهی باشد. در مجموع چنین وضعیتی مناسب است چراکه احتمال رونویسی مقادیر متغیرهای همنام تعریف شده در درون تابع و خارج آن وجود ندارد. وانگهی در صورت نیاز تابع به متغیرهای تعریف شده در خارج از آن می‌توان آنها را به‌عنوان آرگومان‌های تابع به آن ارسال نمود.



شکل ۳- ۶ خروجی حاصل از اجرای برنامه لیست ۷- ۶

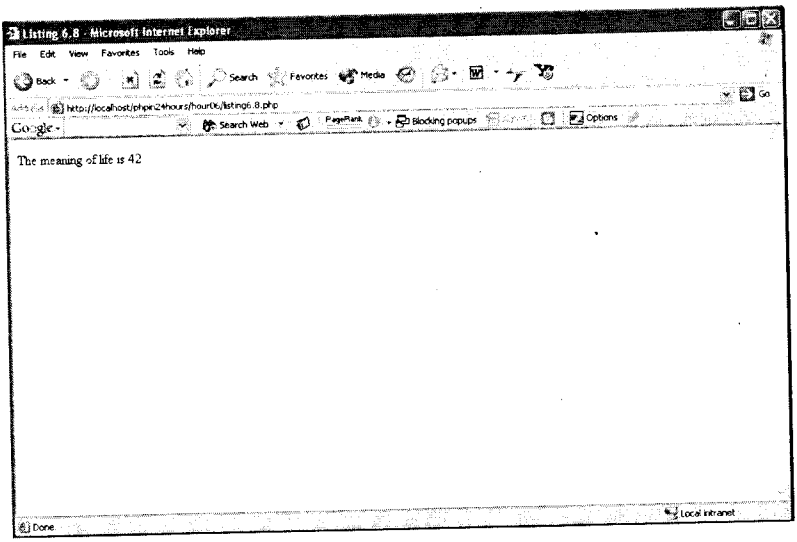
با این وجود گاهی لازم است تا به یک متغیر سراسری مهم که در خارج از تابع تعریف شده است بدون ارسال آن به‌عنوان آرگومان، در درون تابع دسترسی داشته باشیم. در چنین موقعی عبارت global مورد استفاده قرار می‌گیرد. برنامه موجود در لیست ۸- ۶ چگونگی استفاده از آن‌را نشان می‌دهد.

```
<html>
<head>
<title>Listing 6.8</title>
</head>
<body>
<?php
$life=42;
function meaningOfLife()}
global $life;
print "The meaning of life is $life<br>";
```

```
}
MeaningOfLife();
?>
</body>
</html>
```

لیست ۸- ۶ دستیابی به یک متغیر سراسری با استفاده از عبارت global

خروجی حاصل از اجرای این برنامه در شکل ۴- ۶ قابل مشاهده و بررسی است. با قرار دادن واژه global پیش از نام متغیر $life هنگام معرفی آن در درون تابع ( ) meaningOFLIFE (خط ۹ از برنامه) می‌توان ترتیبی داد که موتور PHP متغیر مذکور را همان متغیر $life معرفی شده در خارج از تابع فرض کند.



شکل ۴- ۶ نتیجه حاصل از به‌کارگیری واژه global جهت دستیابی به متغیر سراسری (لیست ۸- ۶)

هر تابعی که مایل به دستیابی به متغیرهای سراسری باشد، باید آن متغیرها را با استفاده از واژه global تعریف کند. با این همه لازم است تا هنگام انجام این کار دقت زیادی به خرج دهید، چراکه با تغییر مقدار متغیرهای این‌چنین، مقدار آنها در سرتاسر برنامه دستخوش تغییر خواهندشد.

با استفاده از واژه global این امکان وجود دارد که چندین متغیر را به‌طور توأم به‌صورت سراسری معرفی کنیم، تنها کافی است تا اسامی این متغیرها را بعد از واژه global ذکر کرده و با استفاده از علامت کاما آنها را از یکدیگر جدا کنیم. عبارت زیر نمونه‌ای از معرفی سه متغیر سراسری است:

```
global $var 1, $var 2, $var 3 ;
```

در درس ساعت نهم با عنوان "بهره‌گیری از فرمها" آرایه ویژه‌ای با نام $GLOBALS را معرفی خواهیم کرد که به کمک آن می‌توانیم از هرجای برنامه متغیرهای سراسری را مورد دستیابی قرار دهیم.

معمولاً آرگومان تابع یک کپی از آن متغیری است که هنگام فراخوانی به تابع ارسال می‌شود. از این جهت اعمال هرگونه تغییر به مقدار آرگومان در درون تابع تاثیری بر اصل متغیر که جایی خارج از آن معرفی شده است، ندارد. این در حالی‌است که تغییر متغیرهای سراسری در درون تابع موجب می‌شود که اصل متغیر که آن هم خارج از تابع معرفی شده است، دستخوش تغییر شود. از این‌رو هنگام کار با متغیرهای سراسری لازم است تا برنامه‌نویس دقت بیشتری به خرج دهد.

ثبت حالات وضعیتهای مابین فراخوانی توابع با استفاده از واژه کلیدیstatic

متغیرهای موجود در درون تابع از طول عمر کوتاهی به اندازه طول عمر خود تابع برخوردارند. طول زندگی این‌گونه متغیرها از زمان فراخوانی تابع آغاز شده و هنگامی که اجرای تابع به انتهای خود می‌رسد، پایان می‌یابند. بار دیگر ذکر می‌کنیم که این وضعیت طبیعی مناسب است. معمولاً بهترین روش برای ایجاد یک برنامه اسکریپت این است که برنامه را به‌عنوان یکسری بلوک کدهای مستقل طراحی کنیم به‌گونه‌ای که هر بلوک تا جای ممکن اطلاعات کمی از بلوک‌های دیگر داشته باشد. با این همه در برخی موارد لازم است تا حافظه‌ای هرچند کوچک در اختیار توابع خود قرار دهیم.

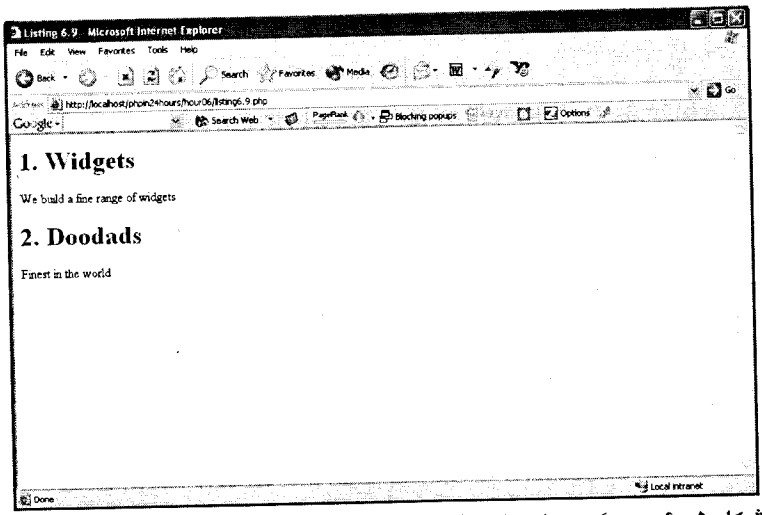
فرض کنید بخواهیم تابعی طراحی کنیم که از تعداد دفعاتی که آن‌را در طول اجرای برنامه فراخوانی کرده‌ایم، مطلع باشد. اما چنین تابعی چه استفاده‌ای می‌تواند داشته باشد؟ یکی از استفاده‌ها می‌تواند در رابطه با ایجاد اسناد online به‌صورت پویا باشد. برای مثال ایجاد تیترهای شماره‌گذاری شده از این دسته اسناد هستند.

در نگاه اول، بهره‌گیری از عبارتی که در قسمت قبل راجع به آن صحبت کردیم، یعنی عبارت global می‌تواند چنین امکانی را در اختیارمان قرار دهد. برنامه لیست ۹- ۶ تلاشی را جهت انجام این کار نشان می‌دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 6.9</title>
 4: </head>
 5: <body>
 6: <?php
 7: $num_of_calls = 0;
 8: function numberedHeading( $txt ) {
 9:     global $num_of_calls;
10:     $num_of_calls++;
11:     print "<h1>$num_of_calls. $txt</h1>";
12: }
13: numberedHeading("Widgets");
14: print("We build a fine range of widgets<p>");
15: numberedHeading("Doodads");
16: print("Finest in the world<p>");
17: ?>
18: </body>
19: </html>
```

لیست ۹- ۶ بهره‌گیری از عبارت global جهت ذخیره مقدارمتغیرمابین فراخوانی‌های مختلف تابع

همان‌گونه که مشاهده می‌کنید کار به‌درستی انجام شده است. در خط ۷ از برنامه متغیری با نام $num _ of _ calls در خارج از تابع ( ) numbered Heading معرفی شده است و با استفاده از واژه global در خط ۹ از برنامه، این متغیر در دسترس تابع مذکور قرار گرفته است. شکل ۵- ۶ خروجی حاصل از اجرای این برنامه را بر روی صفحه مرورگر اینترنت نشان می‌دهد.



شکل ۵- ۶ بهره‌گیری از واژه global جهت ثبت تعداد دفعات فراخوانی تابع

هر بار که تابع ( ) numberedHeading فراخوانی می‌شود، مقدار متغیر $num _ of _ calls در خط ۱۰ برنامه یک واحد افزایش می‌یابد. بدین ترتیب می‌توانیم تیتر مورد نظرمان را به‌همراه شماره

مربوطه که بیانگر تعداد دفعات فراخوانی تابع است، بر روی صفحه نمایش دهیم.

با اینکه به جواب رسیدیم اما روش استفاده از متغیرهای سراسری را برای حل مسائلی از این دست توصیه نمی‌کنیم. توابعی که از متغیرهای سراسری بهره می‌برند به عنوان بلوک کدهای مستقل در نظر گرفته نمی‌شوند، چراکه هنگام استفاده مجدد از آنها همواره مراجعه به متغیرهای سراسری مورد استفاده در آن توابع امری ضروری است.

در مقابل روش فوق، عبارت دیگری با عنوان static روش مفیدتری را پیش روی ما قرار می‌دهد. متغیرهایی که در درون تابع با استفاده از واژه static معرفی می‌شوند، حکم متغیرهای محلی را برای آن تابع دارند، با این ویژگی مهم که مقدار آن متغیر از فراخوانی به فراخوانی دیگر حفظ خواهد شد. برنامه لیست ۱۰-۶روش‌دیگری‌برای پیاده سازی برنامه لیست ۹- ۶ را با استفاده از عبارت static نشان‌می‌دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 6.10</title>
 4: </head>
 5: <body>
 6: <?php
 7: function numberedHeading( $txt ) {
 8:       static $num_of_calls = 0;
 9:       $num_of_calls++;
10:       print "<h1>$num_of_calls. $txt</h1>";
11: }
12: numberedHeading("Widgets");
13: print("We build a fine range of widgets<p>");
14: numberedHeading("Doodads");
15: print("Finest in the world<p>");
16: ?>
17: </body>
18: </html>
```

لیست ۱۰- ۶ بهره‌گیری‌ازعبارت static جهت ثبت مقدارمتغیرمابین فراخوانی‌های مختلف یک تابع

همان‌گونه که مشاهده می‌کنید هم‌اکنون تابع numberedHeading ( ) کاملاً مستقل است. در خط ۸ از برنامه، متغیر $num _ of _calls معرفی و مقداردهی شده است. در حقیقت این مقداردهی زمانی صورت می‌گیرد که تابع مذکور در خط ۱۲ برنامه برای اولین بار فراخوانی می‌شود. با فراخوانی تابع فوق برای دومین بار در خط ۱۴ مقدار قبلی متغیر $num _ of _ calls نادیده گرفته می‌شود و این در حالی‌است که مقدار قبلی این متغیر هنوز در جایی ثبت شده و قابل استفاده است. بدین ترتیب می‌توان بدون نگرانی تابع numberedHeading ( ) را در سایر برنامه‌های اسکریپت نیز مورد استفاده قرار داد چراکه این تابع مستقل از متغیرهای سراسری عمل می‌کند. با اینکه خروجی حاصل از برنامه این لیست دقیقاً مشابه برنامه لیست قبل است، اما روش اخیر به‌دلیل استقلال و قابلیت استفاده مجدد به سایر برنامه‌ها ارجحیت دارد.

چند نکته مهم در مورد آرگومان‌های تابع

تاکنون چگونگی ارسال متغیرها را به تابع مورد بررسی قرار دادیم اما مطالب بیشتری برای بحث و گفتگو در این رابطه وجود دارد. در این قسمت قصد داریم روشی را جهت استفاده از مقدار پیش‌فرض آرگومان یک تابع و همچنین روشی را نیز جهت ارسال متغیرها به تابع به شیوه ارسال از طریق مرجع (به‌جای شیوه ارسال از طریق مقدار که تا کنون از آن استفاده می‌کردیم) ارائه دهیم. بدین ترتیب می‌توانیم اصل مقدار را به جای کپی آن در اختیار تابع مورد نظرمان قرار دهیم.

تنظیم مقادیر پیش فرض برای آرگومان‌ها

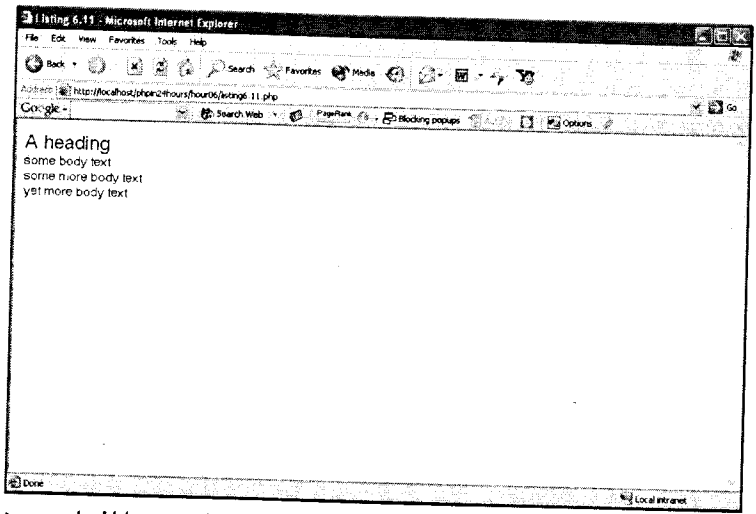
زبان PHP امکانات بسیار مناسبی را جهت ایجاد توابع پویا و منعطف در اختیار برنامه‌نویس قرار می‌دهد. تا بدین جا چنین اظهار کردیم که برخی از توابع جهت انجام عملیات مورد نظرشان نیاز به یک یا چند آرگومان دارند. با تعریف برخی از آرگومان‌ها به‌عنوان آرگومان‌های اختیاری می‌توان دست خود را در بهره‌گیری از آن توابع بیش از پیش بازگذاشت. برنامه موجود در لیست ۱۱- ۶ تابع کوچک اما مفیدی را معرفی می‌کند که یک دنباله کاراکتری را در قالب المان font زبان نشانه گذاری HTML مورد استفاده قرار می‌دهد. قصد ما این است که امکان تغییر اندازه فونت خروجی برنامه را از طریق تغییر صفت size این المان در اختیار قرار دهیم. بنابراین باید به نحوی آرگومان $size را علاوه بر دنباله کاراکتری خروجی در اختیار تابع قرار دهیم.

```
1: <html>
2: <head>
3: <title>Listing 6.11</title>
4: </head>
5: <body>
6: <?php
7: function fontWrap( $txt, $size ) {
8:     print "<font size=\"$size\"
9:         face=\"Helvetica,Arial,Sans-Serif\">
10:        $txt</font>";
11: }
12: fontWrap("A heading<br>",5);
13: fontWrap("some body text<br>",3);
14: fontWrap("some more body text<BR>",3);
15: fontWrap("yet more body text<BR>",3);
16: ?>
17: </body>
18: </html>
```

لیست ۱۱- ۶ تابعی با دو آرگومان

خروجی حاصل از اجرای این برنامه در شکل ۶- ۶ قابل بررسی است. با اینکه تابع فوق بسیار مفید به نظر می‌رسد، اما حقیقت این است که فرآیند تغییر اندازه فونت خروجی عمل بسیار متداولی نیست. در بیشتر مواقع این اندازه برابر با مقدار 3 تنظیم می‌شود. با منسوب کردن مقداری به آرگومان تابع هنگام تعریف آن می‌توانیم ترتیبی دهیم تا ذکر مقدار آرگومان $size هنگام فراخوانی تابع

( ) fontWrap به امری اختیاری مبدل شود. در این صورت هنگام فراخوانی تابع مذکور چنان‌چه برنامه‌نویس مقداری را برای این آرگومان مشخص نکند از مقدار پیش‌فرض آن استفاده خواهد شد. برنامه لیست ۱۲- ۶ با انجام چنین عملی آرگومان $size را به آرگومانی اختیاری تبدیل کرده‌است.



شکل ۶- ۶ تابعی که قالب‌بندی و نمایش دنباله‌های کاراکتری را انجام می‌دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 6.12</title>
 4: </head>
 5: <body>
 6: <?php
 7: function fontWrap( $txt, $size=3 ) {
 8:     print "<font size=\"$size\"
 9:         face=\"Helvetica,Arial,Sans-Serif\">
10:         $txt</font>";
11: }
12: fontWrap("A heading<br>",5);
13: fontWrap("some body text<br>");
14: fontWrap("some more body text<br>");
15: fontWrap("yet more body text<br>");
16: ?>
17: </body>
18: </html>
```

لیست ۱۲- ۶ تابعی با یک آرگومان اختیاری

همان‌گونه که مشاهده می‌کنید، هنگامی که تابع ( ) fontWrap در خط ۱۲ برنامه با ذکر هر دو آرگومان فراخوانی می‌شود، اندازه فونت خروجی با مقدار عددی 5 تنظیم می‌گردد. چنان‌چه ملاحظه می‌کنید، در خطوط ۱۳، ۱۴ و ۱۵ از برنامه هنگام فراخوانی تابع مذکور آرگومان دوم را که یک آرگومان اختیاری است، حذف کرده‌ایم. این عمل باعث می‌شود تا مقدار پیش‌فرض 3 به‌عنوان اندازه

فونت خروجی مورد استفاده قرار بگیرد. تعداد آرگومان‌های اختیاری یک تابع مشمول هیچ‌گونه محدودیتی نیست اما توجه به این نکته ضروری است که با تعیین آرگومانی از یک تابع به‌صورت یک آرگومان اختیاری، تمامی آرگومان‌های بعد از آن نیز باید از نوع اختیاری (با مقدار پیش‌فرض) تعریف شوند.

ارسال مقادیر به تابع (در قالب آرگومان) از طریق مرجع

هنگام ارسال مقادیر به تابع، کپی آن مقادیر در آرگومان‌های تابع موردنظر ذخیره می‌شوند. بدین ترتیب تابع مذکور به‌جای خود مقادیر به کپی آنها دسترسی خواهد داشت. این بدان معنی است که هرگونه تغییر در این متغیرها هیچ‌گونه تأثیری بر خود متغیرها ندارد. این وضعیت به‌خوبی در برنامه لیست ۱۳- ۶ دیده می‌شود.

```
 1: <html>
 2: <head>
 3: <title>Listing 6.13</title>
 4: </head>
 5: <body>
 6: <?php
 7: function addFive( $num ) {
 8:       $num += 5;
 9: }
10: $orignum = 10;
11: addFive( $orignum );
12: print( $orignum );
13: ?>
14: </body>
15: </html>
```

لیست ۱۳- ۶ ارسال آرگومان به تابع از طریق مقدار

همان‌گونه که مشاهده می‌کنید تابع ( ) addFive یک متغیر عددی را به‌عنوان آرگومان پذیرفته و عدد 5 را به مقدار آن متغیر اضافه می‌کند. این تابع از آن دسته توابعی است که مقداری را به برنامه فراخواننده باز نمی‌گرداند. در خط ۱۰ از برنامه جهت بررسی عملکرد تابع فوق ابتدا متغیر $orignum را مقداردهی کرده و سپس در خط ۱۱ آن‌را به عنوان آرگومان به تابع ( ) addFive ارسال کرده‌ایم. با این عمل یک کپی از مقدار متغیر $orignum در متغیر $num که نام آرگومان تابع مورد بحث است، ذخیره می‌شود. هرچند که مقدار متغیر $num به اندازه 5 واحد در درون تابع افزایش یافته است اما این عمل تأثیری روی مقدار متغیر $orignum ندارد. چنان‌چه مقدار متغیر $orignum را بر روی صفحه چاپ کنیم، خواهید دید که کماکان مقدار این متغیر برابر با 10 است. طبق یک قاعده پیش‌فرض،

آرگومان‌ها به روش ارسال از طریق مقدار به درون توابع راه می‌یابند. به‌عبارت دیگر، کپی مقادیر از طریق آرگومان‌ها به درون توابع ارسال می‌شوند.

با این حال می‌توان به‌واسطه ایجاد مرجعی به متغیر اصلی و ارسال آن مرجع به‌عنوان آرگومان به درون تابع، چنین رفتاری را به‌سادگی تغییر داد. می‌توان مرجع را به منزله علامت اشاره کننده‌ای فرض کرد که به یک متغیر اشاره می‌کند. با در درست داشتن مرجعی که به یک متغیر اشاره می‌کند، می‌توان مقدار آن متغیر را دستخوش تغییر نمود.

برنامه موجود در لیست ۱۴- ۶ ، استفاده از این روش را در عمل نشان می‌دهد. همان‌گونه که خط ۱۱ از این برنامه نیز نشان می‌دهد، هنگامی که آرگومانی از طریق مرجع به یک تابع ارسال می‌شود، محتوای متغیر ارسال شده به تابع (در اینجا متغیر $orignum) در دسترس کد تابع قرارگرفته و بدین ترتیب تابع قادر است تا مقدار آن را به راحتی تغییر دهد (این وضعیت را با حالت پیش فرض که تابع کپی مقدار متغیر را تغییر می‌دهد، مقایسه کنید). بدین ترتیب اعمال هر گونه تغییری بر روی آرگومان موجب تغییر مقدار اصلی خواهد شد. همان‌گونه که خط ۷ از این برنامه نیز نشان می‌دهد، جهت ارسال مقادیر به تابع از طریق مرجع لازم است تا هنگام تعریف تابع پیش از نام آرگومان مورد نظر از علامت آدرس‌دهی (&) استفاده کنیم.

```
 1: <html>
 2: <head>
 3: <title>Listing 6.14</title>
 4: </head>
 5: <body>
 6: <?php
 7: function addFive( &$num ) {
 8:      $num += 5;
 9: }
10: $orignum = 10;
11: addFive( $orignum );
12: print( $orignum );
13: ?>
14: </body>
15: </html>
```

لیست ۱۴- ۶ نمایش چگونگی ارسال مقادیر به تابع از طریق مرجع

تا پیش از این، چنین متداول بود که به‌جای تعیین فراخوانی توابع به روش ارسال آرگومان‌ها از طریق‌مرجع‌در بخش تعریف تابع این کار را هنگام فراخوانی در درون برنامه‌ای که تابع موردنظر را فراخوانی می‌کرد، انجام می‌دادند. این روش که به " call time pass - by - reference " یا " ارسال از طریق مرجع به‌هنگام فراخوانی" نامیده می‌شود، مستلزم استفاده از علامت & در جلوی نام آرگومان هنگام فراخوانی تابع (به‌جای تعریف تابع) بود. در حال حاضر این شیوه منسوخ شده و بنابراین نباید از آن استفاده کرد.

چنان‌چه کتابخانه مورد استفاده‌تان شامل توابعی است که از این شیوه منسوخ شده جهت ارسال آرگومان‌ها از طریق مرجع بهره می‌برند، می‌توانید با تنظیم گزینه ویژه‌ای در فایل .php Ini با عنوان allow _ call _ time _ pass _ reference از شر پیغامهای اخطاری که PHP هنگام استفاده از آنها پی در پی نمایش می‌دهد، خلاص شوید (به‌خاطر داشته باشید که این کار را به‌طور موقت انجام داده و پس از پایان کارتان گزینه مذکور را به صورت قبلی خود تنظیم نمایید).

ایجاد توابع بدون نام

در زبان PHP می‌توان توابع را هنگام اجرای برنامه‌های اسکریپت نیز ایجاد نمود. از آنجا که نام مشخصی به این‌گونه توابع منسوب نشده اما این توابع در درون متغیرهای برنامه ذخیره شده و یا خود به‌عنوان آرگومان به توابع دیگری در برنامه ارسال می‌شوند: معمولاً از اصطلاح توابع بدون نام در مورد آنها استفاده می‌شود. در زبان PHP4 از تابع سیستمی ویژه‌ای با نام ( ) create _ function جهت ایجاد توابع بدون نام استفاده می‌شود. تابع مذکور جهت اجرا به دو آرگومان نیاز دارد. اولین آرگومان این تابع لیستی از اسامی متغیرها که با علامت کاما از یکدیگر جدا شده‌اند، می‌باشد. این متغیرها در واقع آرگومان‌های تابع بدون نام می‌باشند و وظیفه‌ای کاملاً مشابه آرگومان‌های توابع معمولی را عهده‌دار هستند. دومین آرگومان شامل دستورالعملهای تابع بدون نام است.

برنامه لیست ۱۵- ۶ چگونگی ایجاد یک تابع بدون نام را نشان می‌دهد. این تابع بدون نام به‌سادگی حاصل مجموع دو عدد را محاسبه کرده و باز می‌گرداند.

```
1: <html>
2: <head>
3: <title>Listing 6.15</title>
4: </head>
5: <body>
6: <?php
7: $my_anon = create_function( '$a, $b', 'return $a+$b;' );
8: print $my_anon( 3, 9 );
9: // prints 12
10: ?>
11: </body>
12: </html>
```

لیست ۱۵- ۶ نمونه‌ای از یک تابع بدون نام

همان‌گونه که در این برنامه نیز مشاهده می‌کنید همواره بهتر است تا آرگومان‌های تابع ( ) create_ function را در قالب جفت علامت ' ' به این تابع ارسال کنیم چراکه این عمل باعث

می‌شود تا به‌دلیل وجود اسامی متغیرها که با علامت $ آغاز می‌شوند نیازی به روش escaping نداشته باشیم. در صورتی که آرگومان‌های تابع فوق را در درون جفت علامت " " به این تابع ارسال کنیم به صورت زیر لازم است تا به‌واسطه علامت $ در ابتدای نام متغیرها از تکنیک escaping بهره بگیریم:

```
$my _ anon = create _ function ( "\$a, \$b ", "return \$a + \$b ; ") ;
```

اما فایده از توابع بدون نام واقعاً در چیست؟ در عمل به احتمال قوی تنها هنگامی از آنها استفاده می‌کنید که بخواهید توابع سیستمی را با ارسال این توابع به آنها به‌عنوان توابع callback تغذیه کنید. تابع callback به تابعی گفته می‌شود که عموماً توسط برنامه‌نویس و با هدف فراخوانی (معمولاً مکرر) توسط تابعی که به‌عنوان آرگومان به آن ارسال می‌شود، طراحی می‌گردد. در درس ساعت شانزدهم با عنوان " بهره‌گیری از داده‌ها " مثالی را در این زمینه بررسی خواهیم کرد.

دومین آرگومان تابع ( ) create _ function بدنه تابعی است که قصد ایجاد آن‌را داریم. همواره به‌خاطر داشته باشید که آخرین عبارت بدنه مذکور را که در قالب یک دنباله کاراکتری به تابع فوق ارسال می‌شود، با علامت سمی‌کولون خاتمه دهید. در صورتی که این قاعده را رعایت نکنید، موتور PHP با نمایش پیغام خطا از اجرای تابع جلوگیری به‌عمل می‌آورد.

بررسی وجود تابع

همان‌گونه که احتمالاً تا به حال دریافته‌اید همواره از وجود تابعی که قصد فراخوانی آن‌را داریم، مطلع نیستیم. برای مثال اگر کد یک برنامه اسکریپت نمونه از نام تابعی که در درون یک متغیر ذخیره شده است، استفاده می‌کند بسیار به‌جا و مفید خواهد بود؛ اگر بتوانیم پیش از هر اقدامی جهت فراخوانی تابع مذکور به‌نحوی از وجود آن اطلاع حاصل کنیم. از این مهم‌تر باید به‌خاطر داشته باشید که نسخه‌های مختلف موتور PHP ممکن است عملکردهای مختلفی داشته باشند. اگر برنامه‌ای که در حال ایجاد آن هستید به احتمال زیاد بر روی چندین وب سرور به اجرا در خواهد آمد، بهتر است تا از وجود قابلیت و عملکرد مورد نظرتان بر روی آن سرورها اطمینان پیدا کنید. برای مثال اگر توابع mysql در دسترس باشند، می‌توانید کدی بنویسید که از بانک اطلاعاتی Mysql بهره‌مند شود. در صورتی که توابع مذکور در دسترس نباشند، داده‌ها به سادگی در یک فایل متن ثبت می‌شوند.

جهت بررسی وجود یک تابع پیش از استفاده از آن می‌توانیم از تابع دیگری با نام function _ exists ( ) بهره بگیریم. تابع function _ exists ( ) آرگومانی از نوع دنباله کاراکتری دریافت می‌کند که مشخص کننده نام تابع مورد بررسی است. چنانچه تابع در دسترس باشد، مقدار true و در غیر این‌صورت مقدار false به‌عنوان نتیجه بررسی باز می‌گردد.

برنامه لیست ۱۶- ۶ چگونگی استفاده از تابع ( ) function _ exists را در عمل نشان می‌دهد. این برنامه همچنین قابلیتهای دیگری را نیز به نمایش می‌گذارد که پیشتر در درس این ساعت به آنها اشاره کردیم.

```
1: <html>
2: <head>
3: <title>Listing 6.16</title>
4: </head>
5: <body>
6: <?php
7:
8: function tagWrap( $tag, $txt, $func="" ) {
9:     if ( ! empty( $txt ) && function_exists( $func ) )
10:         $txt = $func( $txt );
11:     return "<$tag>$txt</$tag>\n";
12: }
13:
14: function underline( $txt ) {
15:     return "<u>$txt</u>";
16: }
17:
18: print tagWrap('b', 'make me bold');
19: // <b>make me bold</b>
20:
21: print tagWrap('i', 'underline me too', "underline");
22: // <i><u>underline me too</u></i>
23:
24: print tagWrap('i', 'make me italic and quote me',
25:     create_function('$txt', 'return "&quot;$txt&quot;";'));
26: // <i>&quot;make me italic and quote me&quot;</i>
27:
28: ?>
29: </body>
30: </html>
```

لیست ۱۶- ۶ چگونگی بررسی وجود تابع

همان‌گونه که مشاهده می‌کنید، در خط ۹ از برنامه تابع ( ) tagWarp و در خط ۱۴ نیز تابع ( ) underLine تعریف کرده‌ایم. تابع ( ) tagWrap سه دنباله کاراکتری را به‌عنوان آرگومان دریافت می‌کند. آرگومان اول یک نشانه (تگ) از نشانه‌های زبان HTML است. آرگومان دوم متن مورد نظر برای قالب‌بندی است و آرگومان سوم نیز نام یک تابع اختیاری است. حاصل اجرای این تابع یک دنباله کاراکتری قالب‌بندی شده است. از طرف دیگر، تابع ( ) underline تنها یک آرگومان از نوع دنباله کاراکتری دریافت می‌کند. این آرگومان متنی است که باید قالب‌بندی شود. حاصل اجرای این تابع متنی است که توسط نشانه ‹ u › از زبان HTML قالب‌بندی می‌شود.

هنگام اولین فراخوانی، تابع ( ) tagWrap در خط ۱۸ کاراکتر ' b ' و دنباله کاراکتری ' make me bold ' به این تابع ارسال می‌شوند. از آن‌جا که هیچ مقداری را به‌عنوان آرگومان سوم در این

فراخوانی مشخص نکردیم، مقدار پیش‌فرض (دنباله کاراکتری تهی) به‌عنوان آرگومان سوم مورد استفاده قرار خواهد گرفت. در خط ۹ ابتدا بررسی تهی بودن متغیر $txt از کاراکتر مورد بررسی قرار گرفته و در صورتی که این متغیر شامل کاراکتر باشد، وجود تابع $func با استفاده از فراخوانی تابع function_exists ( ) بررسی می‌شود. البته به‌دلیل تهی بودن متغیر$func در خط ۱۱، متغیر $txt توسط تگ <b> نشانه‌گذاری شده و نتیجه به تابع فراخواننده بازگردانده شده است. در گام بعد، در خط ۲۱ از برنامه تابع tagWrap ( ) مجدداً فراخوانی شده و این بار کاراکتر 'i'، دنباله کاراکتری 'underline me too' و دنباله کاراکتری "underline"، به‌عنوان آرگومان‌های این تابع به آن ارسال شده‌اند. از آنجا که تابع function_exists ( ) وجود تابع underline ( ) را در خط ۱۴ تشخیص می‌دهد، تابع مذکور فراخوانی شده و پیش از هرگونه قالب‌بندی آرگومان $txt به آن ارسال می‌شود. حاصل فرآیند دنباله کاراکتری زیرخط‌داری است که به‌صورت ایتالیک به نمایش در می‌آید.

در نهایت، تابع tagWrap ( ) در خط ۲۴ برنامه با آرگومانی از نوع تابع بدون نام (که بدنه‌ای شامل یک متن داخل کوتیشن دارد) فراخوانی می‌شود. البته روش سریع‌تر این بود که به‌سادگی عناصر مورد نظرمان را به متن مورد قالب‌بندی اضافه کنیم ولی از این جهت روش فوق را انتخاب کردیم تا قابلیت استفاده از تابع function_exists ( ) را به‌همراه توابع بدون نام و تشابه آن در استفاده از تابع مذکور با اسامی توابع نشان دهیم.

جمع بندی

در درس این ساعت با چگونگی تعریف و استفاده از توابع و جزئیات مربوطه آشنا شدید. مشاهده کردید که به چه نحوی می‌توان توابع را تعریف کرده و اطلاعات دنیای خارج را به‌عنوان آرگومان به درون آنها ارسال کرد. در این بین با نحوه استفاده از عباراتی چون global و static و فواید مربوط به آنها آشنا شدید. ضمناً چگونگی ارسال آرگومان‌ها به روش ارسال از طریق مرجع را فراگرفته و نحوه استفاده از مقادیر پیش‌فرض آرگومان‌های تابع را مشاهده نمودید. در انتهای درس نیز چگونگی ایجاد توابع بدون نام و نحوه بررسی وجود تابع مورد استفاده را فرا گرفتید.

پرسش و پاسخ

پرسش: جدای از عبارت global، آیا روش دیگری می‌شناسید که با استفاده از آن بتوان در درون توابع متغیرهای سراسری را مورد دستیابی و تغییر قرار داد؟

پاسخ: آرایه انجمنی ویژه‌ای با نام $GLOBALS روش دیگری است که به کمک آن می‌توان از هرجایی از برنامه اسکریپت به متغیرهای سراسری دسترسی پیدا کرد. برای مثال، جهت دستیابی به

متغیر سراسری $test در درون یک تابع، می‌توان از عبارت $GLOBALS [' test '] جهت اشاره به آن استفاده کرد. در درس ساعت بعدی مطالب بیشتری در مورد این آرایه فرا می‌گیرید.

پرسش: آیا می‌توان فراخوانی تابعی را در درون دنباله‌های کاراکتری (یعنی به همان صورتی که در مورد متغیرها عملی است) انجام داد؟

پاسخ: خیر. فراخوانی توابع همواره باید در خارج از علامت کوتیشن انجام شود.

تمرینها

هدف‌از این بخش، دوره مطالب فراگیری شده در قالب آزمون است. پاسخ بخش آزمون بلافاصله بعد از آن آمده است. بخش فعالیتها شامل تمرینهایی است که به منظور افزایش مهارت برنامه‌نویسی خواننده طراحی شده و البته فاقد پاسخ است.

آزمون

۱- درستی یا نادرستی عبارت زیر را تعیین کنید:

" چنانچه تابعی جهت فراخوانی نیازی به آرگومان نداشته باشد، می‌توان از ذکر جفت پرانتز بعد از نام آن تابع به‌هنگام فراخوانی اجتناب کرد. "

۲- چگونه می‌توان مقداری را به‌عنوان نتیجه عملیات تابع به برنامه فراخواننده باز گرداند؟

۳- خروجی قطعه برنامه زیر چیست؟

```
$number = 50 ;
function tenTimes ( ) {
   $number = $number * 10 ;
}
tenTimes ( ) ;
print  $number ;
```

۴- خروجی قطعه برنامه زیر چیست؟

```
$number = 50 ;
function tenTimes ( ) {
   global  $number ;
   $number = $number * 10 ;
}
tenTimes ( ) ;
print  $number ;
```

۵- خروجی قطعه برنامه زیر چیست؟

```
$number = 50 ;
function tenTimes ($n ) {
```

```
    $n = $n * 10 ;
}
tenTimes ( $number ) ;
print  $number ;
```

۶- خروجی قطعه برنامه زیر چیست؟

```
$number = 50 ;
function tenTimes ( &$n) {
    $n = $n * 10 ;
}
tenTimes ($number ) ;
print  $number ;
```

پاسخ آزمون

۱- این عبارت نادرست است. استفاده از علامت جفت پرانتز هنگام فراخوانی توابع ضروری بوده و این امر ربطی به وجود آرگومان یا عدم وجود آن ندارد.

۲- با استفاده از عبارت return می‌توان نتیجه را به تابع فراخواننده بازگرداند.

۳- خروجی این برنامه عدد 50 است. تابع ( ) tenTimes هیچ‌گونه دسترسی به متغیر سراسری $number ندارد و از این‌رو هنگام فراخوانی متغیر محلی هم نام با آن‌را تغییر می‌دهد.

۴- خروجی این برنامه عدد 500 است. از آنجا که هنگام معرفی متغیر $number در درون تابع از عبارت global استفاده شده است تابع مذکور به متغیر سراسری هم نام با آن دسترسی دارد.

۵- خروجی این برنامه عدد 50 است. ارسال آرگومان به تابع ( ) tenTimes به روش عادی یعنی ارسال از طریق مقدار صورت گرفته است. به‌عبارت دیگر، کپی مقدار ارسالی در متغیر $n ذخیره می‌شود. از این‌رو هرگونه تغییری بر روی این متغیر هیچ‌گونه تغییری را بر متغیر $number در پی ندارد.

۶- خروجی این برنامه عدد 500 است. این بار با افزودن علامت & به ابتدای نام آرگومان (هنگام تعریف تابع) این اطمینان حاصل می‌شود که ارسال آرگومان به تابع به روش ارسال از طریق مرجع انجام می‌شود. بدین ترتیب هرگونه تغییر در متغیر $n در متغیر $number نیز منعکس می‌گردد.

فعالیتها

۱- تابعی با چهار آرگومان از نوع دنباله کاراکتری ایجاد کنید. این تابع باید یک دنباله کاراکتری را که شامل المان table از زبان HTML است بازگرداند. هر یک از خانه‌های این جدول باید شامل یکی از آرگومان‌های تابع باشد.

ساعت هفتم

آرایه‌ها

آرایه‌ها و روشهای دستکاری آنها یکی از نقاط قوت زبان برنامه‌نویسی PHP4 در ایجاد برنامه‌های کارآمد و بهبود آنهاست. پس از کسب مهارت در کار با آرایه‌ها به راحتی قادر خواهید بود تا به ذخیره و سازماندهی ساختارهای داده پیچیده در برنامه‌ها بپردازید.

در درس این ساعت به معرفی آرایه‌ها و برخی از توابعی می‌پردازیم که کار با آرایه‌ها را بسیار ساده می‌کنند، در درس ساعت حاضر با مباحث زیر آشنا خواهید شد:

- **ماهیت آرایه‌ها و چگونگی ایجاد آنها**
- **چگونگی دستیابی به داده‌های موجود در آرایه‌ها**
- **چگونگی مرتب‌سازی داده‌های موجود در آرایه ها**
- **چگونگی ایجاد توابعی که کار با آرایه‌ها را ساده می‌کنند**

در ادامه به بحث در مورد مطالب فوق می‌پردازیم.

ماهیت آرایه‌ها

تا کنون باید این نکته را به‌خوبی درک کرده باشید که می‌توان به‌طور موقت، مقداری را در یک متغیر محلی ذخیره نمود. با استفاده از متغیرها در برنامه می‌توان مقادیر را ذخیره کرده و آنها را پردازش نمود. بدین ترتیب با دادن مقادیر مختلف می‌توان انتظار خروجی‌های مختلفی را نیز داشت. متاسفانه متغیرها دارای نقطه ضعف مشهودی هستند و آن این است که تنها یک مقدار را در یک زمان مشخص می‌توان در یک متغیر ذخیره نمود. این در حالی است که آرایه‌ها بر این مشکل فائق آمده‌اند. آرایه در واقع یک متغیر خاص است. بدین ترتیب این امکان وجود دارد که چندین مقدار مختلف را به‌طور همزمان در این متغیر خاص، ذخیره کنیم. هر یک از مقادیر ذخیره شده در آرایه با استفاده از یک عدد یا یک دنباله کاراکتری منحصر به‌فرد شاخص‌گذاری می‌شوند. اگر متغیر را به‌عنوان جعبه‌ای فرض کنیم که می‌توان در آن چیزی را نگهداری کرد، آن‌گاه می‌توانیم آرایه را در حکم قفسه‌ای فرض کنیم که شامل چندین عدد از این جعبه‌ها می‌باشد.بدین ترتیب فضایی در اختیارمان است که می‌توانیم از آن، جهت نگهداری چندین موجودیت مجزا استفاده نماییم.

واضح است اگر تعداد پنج مقدار مختلف جهت ذخیره سازی در اختیار داشته باشیم، می‌توانیم به‌ازای هرکدام یک متغیر و در مجموع پنج متغیر مختلف ایجاد کنیم. وضعیت فوق ممکن است این پرسش را در ذهن ایجاد کند که لزوم استفاده از آرایه در حالی‌که می‌توان از چندین متغیر به‌جای آن استفاده کرد در چیست؟

پیش از هر چیز باید گفت که آرایه‌ها بسیار قابل انعطاف هستند. همواره می‌توان دو مقدار یا صد مقدار و یا بیشتر را بدون نیاز به تعریف متغیرهای بیشتر در یک آرایه نگهداری کرد. نکته بعدی این است که کار با عناصر ذخیره شده در آرایه همواره با تسهیلات و امکانات ویژه‌ای همراه است. برای مثال فرآیند پردازش آرایه به‌گونه‌ای که در هربار گذر از درون یک ساختار تکرار یکی از عناصر آرایه پردازش شود (مثل انتخاب تصادفی یک عنصر در هر بار گذر از حلقه)، امری است که به‌سادگی امکان‌پذیر است. همچنین انجام فرآیندهای دیگری مثل مرتب سازی عناصر آرایه بر حسب ترتیب عددی یا الفبایی یا هر ترتیب دیگری که برنامه‌نویس تعیین می‌کند، بسیار ساده است.

چنین متداول است که به هر یک از مقادیر ذخیره شده در آرایه، یک عنصر یا المان گفته می‌شود. به هریک از عناصر آرایه می‌توان از طریق شاخص مربوطه دست پیداکرد. همان‌گونه که پیشتر نیز اشاره کردیم، شاخص عناصر آرایه می‌تواند از نوع عددی یا دنباله کاراکتری باشد.

طبق پیش‌فرض شاخص عناصر هر آرایه‌ای از نوع شاخص عددی بوده و از عدد صفر آغاز می‌شود. به‌خاطر سپردن این نکته مهم است که به‌دلیل شروع اولین شاخص از عدد صفر، شاخص آخرین عنصر یکی کمتر از تعداد عناصر آن آرایه خواهد بود (شاخص گذاری عددی به‌طور ترتیبی انجام

می‌شود بدین ترتیب که شاخص اول عدد صفر، شاخص دوم عدد یک، شاخص سوم عدد دو و الی آخر می‌باشد).

به‌عنوان یک نمونه، جدول ۱- ۷ را که شامل عناصر یک آرایه با نام users می‌باشد، در نظر بگیرید. دقت کنید که شاخص سومین عنصر این آرایه عدد ۲ می‌باشد.

جدول ۱- ۷ المان‌های موجود در آرایه users

شماره شاخص	مقدار	مرتبه المان در آرایه
0	Bert	اولین
1	Sharon	دومین
2	Betty	سومین
3	Harry	چهارمین

شاخص گذاری عناصر آرایه با استفاده از دنباله‌های کاراکتری هنگامی می‌تواند مفید واقع شود که علاوه بر مقادیر نیازمند، نگهداری اسامی (شاخصها) نیز باشیم.

در زبان PHP4 تسهیلات متعدد و ویژه‌ای جهت دستیابی و دستکاری عناصر آرایه‌های شاخص گذاری شده به هر دو روش پیش‌بینی شده است. برخی از این تسهیلات را در درس این ساعت مورد بررسی قرار می‌دهیم و سایر امکانات را به درس روز شانزدهم با عنوان " بهره‌گیری از داده‌ها " موکول می‌کنیم.

ایجاد آرایه‌ها

طبق پیش‌فرض، آرایه‌ها را می‌توان لیستی از مقادیر فرض کرد که به روش شاخص گذاری عددی عناصر آن از یکدیگر متمایز شده‌اند. به دو روش می‌توان مقادیر مورد نظر را به عناصر آرایه نسبت داد. روش اول استفاده از تابع سازنده‌ای با عنوان ( ) array است. روش دوم بهره‌گیری از عملگر [ ] است. در دو قسمت بعدی این درس با این دو روش آشنا می‌شوید.

تعریف آرایه‌ها با استفاده از تابع سازنده ( ) array

تابع سازنده ( ) array هنگامی مفید است که بخواهیم عناصر یک آرایه را به‌طور هم‌زمان مقداردهی کنیم. مثال ذکر شده آرایه‌ای با نام $users را نشان می‌دهد که در حال مقداردهی با چهار دنباله کاراکتری است:

```
$users = array ( " Bert ", " Sharon ", " Betty ", " Harry " ) ;
```

با در دست داشتن این تعریف، اکنون می‌توان به شیوه زیر با استفاده از شاخص شماره ۲ به سومین عنصر این آرایه دست پیدا کرد:

```
Print $users [ 2 ] ;
```

عبارت فوق به‌سادگی دنباله کاراکتری " Betty " را بر روی صفحه نمایش می‌دهد. شاخص عنصر مورد دستیابی همواره در داخل جفت علامت [ ] که بلافاصله بعد از نام آرایه قرار می‌گیرد، واقع می‌شود. از این روش هم برای مقداردهی و هم برای بازیابی مقدار عنصر خاصی از یک آرایه می‌توان استفاده کرد.

به‌خاطر داشته باشید که طبق پیش‌فرض، عناصر آرایه‌ها همواره از عدد صفر شاخص گذاری می‌شوند. از این‌رو شاخص هر عنصر آرایه یکی کمتر از موقعیت آن عنصر در آرایه است. برای مثال شاخص سومین عنصر آرایه برابر با عدد ۲ وشاخص هشتمین عنصر برابر با عدد ۷ است.

تعریف آرایه و مقداردهی آن با استفاده از عملگر شناسه آرایه

با استفاده از عملگر شناسه آرایه (جفت علامت [ ]) به‌همراه نام آرایه می‌توان آرایه‌ای را تعریف کرده و یا در صورت وجود، آن را مقدار دهی کرد. شناسه آرایه به‌سادگی یک جفت علامت [ ] است که داخل آن هیچ‌عدد یا دنباله کاراکتری واقع نشده باشد.

اجازه دهید در این قسمت آرایه $users را با همین روش تعریف کنیم:

```
$users [ ] = " Bert " ;
$users [ ] = " Sharon " ;
$users [ ] = " Betty " ;
$users [ ] = " Harry " ;
```

دقت کنید که از هیچ عددی در درون جفت علامت [ ] استفاده نکرده‌ایم. زبان برنامه‌نویسی PHP4 دارای مکانیزمی است که به طور خودکار مراقب صحت شاخص گذاری عناصر آرایه می‌باشد. بدین ترتیب برنامه‌نویس از دردسرهای معمول شاخص گذاری عددی خلاصی می‌یابد.

البته در صورت تمایل می‌توانیم از شاخص نیز در درون جفت علامت [ ] استفاده کنیم که البته تفاوتی در نتیجه نهایی ایجاد نمی‌کند. با این حال توصیه می‌کنیم که از این روش خودداری کنید. به نمونه زیر که مقداردهی دو عنصر آرایه $users است، توجه نمایید:

```
$users [ 0 ] = " Bert " ;
$users [ 200 ] = " Sharon " ;
```

همان‌گونه که ملاحظه می‌کنید قطعه کد فوق آرایه $users را با دو عنصر تعریف می‌کند، اما توجه کنید که شاخص آخرین عنصر آن عدد 200 است. در چنین حالتی PHP4 عناصر میانی این دو عنصر را مقداردهی نمی‌کند. این وضعیت می‌تواند هنگام دستیابی به عناصر آرایه، موجب سردرگمی

شود. از طرف دیگر، ممکن است با شرایطی مواجه شویم که بخواهیم از اعداد دلخواه به‌عنوان شاخص عناصر آرایه استفاده کنیم.

از عملگر شناسه آرایه علاوه بر تعریف آرایه‌ها می‌توانیم جهت اضافه کردن مقادیر جدید به انتهای یک آرایه موجود نیز استفاده نماییم. در قطعه کدی که در ادامه مشاهده خواهید کرد، ابتدا با استفاده از تابع سازنده ( ) array، آرایه‌ای را ایجاد کرده و سپس با به‌کارگیری عملگر مذکور عنصر جدیدی را به آن اضافه کرده‌ایم:

```
$users = array ( " Bert "," Sharon ", " Betty ", " Harry " ) ;
$users [ ] = " Sally " ;
```

آرایه‌های انجمنی

آرایه‌هایی که با اعداد شاخص‌گذاری شده‌اند، معمولاً هنگامی مورد استفاده قرار می‌گیرند که بخواهیم عناصر مورد نظر را با همان ترتیبی که به آرایه وارد می‌شوند و یا در حالت پیچیده‌تر، مطابق با یک الگوی مشخص، مرتب نماییم. با وجود این، گاهی اوقات لازم است تا عناصر آرایه را با استفاده از اسامی آنها مورد دستیابی قرار دهیم. آرایه‌های انجمنی آرایه‌هایی هستند که به جای اعداد از دنباله‌های کاراکتری جهت شاخص‌گذاری آنها استفاده می‌کنیم (در این حالت نیز مانند حالت قبل از عملگر شناسه آرایه استفاده می‌شود). دفترچه تلفنی را درنظر بگیرید. کدامیک را ساده می‌یابید؟ شاخص عددی 4 برای مقدار " name " یا شاخص دیگری از نوع دنباله کاراکتری، مثلاً " name " ؟

در مورد آرایه‌های انجمنی نیز می‌توانید هم از تابع سازنده ( ) array و هم از عملگر شناسه آرایه، یعنی [ ] جهت تعریف این‌گونه آرایه‌ها استفاده کنید.

> توجه داشته باشید که تقسیم‌بندی آرایه‌هایی که با مقادیر عددی شاخص‌گذاری می‌شوند و آرایه‌هایی که با دنباله‌های کاراکتری شاخص بندی می‌گردند (آرایه‌های انجمنی)، مرز مشخصی ندارد. به‌عبارت دیگر آن‌چه که در زبان برنامه‌نویسی perl در مورد تفاوت کاملاً مشهود مابین آرایه‌ها و hash ها مشاهده می‌کنیم، در زبان PHP در مورد آرایه‌های عادی و انجمنی وجود صددرصد ندارد. با این همه بهتر است در صورت امکان این دو را از یکدیگر متمایز درنظر بگیریم چراکه روش دستیابی و تغییر مقادیر ذخیره شده در هریک از آنها کاملاً مستقل و متفاوت از دیگری است.

تعریف آرایه‌های انجمنی با استفاده از تابع سازنده ( ) array

جهت تعریف آرایه‌های انجمنی با استفاده از تابع سازنده ( ) array لازم است تا هم کلید و هم مقدار هر عنصر را به‌طور مجزا مشخص کنیم. برای نمونه قطعه کد زیر که یک آرایه انجمنی با نام

$character را با چهار عنصر تعریف می‌کند، در نظر بگیرید:

```
$character = array (
   "name " ⇒ "bob ",
   " occupation " ⇒ " superhero ",
   " age " ⇒ 30 ,
   " special power " ⇒ " x - ray vision "
   ) ;
```

با در دست داشتن این تعریف اکنون می‌توانیم به هر یک از عناصر آرایه فوق دسترسی داشته باشیم. به دسترسی نمونه زیر توجه کنید:

```
Print $character [ ' age ' ] ;
```

کلیدهای دستیابی به عناصر آرایه‌های انجمنی از نوع دنباله کاراکتری هستند. چنان‌چه هنگام ذکر نام این کلیدها در عملگر شناسه آرایه از علامت کوتیشن استفاده نکنید، بنابه پیش‌فرض موتور PHP هیچ‌گونه اعتراضی نخواهد داشت(به‌همین دلیل در ویرایش اول کتاب هنگام ذکر اسامی کلیدها از علامت کوتیشن استفاده‌ای به‌عمل نیامد).

با این حال توصیه ما این است که هنگام دستیابی به مقادیر آرایه‌های انجمنی کلیدهای مربوطه را در درون علامت کوتیشن قرار دهید. چنانچه تنظیمات موتور PHP مورد استفاده‌تان به‌گونه‌ای است که از کنار این موضوع به‌سادگی گذر نمی‌کند، به‌ازای هر مرتبه دستیابی بدون استفاده از علامت کوتیشن، موتور PHP با نمایش پیغام شما را مطلع خواهد کرد. از آن بدتر هنگامی است که تصادفاً نام یک کلید دستیابی با یک مقدار ثابت از برنامه معادل باشد. در این حالت مقدار ثابت آن، جایگزین نام کلید دستیابی معادل خواهد شد.

این مطلب را به‌خاطر بسپارید: چنانچه کلید دستیابی مورد نظرتان به‌سادگی یک دنباله کاراکتری است، استفاده از علامت کوتیشن یک ضرورت انکارناپذیر است:

```
Print $ character [ age ] ; // wrong
Print  $character [ " age " ] ; // right
```

اما در صورتی که کلید مذکور در یک متغیر ذخیره شده باشد، لازم است که استفاده از علامت کوتیشن را از یاد ببرید:

```
$agekey = " age " ;
print $character [ $agekey ] ; // right
```

تعریف مستقیم آرایه‌های انجمنی و افزودن مستقیم عناصر به آنها

با منسوب کردن مقداری به یک عنصر نام‌گذاری شده می‌توان اقدام به ایجاد یا اضافه کردن زوج " نام ـ مقدار " به آرایه‌های انجمنی نمود. در قطعه کدی که در زیر مشاهده می‌کنید آرایه انجمنی

$character را مجددا با نسبت دادن مستقیم مقادیر مورد نظرمان به کلیدهای مربوطه ایجاد کرده‌ایم:

```
$character [ "name " ] = "bob" ;
$character [ "occupation" ] = "superhero" ;
$character [ "age" ] = 30 ;
$character [ "special power" ] = "x - ray vision" ;
```

آرایه‌های چند بعدی

تا بدین جا به‌سادگی اظهار کردیم که عناصر آرایه مجموعه‌ای از مقادیر هستند. در مورد آرایه $character که در قسمت قبل آن‌را ایجاد کردید، سه عنصر از نوع دنباله کاراکتری و یک عنصر از نوع عدد صحیح بودند. با این وجود شرایط در دنیای واقعی اندکی پیچیده‌تر از این می‌باشد. در واقع هر عنصر از یک آرایه می‌تواند به‌سادگی یک مقدار، یک شئ و یا حتی خود آرایه‌ای دیگر باشد. آرایه‌های چند بعدی آرایه‌هایی هستند که عناصر آنها، نیز آرایه دیگری هستند. آرایه‌ای را فرض کنید که در هر یک از عناصر خود، آرایه دیگری را ذخیره کرده‌است، در این صورت جهت دستیابی به سومین عنصر از آرایه دوم لازم است تا از دو شاخص استفاده کنیم:

```
$array [ 1 ] [ 2 ]
```

این حقیقت که عناصر یک آرایه می‌توانند خود، آرایه‌های دیگری باشند به ما اجازه می‌دهد تا به‌سادگی قادر به ایجاد ساختارهای داده‌ای بسیار پیچیده باشیم. برنامه لیست ۱- ۷ آرایه‌ای را تعریف می‌کند که هر عنصر آن خود یک آرایه انجمنی است.

```
 1: <html>
 2: <head>
 3: <title>Listing 7.1</title>
 4: </head>
 5: <body>
 6: <?php
 7:
 8: $characters = array (
 9:             array (
10:                 "name"=> "bob",
11:                 "occupation" => "superhero",
12:                 "age" => 30,
13:                 "specialty" =>"x-ray vision"
14:             ),
15:             array (
16:                 "name" => "sally",
17:                 "occupation" => "superhero",
18:                 "age" => 24,
19:                 "specialty" => "superhuman strength"
20:             ),
21:             array (
22:                 "name" => "mary",
23:                 "occupation" => "arch villain",
24:                 "age" => 63,
```

لیست ۱- ۷ تعریف یک آرایه چند بعدی

```
25:                 "specialty" =>"nanotechnology"
26:             )
27:         );
28:
29: print $characters[0]['occupation'];
30: // prints "superhero"
31: ?>
32: </body>
33: </html>
```

دنباله لیست ۱-۷

همان‌گونه که در این برنامه مشاهده می‌کنید، از چند تابع سازنده array ( ) در درون یک تابع سازنده array ( ) استفاده کرده‌ایم. در نخستین سطح، آرایه‌ای را با این تابع سازنده تعریف نموده‌ایم. در سطح بعد، به عنوان هر یک از عناصر آرایه، یک آرایه انجمنی تعریف کرده‌ایم.

از این جهت عبارت $character[2] سومین آرایه انجمنی (که تعریف آن از خط ۲۱ برنامه تا خط ۲۶ صورت گرفته است) از آرایه اصلی که تعریف آن از خط ۸ برنامه آغاز شده است را در دسترس ما قرار می‌دهد. با پیروی از همین رویه می‌توانیم به‌سادگی عناصر هر یک از آرایه‌های انجمنی (آرایه‌های فرعی) را مورد دستیابی قرار دهیم. بدین ترتیب عبارت $character[2] [' name '] دنباله کاراکتری " mary" و عبارت $character[2] [' age '] مقدار عددی 63 را در دسترس ما قرار می‌دهد.

اکنون با روشن شدن مفهوم فوق، به‌سادگی می‌توانیم ترکیبات پیچیده‌ای از آرایه‌های انجمنی و آرایه‌هایی که به‌صورت عددی شاخص‌گذاری شده‌اند، ایجاد کنیم.

دستیابی به آرایه‌ها

تا بدین‌جا روشهایی را جهت ایجاد آرایه‌ها و اضافه‌کردن عناصر مورد نظرمان به آنها مورد بررسی قرار دادیم. در این قسمت قصد داریم تا برخی از ابزارها و امکانات PHP4 را که جهت دستیابی به اطلاعات مهمی درباره آرایه‌ها و نیز دستیابی به عناصر آنها طراحی و پیش‌بینی شده‌اند، مورد آزمایش قرار دهیم.

دستیابی به اندازه یک آرایه

همان‌گونه که می‌دانید، دستیابی به عنصری از یک آرایه از طریق شاخص مربوطه امکان‌پذیر است:

```
Print $user[ 4 ] ;
```

با این همه، به‌دلیل انعطاف آرایه‌ها در اضافه کردن مقادیر جدید به آنها بعید است که همواره بتوانیم تعداد عناصر آرایه‌هایی را که در حال کار با آنها هستیم، به‌خاطر بسپاریم. از این‌رو طراحان

PHP4 تابعی را با عنوان ( ) count جهت حل این مشکل طراحی کرده‌اند. تابع ( ) count به سادگی تعداد عناصر موجود در یک آرایه را باز می‌گرداند. در قطعه کد زیر آرایه‌ای را تعریف کرده‌ایم که عناصر آن با استفاده از روش شاخص‌گذاری عددی از یکدیگر متمایز شده‌اند و سپس با به‌کارگیری تابع ( )count سعی نموده‌ایم تا آخرین عنصر آن‌را مورد دستیابی قرار دهیم:

```
$users = array (" Bert "," Sharon ", " Betty ", " Harry " ) ;
print $users [ count ( $users ) _ 1 ] ;
```

دقت کنید که عدد 1 را از حاصل فراخوانی تابع ( ) count با آرگومان $users کم کرده‌ایم. این بدان دلیل است که تابع ( ) count همواره تعداد عناصر موجود در آرایه را باز می‌گرداند (معمولاً این فرض غلط و متداول در میان برنامه‌نویسان مبتدی که تابع فوق شاخص آخرین عنصر موجود در آرایه را به دست می‌دهد نتایج غیر منتظره ای را به‌بار می‌آورد).

با وجودی که بنا به پیش‌فرض شاخص‌گذاری عددی آرایه‌ها از عدد صفر آغاز می‌شود، می‌توان این رفتار را به‌سادگی تغییر داد. با این حال به‌دلیل حفظ سازگاری و وضوح برنامه معمولاً کمتر کسی تن به این کار می‌دهد. ما نیز چنین عملی را پیشنهاد نمی‌کنیم. هرچند که تابع ( ) count اندازه آرایه را در اختیار می‌گذارد، با استفاده از این تابع تنها می‌توان آخرین عضو آرایه را مورد دستیابی قرار داد، آن‌هم به شرطی که مطمئن باشیم عناصر آرایه به‌ترتیب وارد آرایه شده‌اند. چنانچه آرایه $users را با شاخصهای دلخواه و بدون ترتیب به‌صورت زیر مقداردهی کنیم:

```
$users[ 66 ]= " Bert " ;
$users[ 100 ]= " Sharon " ;
$users[ 556 ]= " Betty " ;
$users[ 703 ]= " Harry ";
```

آن‌گاه تابع ( ) count در یافتن عنصر آخر آرایه موفق نخواهد بود. این آرایه کماکان شامل ۴ عنصر است، اما عنصری که شاخص آن عدد ۳ باشد، در این آرایه وجود ندارد. از این‌رو اگر از ترتیب شاخص‌گذاری عناصر آرایه مورد استفاده‌تان اطمینان ندارید، بهتر است استفاده از تابع ( ) count را جهت دستیابی به فراموشی بسپارید. در این‌گونه موارد می‌توانید از تابع ( ) end کمک بگیرید. این تابع به‌سادگی آخرین عنصر آرایه را در اختیارتان قرار می‌دهد. تابع مذکور جهت اجرا نیازمند نام آرایه مورد نظر است و همان‌گونه که گفته شد، مقدار بازگشتی این تابع آخرین عنصر آرایه مورد نظر است. عبارت زیر بدون توجه به نحوه شاخص‌گذاری عناصر آرایه قادر است تا آخرین عنصر موجود در آن را نمایش دهد:

```
Print end ( $users) ;
```

استفاده از ساختارهای تکرار جهت پردازش عناصر آرایه

روشهای متعددی جهت پردازش عناصر آرایه با استفاده از ساختارهای تکرار وجود دارد. برای این‌کار ترجیح می‌دهیم تا از ساختار بسیار کارآمد foreach زبان PHP استفاده کنیم. در درس ساعت شانزدهم با عنوان " بهره‌گیری از داده‌ها " روشهای دیگری را ملاحظه خواهید نمود.

استفاده از ساختار تکرار foreach جهت پردازش آرایه‌هایی که به روش عددی شاخص گذاری شده‌اند، به شکل عمومی زیر صورت می‌گیرد:

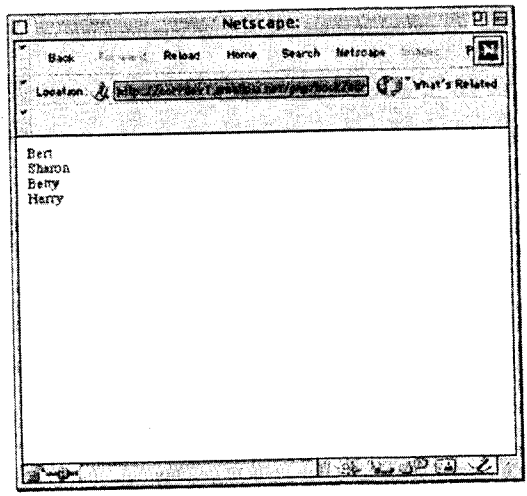
```
foreach ( $array as $temp ) {
   // ...
}
```

در این شکل کلی، $array نام آرایه‌ای است که در ساختار تکرار foreach قصد پردازش آن‌را داریم. همچنین $temp نام متغیری است که عناصر آرایه را به‌طور موقت در آنجا ذخیره خواهیم کرد.

در قطعه کد زیر ابتدا آرایه‌ای را که عناصر آن شاخص‌گذاری عددی شده‌اند، تعریف کرده و سپس با استفاده از ساختار foreach مقادیر عناصر آن‌را مورد دستیابی قرار داده‌ایم:

```
$users = array (" Bert "," Sharon ", " Betty ", " Harry " ) ;
foreach ( $users as $val ) {
   print "$val < br >" ;
}
```

خروجی حاصل از این قطعه کد در شکل ۱- ۷ قابل بررسی است.



شکل ۱- ۷ دستیابی به هریک از عناصر آرایه در درون ساختار حلقه تکرار

همان‌گونه که مشاهده می‌کنید ابتدا مقدار هر عنصر آرایه در متغیر موقت $val ذخیره می‌شود و سپس توسط تابع ( ) print بر روی صفحه به نمایش در می‌آید. چنانچه از آن دسته خوانندگانی

هستید که پیشتر با زبان برنامه‌نویسی perl کار می‌کردید، دقت کنید که ساختار foreach رفتار متفاوتی در اینجا دارد که در نوع خود دارای اهمیت است. نکته اینجاست: تغییر متغیر موقت$temp در ساختار تکرار Foreach زبان perl موجب تغییر عناصر متناظر با آن آرایه می‌شود. این در حالی‌است که اعمال تغییر بر روی متغیر موقت $temp در ساختار تکرار foreach زبان PHP هیچ‌گونه تأثیری بر روی عنصر متناظر خود نخواهد داشت. در درس ساعت شانزدهم به بررسی روشی خواهیم پرداخت که تغییر مقادیر آرایه‌های شاخص گذاری شده به روش عددی با استفاده از ساختار foreach را ممکن می‌سازد.

پردازش عناصر آرایه‌های انجمنی با استفاده از ساختارهای تکرار

جهت دستیابی به کلیدهای دستیابی و مقادیر متناظر با آنها در آرایه‌های انجمنی، لازم است تا تغییراتی را در به‌کارگیری ساختار تکرار foreach لحاظ کنید.

شکل عمومی استفاده از ساختار foreach در رابطه با آرایه‌های انجمنی به‌صورت زیر است:

```
foreach ( $array as $key ⇒ $value ) {
  // ...
}
```

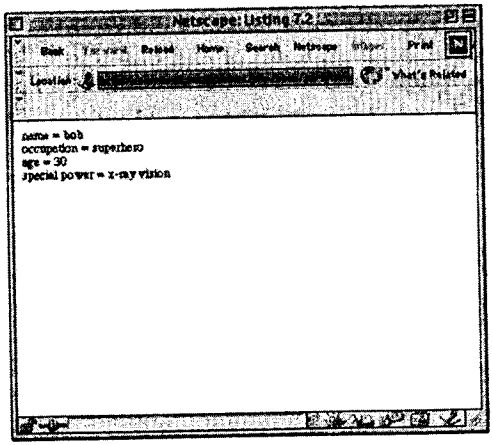
در این الگوی کلی $array نام آرایه انجمنی مورد نظر است. همچنین $key نام متغیری است که موقتاً نام هریک از کلیدهای دستیابی را نگه می‌دارد و بالاخره $value نیز نام متغیری است که باز هم به طور موقت مقدار متناظر با هر کلید دستیابی را ذخیره می‌کند.

برنامه موجود در لیست ۲- ۷ چگونگی دستیابی به عناصر یک آرایه انجمنی را با استفاده از این ساختار تکرار نشان می‌دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 7.2</title>
 4: </head>
 5: <body>
 6: <?php
 7: $character = array (
 8:              "name" => "bob",
 9:              "occupation" => "superhero",
10:              "age" => 30,
11:              "special power" => "x-ray vision"
12:              );
13: foreach ( $character as $key=>$val ) {
14:     print "$key = $val<br>";
15: }
16:
17: ?>
18: </body>
19: </html>
```

لیست ۲- ۷ استفاده از ساختار تکرار foreach جهت دستیابی به عناصر یک آرایه انجمنی

در این برنامه آرایه انجمنی $character ، در خط ۷ ایجاد شده است. در خط ۱۳ از برنامه با استفاده از ساختار تکرار foreach عناصر این آرایه مورد دستیابی قرار گرفته است. مقدار هر کلید دستیابی در متغیر $key و مقدار متناظر با آن در آرایه، در متغیر $val به‌ازای هر عنصر از آرایه ذخیره می‌شود. در خط ۱۴ از برنامه نهایتاً این مقدار بر روی صفحه نمایش داده می‌شود. خروجی حاصل از این برنامه در شکل ۲- ۷ قابل بررسی است.



شکل ۲- ۷ دستیابی به عناصر یک آرایه انجمنی با استفاده از ساختار تکرار foreach

پردازش آرایه‌های چند بعدی

اکنون می‌توانید با ترکیب تکنیکهایی که تا بدین‌جا فرا گرفته‌اید، مقادیر عناصر آرایه چند بعدی را که در برنامه لیست ۱- ۷ آن‌را ایجاد کردید، به نمایش درآورید. برنامه موجود در لیست ۳- ۷ ابتدا آرایه مشابهی را تعریف کرده و سپس با استفاده از ساختار تکرار foreach عناصر آن را نمایش می‌دهد.

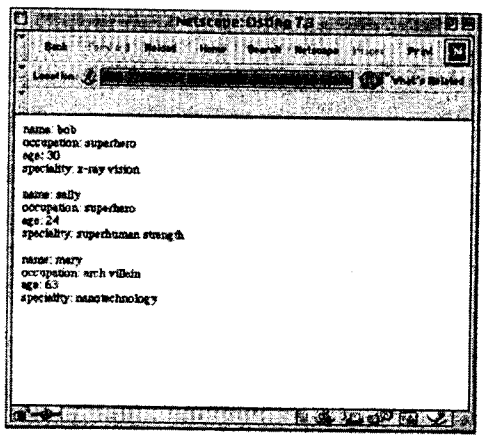
```
1: <html>
2: <head>
3: <title>Listing 7.3</title>
4: </head>
5: <body>
6: <?php
7: $characters = array (
8:             array (
9:                 "name"=> "bob",
10:                "occupation" => "superhero",
11:                "age" => 30,
12:                "specialty" =>"x-ray vision"
13:            ),
14:            array (
15:                "name" => "sally",
16:                "occupation" => "superhero",
17:                "age" => 24,
```

لیست ۳-۷ دستیابی به عناصر یک آرایه چند بعدی

```
18:                 "specialty" => "superhuman strength"
19:             ),
20:             array (
21:                 "name" => "mary",
22:                 "occupation" => "arch villain",
23:                 "age" => 63,
24:                 "specialty" =>"nanotechnology"
25:             )
26:         );
27:
28: foreach ( $characters as $val ) {
29:     foreach ( $val as $key=>$final_val ) {
30:         print "$key: $final_val<br>";
31:     }
32:     print "<br>";
33: }
34:
35: ?>
36: </body>
37: </html>
```

دنباله لیست ۳- ۷

خروجی حاصل از اجرای برنامه این لیست در شکل ۳- ۷ قابل ملاحظه است. همان‌گونه که در کد برنامه مشاهده می‌کنید، در خطوط ۲۸ و ۲۹ دو ساختار تکرار foreach ایجاد کرده‌ایم. حلقه خارجی در خط ۲۸ هر یک از عناصر آرایه $characters (که یک آرایه با شاخصهای عددی است) را مورد دستیابی قرار داده و در هربار اجرای حلقه مقدار هر یک از این عناصر را در متغیر $val قرار می‌دهد. از آنجا که خود متغیر $val بدین ترتیب در هربار اجرای حلقه خارجی شامل یک آرایه انجمنی خواهد بود، حلقه دیگری که در خط ۲۹ آن را تعریف کرده‌ایم هر یک از عناصر آرایه انجمنی فوق را (که کلید دستیابی و مقدار هر یک به‌ترتیب در متغیرهای $key و $final _ val ذخیره می‌شود) به نمایش خواهد گذاشت.



شکل ۳- ۷ دستیابی به عناصر یک آرایه دو بعدی

برای اینکه تکنیک فوق به‌خوبی کار کند، لازم است مطمئن شویم که متغیر $val همواره شامل یک آرایه می‌باشد. بدین ترتیب برای اینکه قابلیت و کارایی برنامه را اندکی افزایش دهیم، می‌توانیم از تابعی با نام is _ array ( ) جهت اطمینان از مطلب فوق استفاده کنیم. تابع is _ array ( ) جهت اجرا به یک آرگومان نیاز دارد. چنانچه این آرگومان یک آرایه ارزیابی شود، تابع فوق مقدار true و در غیر این‌صورت مقدار false را باز می‌گرداند. روش دیگر این است که به محض ایجاد متغیر $val در خط ۲۹ با استفاده از روش escaping آن‌را به یک آرایه تبدیل کنیم. بدین ترتیب می‌توانیم اطمینان حاصل کنیم که متغیر فوق علیرغم آنچه که در ابتدا معرفی شده است، همواره شامل یک آرایه می‌باشد. عبارت زیر چگونگی انجام این کار را نشان می‌دهد:

```
$val = (array) $val ;
```

آرایه‌ها

تا بدین‌جا با چگونگی ذخیره مقادیر در آرایه‌ها و نحوه دستیابی به آنها آشنا شدید، اما لازم است بدانید که PHP4 توابعی دارد که امکان انجام عملیات متنوع‌تری را نسبت به ذخیره و بازیابی عناصر آرایه در اختیارمان قرار می‌دهد. اگر پیشتر با زبان perl برنامه نوشته باشید برخی از این توابع برای شما آشنا خواهند بود.

ترکیب دو آرایه با استفاده از تابع ( ) array - merge

تابع( ) array _ merge نام دو یا چند آرایه را به‌عنوان آرگومان پذیرفته و ترکیبی از تمام آنها را به‌عنوان حاصل عملیات باز می‌گرداند. در قطعه کد زیر ابتدا دو آرایه تعریف شده است. سپس با استفاده از این تابع، آرایه دوم به انتهای آرایه اول اضافه شده و در نهایت با استفاده از ساختار تکرار foreach مقادیر عناصر آرایه حاصل به نمایش درآمده است:

```
$first = array ( "a " ," b " ," c " ) ;
$second = array ( 1, 2, 3 ) ;
$third = array _ merge ($first, $second ) ;

 foreach ( $third as  $val ) {
    print  "$val < BR  >" ;
}
```

همان‌گونه که مشاهده می‌کنید، آرایه $third شامل یک کپی از عناصر هر دو آرایه $first و $second است. ساختار تکرار foreach مقادیر آرایه حاصل را که شامل عناصر 'a'، 'b'، 'c'، 1، 2 و 3 می‌باشد با قرار دادن نشانه < BR > مابین هریک از آنها بر روی صفحه مرورگر اینترنت به‌نمایش می‌گذارد.به‌خاطر داشته باشید که نسخه اصلی آرایه‌هایی که به تابع array _ merge( ) ارسال می‌شوند، دست‌نخورده باقی می‌ماند. اگر دو آرایه‌ای که به تابع مذکور ارسال می‌شوند، شاخصهای مشابهی را

مورد استفاده قرار داده باشند، همواره به یاد داشته باشید که عناصر دومین آرایه عناصر معادل آرایه اول را رونویسی خواهند کرد.

اضافه کردن توأم چندین مقدار به یک آرایه با استفاده از تابع ( ) array_ push

تابع ( )array_ push از این نظر که تعداد آرگومان‌های آن حین استفاده از آن مشخص می‌شود تابع ویژه‌ای محسوب می‌گردد. آرگومان اول این تابع نام یک آرایه است. آرگومان‌های بعدی (به هر تعداد که باشند) عناصری را مشخص می‌کنند که این تابع باید آنها را به آرایه تعیین شده در اولین آرگومان اضافه کند. توجه کنید که تابع مورد بحث برخلاف تابع ( ) array_ merge ، آرایه تعیین شده توسط اولین آرگومان خود را تغییر می‌دهد. تابع ( ) array_ push تعداد مجموع عناصر آرایه را به‌عنوان نتیجه عملیات به تابع فراخواننده باز می‌گرداند. اجازه دهید تا در این قسمت آرایه‌ای را با هم ایجاد کرده و مقادیری را به آن اضافه کنیم:

```
$first = array ( "a", "b", "c" ) ;
$total = array_ push ( $first, 1, 2, 3 ) ;

print  "There are $total elements in \$first < p >" ;
foreach ( $first as $val ) {
   print  "$val < BR >" ;
}
```

از آنجا که تابع ( ) array_ push تعداد مجموع عناصر آرایه حاصل را که اکنون دستخوش تغییر شده است باز می‌گرداند، می‌توانیم مقدار بازگشتی (در این‌جا عدد 6) را در متغیری ذخیره کرده و آن‌را در پنجره مرورگر اینترنت نمایش دهیم. اکنون آرایه $first علاوه بر مقادیر اصلی خود شامل مقادیری است که توسط تابع( ) array_ push به آن اضافه کردیم. همان‌گونه که ملاحظه می‌کنید، کلیه مقادیر آرایه حاصل با استفاده از ساختار تکرار foreach بر روی صفحه به نمایش گذاشته شده‌اند.

همچنین دقت کنید که هنگام نمایش دنباله کاراکتری " $first \ " از علامت \ استفاده کرده‌ایم. اگر چنانچه در قالب یک دنباله کاراکتری پیش از حروف یا اعداد از علامت دلار استفاده کنید، PHP سعی خواهد کرد تا مقدار آن متغیر را به جای نام آن، مورد استفاده قرار دهد. عبارت " $first \ " بدین معنی است که مایلیم به جای مقدار متغیر $first ، خود دنباله کاراکتری ' $first ' را در خروجی، نمایش دهیم. بنابراین جهت نمایش کاراکتر ویژه ' $ ' باید پیش از آن از علامت \ استفاده نماییم. در چنین وضعیتی PHP به‌جای تفسیر آن به‌سادگی خود کاراکتر را در خروجی نمایش خواهد داد. این فرآیند را معمولاً با عنوان گریز کاراکتر یا escaping می‌شناسیم.

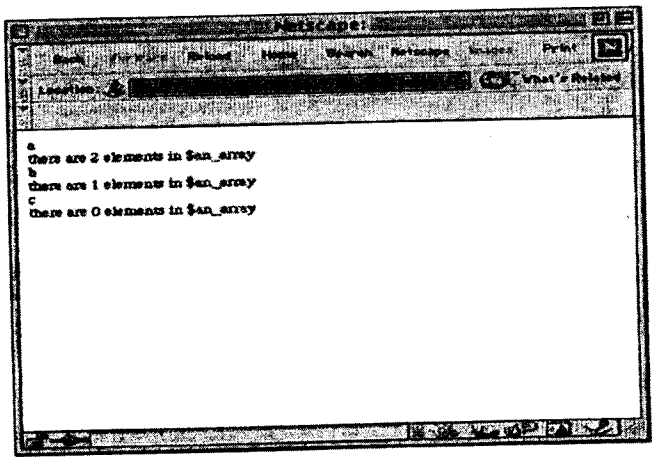
اگر پیشتر با زبان برنامه‌نویسی Perl برنامه می‌نوشتید، اکنون باید دقت زیادی را به‌خرج دهید. چنانچه در زبان برنامه‌نویسی Perl عادت به استفاده از تابع ( )push داشتید، توجه کنید که استفاده از یک آرایه به‌عنوان آرگومان دوم تابع ( )array_ push موجب ایجاد یک آرایه دو بعدی در PHP خواهد شد. در صورتی که قصد ترکیب دو آرایه را داشته باشید، از تابع ( ) array_ merge بهره بگیرید.

حذف اولین عنصر آرایه با استفاده از تابع ( ) array_ shift

تابع ( ) array_ shift به‌سادگی اولین عنصر از آرایه‌ای را که به‌عنوان آرگومان به این تابع ارسال شده است، حذف کرده و آن‌را به‌عنوان نتیجه تابع به برنامه فراخواننده باز می‌گرداند. در کد برنامه نمونه زیر از این تابع به‌همراه ساختار تکرار while استفاده شده است. همان‌گونه که ملاحظه می‌کنید با بررسی مقدار بازگشتی از تابع ( ) count قصد داریم تا از وجود عناصر در آرایه اطمینان حاصل کنیم:

```
< ? php
$an_array = array ( "a", "b", "c" ) ;
while (count ( $an_ array ) ) {
        $val = array_shift ( $an _ array ) ;
        print  "$val < BR >" ;
        print "there are". count ($an _ array ). "elements in \ $an- array< br>";
                \ $an _ array < br >" ;
}
? >
```

خروجی حاصل از اجرای این کد در شکل ۴- ۷ قابل بررسی است.



شکل ۴- ۷ استفاده از تابع ( ) array_shift جهت حذف و نمایش عنصری از یک آرایه

تابع ( ) array_ shift هنگامی مفید است که بخواهیم صفی از مقادیر تشکیل داده و تا زمان خالی شدن آن صف مقادیر آن‌را یکی یکی مورد پردازش قرار دهیم.

بازیابی بخشی از یک آرایه با استفاده از تابع ( ) array_ slice

به کمک تابع ( ) array _ slice می‌توان بخشی از یک آرایه را مورد بازیابی قرار داد. این تابع نام یک آرایه، موقعیت شروع و طول بخش مورد نظر را به‌عنوان آرگومان می‌پذیرد (دو مورد اول ضروری و مورد آخر اختیاری است). چنانچه طول‌بخش مورد نظر هنگام فراخوانی این تابع مشخص نگردد چنین فرض می‌شود که کلیه عناصر آرایه از نقطه شروع (آرگومان دوم) تا انتها مورد نظر بوده است. تابع ( ) array_ slice آرایه ارسال شده به آن را تغییر نمی‌دهد. حاصل عملیات این تابع آرایه جدیدی است که شامل عناصر مورد نظر می‌باشد.

در قطعه کد زیر ابتدا آرایه‌ای ایجاد شده و سپس یک آرایه جدید که شامل سه عنصر از آرایه اصلی است توسط تابع ( ) array_ slice تشکیل می‌گردد:

```
$first = array ( "a", "b", "c", "d", "e", "f " ) ;
$second = array_slice ( $first, 2, 3 );

foreach ( $second as $var ) {
        print  "$var < br > " ;
}
```

این قطعه کد به‌سادگی عناصر ' e ' ، ' d ' و ' c ' را در حالی‌که با استفاده از نشانه ‹ br › از یکدیگر تفکیک شده‌اند بر روی صفحه مرورگر اینترنت نمایش می‌دهد. دقت کنید که موقعیت شروع تعیین شده در تابع ( ) array_ slice نیز به‌عنوان یکی از عناصر مورد نظر بازیابی شده و به آرایه دوم انتقال یافته است. به بیان دیگر، عنصر [2] $first اولین عنصر آرایه $second می‌باشد.

اگر موقعیت شروع ارسالی به تابع ( ) array_slice (یعنی آرگومان دوم) عددی منفی باشد، فرآیند بازیابی به فاصله قدر مطلق این عدد از انتهای آرایه آغاز خواهد شد.

اگر طول آرایه مورد بازیابی (آرگومان سوم تابع ( ) array_slice) عددی منفی باشد، بخش بازیابی شده از آرایه اصلی از نقطه شروع (آرگومان دوم) آغاز شده و به تعداد قدر مطلق آن عدد منفی از انتهای آرایه اصلی، عناصر را شامل خواهد شد.

مرتب سازی عناصر آرایه

فرآیند مرتب سازی شاید مهم‌ترین عملیاتی باشد که بر روی عناصر یک آرایه انجام می‌شود. به‌واسطه توابع بسیار کارآمدی که در این رابطه در زبان PHP معرفی شده است، علاوه بر مرتب‌سازی استاندارد می‌توان فرآیند را به ترتیب دلخواه نیز انجام داد. در این قسمت توابعی را مورد بررسی و

استفاده قرار می‌دهیم که از آنها می‌توان جهت مرتب‌سازی آرایه‌های شاخص‌گذاری شده عددی و نیز آرایه‌های انجمنی استفاده کرد.

مرتب سازی آرایه‌های با شاخص عددی با استفاده از تابع ( ) sort

تابع ( ) sort آرایه‌ای را به‌عنوان آرگومان پذیرفته و عناصر آن‌را مرتب می‌کند. چنانچه عناصر آرایه از نوع دنباله‌های کاراکتری باشند، مرتب‌سازی از نوع الفبایی و در صورتی که عناصر مذکور از نوع عددی باشند، مرتب‌سازی از نوع عددی خواهد بود. این تابع هیچ‌نوع مقداری را به برنامه فراخواننده باز نمی‌گرداند اما آرایه ارسالی به آن‌را تغییر می‌دهد. دقت کنید که از این نظر تابع فوق با تابع همنام خود در زبان برنامه نویسی Perl تفاوت دارد. قطعه کد زیر ابتدا آرایه‌ای از کاراکترها را تعریف کرده و سپس با استفاده از تابع ( ) sort عناصر آن‌را مرتب می‌کند و در نهایت مقادیر به دست آمده را در خروجی نمایش می‌دهد:

```
$an_ array = array ( "a", "x", "f ", "c" ) ;
sort ( $an _ array ) ;
foreach ( $an _ array as $var ) {
        print  "$var < BR >" ;
}
```

> از ارسال آرایه‌های انجمنی به تابع ( ) sort ، جهت مرتب‌سازی عناصر خودداری کنید. با وجودی که این تابع قادر به مرتب‌سازی آرایه‌های انجمنی است اما باعث از دست‌رفتن مقادیر کلیدهای دستیابی شده و آنها را به‌سادگی با مقادیر عددی جایگزین می‌کند.

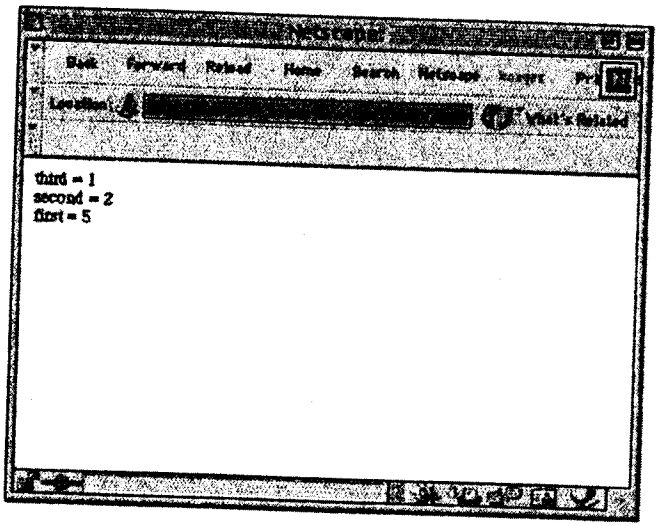
با استفاده از تابع دیگری با نام ( ) rsort می‌توان آرایه‌های با شاخص عددی را مشابه روش تابع ( ) sort به‌طور معکوس مرتب نمود.

مرتب‌سازی آرایه‌های انجمنی با استفاده از تابع ( ) asort (مرتب سازی برمبنای مقادیر):

تابع ( ) asort یک آرایه انجمنی را به‌عنوان آرگومان پذیرفته و عناصر آن‌را مشابه تابع ( )sort مرتب می‌کند. ضمن این فرآیند، کلیدهای دستیابی مقادیر خود را حفظ می‌کنند. به قطعه کد نمونه زیر توجه کنید:

```
$first = array ( "first" ⇒ 5, "second" ⇒ 2, "third" ⇒ 1 ) ;
asort ( $first ) ;
foreach ( $first as $key ⇒ $val ) {
        print $key = $val < BR >" ;
}
```

همان‌گونه که ملاحظه می‌کنید در قطعه کد بالا ابتدا آرایه‌ای انجمنی تعریف و مقداردهی شده و سپس با استفاده از تابع ( ) asort ضمن حفظ کلیدهای دستیابی اقدام به مرتب‌سازی عناصر آن شده است. خروجی حاصل از اجرای این قطعه کد در شکل ۵- ۷ قابل بررسی است.



شکل ۵- ۷ مرتب‌سازی عناصر یک آرایه انجمنی بر مبنای مقادیر آنها بااستفاده از تابع( ) asort

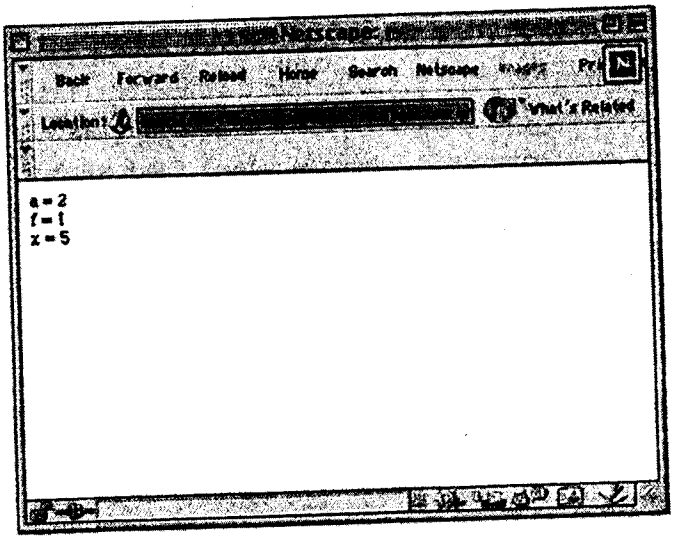
با استفاده از تابع دیگری با عنوان ( ) arsort می‌توان آرایه‌های انجمنی را مشابه روش تابع ( ) asort به‌طور معکوس مرتب نمود.

مرتب سازی آرایه‌های انجمنی با استفاده از تابع ( ) ksort (مرتب سازی بر مبنای کلیدهای دستیابی):

تابع ( ) ksort یک آرایه انجمنی را به‌عنوان آرگومان پذیرفته و عناصر آن‌را بر مبنای کلیدهای دستیابی مرتب می‌کند. بار دیگر، آرایه ارسالی به تابع دستخوش تغییر شده و هیچ‌گونه مقداری نیز به برنامه فراخواننده باز گردانده نمی‌شود. قطعه کد زیر نمونه‌ای از این فرآیند را نشان می‌دهد:

```
$first = array ( "x" ⇒ 5, "a" ⇒ 2, "f " ⇒ 1 ) ;
ksort ( $first ) ;
foreach ( $first as $key ⇒ $val ) {
        print  "$key = $val < BR > " ;
}
```

خروجی حاصل از اجرای این قطعه کد در شکل ۶- ۷ قابل بررسی است.



شکل ۶- ۷ مرتب‌سازی عناصر یک آرایه انجمنی بر مبنای کلیدهای دستیابی با استفاده از تابع ksort ( )

با استفاده از تابع دیگری با عنوان krsort ( ) می‌توان آرایه‌های انجمنی را مشابه روش تابع ksort ( ) به‌طور معکوس مرتب نمود.

نگاهی دیگر به توابع

اکنون با داشتن دید کافی و مناسب در مورد آرایه‌ها می‌توانیم به بررسی برخی از توابع سیستمی PHP که می‌توانند در ساخت توابع غیر سیستمی به‌طور کاملاً کارآمدی مؤثر باشند، بپردازیم. چنانچه قبلاً با زبان Perl برنامه نوشته باشید، به احتمال زیاد از این نکته مطلعید که می‌توان به‌سادگی زیر توابعی نوشت که تعداد آرگومان‌های آنها متغیر باشد. در زبان برنامه‌نویسی PHP4 امکانات قابل ملاحظه‌ای برای انجام این کار درنظر گرفته شده است.

فرض کنید تابعی ایجاد کرده‌اید که جهت اجرا تعداد دو آرگومان از نوع عددی را دریافت و یک دنباله کاراکتری را به عنوان نتیجه عملیات به برنامه فراخواننده باز می‌گرداند. دنباله کاراکتری حاصل شامل هر دو آرگومان ورودی است که در قالب یک جدول HTML در خروجی به نمایش در می‌آید. سطر آخرین جدول مجموع دو مقدار ورودی را به نمایش می‌گذارد:

```
Function addNums ( $num 1, $num 2 ) {
        $result = $num 1 + $num 2 ;
        $ret = " < table border = \" 1 \" >" ;
        $ret .= " < tr > < td > number 1 : < / td > < td > $num1  < / td > < / tr > " ;
        $ret .= " < tr > < td > number 2 : < / td > < td > $num2  < / td > < / tr > " ;
```

```
        $ret .= " < tr > < td > result :  < / td > < td > $result  < / td > < / tr > " ;
        $ret .= " < / table > " ;
        return $ret ;
}
print addNums ( 49 , 60 ) ;
```

این تابع بسیار ساده بوده و عملکرد آن به‌اندازه کافی مناسب می‌باشد، اما از انعطاف خوبی برخوردار نیست. حالتی را تصور کنید که مایل به پردازش بیش از دو آرگومان توسط این تابع هستید. اولین وساده‌ترین روشی که در این مورد به ذهن می‌رسد به احتمال قوی این است که ابتدا مقادیر مورد نظر را در یک آرایه ذخیره کرده و سپس به‌جای ارسال دو یا چند مقدار مجزا به‌تابع، آرایه حاصل را به‌عنوان آرگومان جهت پردازش به تابع ارسال کنیم. این گفته بدان معنی است که بخش اعظم کد مربوط به این تابع و همچنین به احتمال قوی بخشهایی از برنامه اصلی دستخوش تغییر خواهند شد. از این‌رو راه حل فوق آن‌گونه که در ابتدا به‌نظر می‌رسید از کارایی لازم برخوردار نیست. راه‌حل بهتر در این‌گونه موارد این است که تابع را به گونه‌ای تغییر دهیم که تعداد متغیری آرگومان عددی بپذیرد.

ابزار انجام چنین کاری در PHP4 استفاده از دو تابع ( ) func_num_ args و func_ get_ arg( ) است. تابع ( ) func_ get *args* *تعداد آرگومان‌های ارسال شده به تابعی را که در آن فراخوانی می‌شود، باز می‌گرداند.* *تابع* ( ) get _ arg *func* جهت اجرا به یک عدد صحیح به‌عنوان آرگومان نیاز دارد. این عدد صحیح بیانگر شاخص آرگومان مورد نظر است. تابع فوق مقدار این آرگومان را باز می‌گرداند. همانند آرایه‌ها، در اینجا نیز آرگومان‌های تابع از عدد صفر شروع به شاخص‌گذاری می‌شوند. از این‌رو جهت مشاهده مقدار اولین آرگومان ارسالی به یک تابع از عبارت زیر استفاده خواهیم کرد:

```
func_ get_ arg ( 0 ) ;
```

این وظیفه برنامه‌نویس است که از صحت آرگومان ارسالی به تابع ( ) get _ arg func_ از این‌نظر که در محدوده تعداد آرگومان‌های ارسالی به تابع مورد بررسی قرار دارد یا خیر، اطلاع حاصل کند. چنانچه مقدار ارسالی به تابع ( ) func_ get _arg خارج از محدوده باشد، این تابع مقدار false را باز گردانده و ضمناً موجب تولید خطا خواهد شد. اکنون می‌توانیم تابع ( ) addNums را به‌صورت زیر بازنویسی کنیم:

```
Function addNums ( ) {
        $ret = " < table border = \" 1 \" > " ;
        for ( $x = 0 ; $x < func_ num _ args ( ) ; $x + + ) {
                $arg = func_ get _ arg ( $x ) ;
                $result + = $arg ;
                $ret . = " < tr > <td > number ". ( $x + 1 ).
                    " : < / td > < td > $arg < / td > < / tr >" ;
        }
        $ret . = "< tr > < td > result : < / td > < td > $result < / td > < / tr >" ;
        $ret . = "< / table >" ;
        return $ret ;
```

```
}
print addNums ( 49, 60, 44, 22, 55) ;
```

همان‌گونه که مشاهده می‌کنید، هنگام تعریف تابع تعداد آرگومان‌ها را مشخص نکردیم. در عوض از یک ساختار تکرار for جهت دستیابی به هر یک از آرگومان‌ها استفاده کرده‌ایم. این ساختار دقیقاً به تعداد دفعات مورد انتظار اجرا می‌شود چراکه حد بالای آن توسط تابع func_ num_ args ( ) تعیین می‌شود (لازم به یادآوری است که تابع مذکور تعداد آرگومان‌های تابعی را که آن را فراخوانی می‌کند، باز می‌گرداند. تابع فوق جهت اجرا به هیچ آرگومانی نیاز ندارد).

حال ممکن است این پرسش به ذهنتان خطور کند که اگر در این قسمت صحبتی از آرایه‌ها به میان نیامد، پس چرا مورد فوق در بخش مربوط به آرایه‌ها مورد بحث و بررسی قرار گرفت؟ پیش از هرچیز، روشی که آرگومان‌ها در تابع مورد بحث، یعنی func_ get_ arg ( ) شاخص‌گذاری می‌شوند بسیار شبیه به آرایه‌هاست. علاوه بر این تابع دیگری در همین رابطه وجود دارد که هنوز آن‌را مورد بررسی قرار نداده‌ایم. نام این تابع func_ get _ args ( ) است. این تابع جهت اجرا به آرگومان نیاز ندارد اما مقدار بازگشتی حاصل از عملیات آن آرایه‌ای است که تمام آرگومان‌های تابع فراخوانده‌اش را شامل می‌شود. این بدان معنی است که راه‌حل قبلی را به‌صورت بهتری نیز می‌توانیم پیاده‌سازی نماییم:

```
function addNums ( ) {
        $args = func_ get _ args ( ) ;
        $ret = "< table  border = \" 1 \" >" ;
        foreach ( $args as $key ⇒ $val ) {
                $result + = $val ;
                $ret . = "< tr > < td > number " . ( $key + 1 ).
                        "< / td > < td > $val < / td > < / tr > " ;
        }
        $ret . ="< tr > < td > result : < / td > < td > $result < / td > < / tr > " ;
        $ret . = " < / table > " ;
        return  $ret ;
}
print addNums ( 49, 60, 44, 22, 55 ) ;
```

همان‌گونه که ملاحظه می‌کنید این بار به جای اینکه هر بار تنها یکی از آرگومان‌های تابع را مورد دستیابی قرار دهیم به‌سادگی همگی آنها را در آرایه‌ای با نام $args ذخیره کرده و آرایه را در یک ساختار تکرار مورد پردازش قرار داده‌ایم.

جمع‌بندی

در درس این ساعت مطالب مفیدی درباره آرایه‌ها و برخی از ابزارهای کارآمد PHP4 که جهت کار با آرایه‌ها طراحی شده‌اند را مورد بررسی قرار دادیم. اکنون باید بتوانید هر دو نوع آرایه‌ها، یعنی آرایه‌هایی که از طریق مقادیر عددی شاخص‌گذاری شده‌اند و نیز آرایه‌هایی که از طریق دنباله‌های

کاراکتری شاخص‌گذاری شده‌اند را ایجاد نموده و با استفاده از ساختار تکرار foreach مقادیر موجود در عناصر آنها را نمایش دهید..

هم‌اینک با مطالبی که در این درس فراگرفتید، می‌توانید آرایه‌های با ابعاد کمتر را جهت ایجاد آرایه‌های چند بعدی با یکدیگر ترکیب کرده و از طریق ساختارهای تکرار تودرتو مقادیر آرایه‌های چند بعدی حاصل را مورد پردازش قرار دهید. در درس این ساعت مطالب مفیدی راجع به دستکاری آرایه‌ها، مانند اضافه کردن مقادیر جدید یا حذف عناصر موجود فراگرفته و برخی از تکنیکهای مربوط به مرتب‌سازی آرایه‌ها را مورد بررسی قرار دادید. در قسمت آخر این درس نیز مطالب ارزنده‌ای درباره توابعی که از شاخص‌گذاری عددی به روش آرایه‌ها استفاده می‌کنند و می‌توانیم از آنها جهت افزایش قابلیت توابع خود استفاده کنیم، فراگرفتید.

دردرس ساعت هشتم با بررسی چگونگی پشتیبانی از اشیا در زبان PHP، بررسی مطالب مربوط به‌اصول PHP را تکمیل خواهیم کرد. در حال حاضر توسعه کتابخانه‌های جدید و گسترش کتابخانه‌های موجود با استفاده از کلاس‌ها و اشیا یکی از مباحث داغ در زمینه PHP است. لذا از اهمیت خاصی جهت مطالعه برخوردار است.

پرسش و پاسخ

پرسش: اگر ساختار تکرار foreach در PHP4 معرفی شده است، بنابراین برنامه‌نویسان PHP3 چگونه عناصر موجود در آرایه‌ها را مورد پردازش قرار می‌دادند؟

پاسخ: تکنیک مورد استفاده در PHP3 جهت پردازش آرایه‌ها استفاده از تابع ( ) each بود که اغلب به‌همراه ساختار تکرار while مورد بهره‌برداری قرار می‌گرفت. در درس ساعت شانزدهم مطالب بیشتری در رابطه با این تکنیک فراخواهید گرفت.

پرسش: آیا تابعی جهت دستکاری آرایه‌ها باقی‌مانده است که در درس این ساعت مورد بحث قرار نگرفته باشد؟

پاسخ: زبان برنامه نویسی PHP4 توابع بسیاری جهت کار با آرایه‌ها دارد. در درس ساعت شانزدهم درباره برخی از توابع فوق که در درس این ساعت مورد بررسی قرار نگرفتند، بحث خواهیم کرد. علاوه بر این اسناد رسمی PHP به آدرس زیر همواره راهنمای خوبی در این مورد است:
http: // www. php. net / manual / ref. array. php

پرسش: اگر بتوان به‌طریقی از تعداد عناصر یک آرایه مطلع شد، آیا می‌توان با استفاده از ساختار تکرار for عناصر موجود در آرایه را مورد پردازش قرار داد؟

پاسخ: استفاده از این روش باید همراه با دقت فراوانی باشد. مطلقاً نمی‌توان از این موضوع اطمینان حاصل کرد که آیا در مورد آرایه‌های با شاخص عددی ترتیب اعداد رعایت شده‌است یا خیر.

تمرینها

هدف از این بخش دوره مطالب فراگیری شده در قالب آزمون است. پاسخ بخش آزمون بلافاصله بعد از آن آمده است. بخش فعالیتها شامل تمرینهایی است که به منظور افزایش قابلیت برنامه‌نویسی خواننده طراحی شده و البته فاقد پاسخ است.

آزمون

۱- با استفاده از کدام تابع سازنده می‌توان اقدام به ایجاد و تعریف یک تابع کرد؟

۲- شاخص آخرین عنصر موجود در آرایه زیر چیست؟

```
$users = array ( "Harry", "Bob", "Sandy" ) ;
```

۳- بدون استفاده از یک تابع، ساده‌ترین روش برای اضافه کردن عنصر " Susan " را به آرایه ایجاد شده در تمرین قبل بیان نمایید.

۴- با استفاده از کدام تابع می‌توان دنباله کاراکتری " Susan " را به‌عنوان یک عنصر جدید به آرایه تمرین ۲ اضافه کرد؟

۵- چگونه می‌توان از تعداد عناصر یک آرایه اطلاع حاصل کرد؟

۶- ساده‌ترین روش در زبان PHP4 جهت پردازش تمامی عناصر یک آرایه چیست؟

۷- از کدام تابع می‌توان جهت ترکیب عناصر دو آرایه و ایجاد یک آرایه جدید استفاده کرد؟

۸- چگونه می‌توان عناصر یک آرایه انجمنی را بر مبنای مقادیر کلیدهای دستیابی مرتب کرد؟

پاسخ آزمون

۱- به کمک تابع سازنده ( ) array می‌توان آرایه‌های جدیدی ایجاد کرد.

۲- آخرین عنصر این آرایه عبارت از [2]$users است. به‌خاطر داشته باشید که شاخص‌گذاری آرایه همواره از عدد صفر آغاز می‌شود.

۳-

```
$users [ ] = " Susan " ;
```

۴-

```
array_ push ($users, " Susan " ) ;
```

۵- تابع ( ) count همواره تعداد عناصر یک آرایه را باز می‌گرداند.

۶- استفاده از ساختار foreach، ساده‌ترین روش درPHP جهت پردازش عناصر یک آرایه است.

۷- بااستفاده از تابع( ) array_ merge، می‌توان عناصر دو آرایه را جهت ایجاد یک آرایه جدید با یکدیگر ترکیب نمود.

۸- با استفاده از تابع ( ) ksort، می‌توان آرایه‌های انجمنی را بر مبنای کلیدهای دسترسی مرتب نمود.

فعالیتها

۱- آرایه‌ای چند بعدی از فیلمهای سینمایی ایجاد کرده و برحسب نوع فیلم آنها را سازماندهی کنید. به‌عنوان راهنمایی از یک آرایه انجمنی استفاده کنید که نوع فیلم کلید دسترسی آن را تشکیل دهد (مانند " SF "، " Action "، " Romance " و غیره). هر یک از مقادیر این آرایه باید خود، آرایه‌ای از فیلمهایی از آن نوع باشد (مانند "2001 "، " Alien "، " Terminator " و غیره).

۲- آرایه ایجاد شده در تمرین قبل را مورد پردازش قرار دهید، بدین ترتیب که انواع فیلمها و اسامی فیلمهای هر نوع را بر روی صفحه مرورگر اینترنت نمایش دهید.

ساعت هشتم

اشیا

برنامه‌نویسی به شیوه شئ‌گرا (Object - Oriented) اقدام جسورانه و خطرناکی است. این شیوه نحوه تفکر شما را در مورد برنامه‌نویسی تغییرمی‌دهد اما به‌محض درک این شیوه برنامه‌نویسی، بسیار بعیداست که آن‌را کنار بگذارید. زبان برنامه‌نویسی PHP و پیش از آن زبان Perl به‌طور جامعی ویژگی‌ها و مشخصات شیوه برنامه‌نویسی شئ‌گرا را در دستور زبان و همچنین ساختارهای مختلف برنامه مورد پشتیبانی قرار داده‌اند. با معرفی PHP4 اکنون می‌توان پروژه‌های برنامه‌نویسی را با پشتیبانی جامع‌تری از شیوه مذکور به انجام رسانید.

در درس این ساعت ضمن بررسی ویژگی‌های پشتیبانی شده از شیوه برنامه‌نویسی شئ‌گرا در PHP، از این شیوه برای نوشتن برنامه‌های نمونه استفاده خواهیم نمود. در این درس با مباحث زیر آشنا می‌شوید:

- **ماهیت کلاسها و اشیا**
- **چگونگی ایجاد کلاسها و نمونه‌گیری از آنها**
- **چگونگی تعریف و دستیابی به خصوصیات و متدها**
- **چگونگی بهره‌گیری از ویژگی ارث‌بری جهت بهره‌مند شدن از قابلیتهای یک کلاس در سایر کلاسها**
- **چگونگی کسب اطلاع در مورد اشیای موجود در برنامه**
- **چگونگی ذخیره اشیا در قالب دنباله‌های کاراکتری جهت نگهداری در یک فایل یا بانک اطلاعاتی**
- **بررسی برخی از دلایل مبنی بر فواید برنامه‌نویسی به شیوه شئ‌گرا در سازماندهی پروژه‌های برنامه‌نویسی**

در ادامه به بررسی موارد فوق خواهیم پرداخت .

ماهیت اشیا

شئ به بسته‌ای شامل متغیرها و توابع گفته می‌شود که از یک الگوی خاص موسوم به کلاس برگرفته شده است. اشیا به مقدار زیادی جزئیات داخلی و نحوه عملکرد خود را از برنامه‌هایی که آنها را مورد استفاده قرار می‌دهند، پنهان می‌کنند و در مقابل رابطهایی را در اختیار برنامه‌های استفاده کننده قرار می‌دهند تا از طریق آنها برنامه‌ها بتوانند فرمانهای مورد نظرشان را ارسال کرده و نتایج عملیات را دریافت کنند. این رابطها توابع ویژه‌ای هستند که معمولاً از واژه متداول " متد " جهت اشاره به آنها استفاده می‌شود. تمامی متدهای یک شئ دسترسی تمام عیاری به متغیرهای ویژه‌ای از آن شئ که معمولاً از واژه " خصوصیت " جهت اشاره به آنها استفاده می‌شود، دارند.

با تعریف یک کلاس در حقیقت مجموعه‌ای از مشخصه‌ها تعریف می‌شود و با ایجاد اشیا موجودیتهایی پا به عرصه می‌گذارند که این مشخصه‌ها را مشترکاً مورد استفاده قرار می‌دهند اما هر یک از این موجودیتها می‌توانند مقادیر خصوصیات را آن‌گونه که کد مربوطه دیکته می‌کند، مقداردهی کنند. برای مثال ممکن است روزی کلاسی با نام automobile ایجاد کنید. این کلاس به احتمال زیاد مشخصه‌ای با نام color خواهد داشت. از این‌رو تمامی اشیایی که از نوع این کلاس تعریف می‌شوند (تمامی اتومبیلها) این مشخصه را به اشتراک خواهند گذاشت. اما برخی ممکن است آن‌را با مقدار "blue" و برخی با مقدار "green" و برخی نیز با سایر مقادیر مقداردهی کنند.

شاید بتوان ویژگی " قابلیت استفاده مجدد" را مهم‌ترین فایده برنامه‌نویسی به شیوه شئ‌گرا دانست. از آنجا که کلاسهای مورد استفاده جهت ایجاد اشیا موجودیتهای مستقلی هستند، می‌توان به‌آسانی آنها را از یک پروژه برنامه‌نویسی به پروژه دیگر مورد استفاده قرار داد. علاوه بر این می‌توان کلاس‌های فرزندی ایجاد کرد که برخی از مشخصه‌های کلاس پدر خود را به ارث برده یا به‌کل آنها را تغییر دهد. به‌کمک این تکنیک می‌توان کلاسهای پیچیده‌تر و اشیای تخصصی‌تری ایجاد کرد که ضمن دارا بودن عملکردهای پایه، عملکردها و قابلیتهای دیگری را نیز خود به آنها اضافه کنند.

احتمالاً مهم‌ترین و بهترین روش توضیح برنامه‌نویسی به شیوه شئ‌گرا بررسی برنامه‌های نمونه است.

روند ایجاد یک شئ

برای ایجاد یک شئ ابتدا باید الگویی را تعریف کنید که شئ مورد نظرتان از آن نمونه‌گیری خواهد شد. به این الگو آن‌چنانکه در فرهنگ واژه‌های برنامه‌نویسی به شیوه شئ‌گرا مرسوم است، واژه

کلاس گفته می‌شود. در زبان PHP4 برای تعریف کلاس از واژه کلیدی class استفاده می‌کنیم. شکل کلی تعریف یک کلاس به‌صورت زیر است:

```
Class first _ class {
    // a very minimal class
}
```

هم اکنون با در دست داشتن کلاس first _ class می‌توانیم به تعداد دلخواه اشیایی از نوع first _ class را از این کلاس نمونه‌گیری کنیم. در زبان PHP4 جهت نمونه‌گیری شئ از یک کلاس از واژه کلیدی new استفاده می‌کنیم. به نمونه‌های ایجاد شده در زیر توجه کنید:

```
$obj1 = new  first _ class ( ) ;
$obj2 = new  first _ class ( ) ;
print " \ $obj 1 is a " . gettype ( $obj 1) . "< br >" ;
print " \ $obj 2 is a " . gettype ( $obj 2) . "< br >" ;
```

با استفاده از تابع ویژه‌ای در PHP با عنوان gettype ( )، می‌توانیم اطمینان حاصل کنیم که متغیرهای $obj 1 و $obj 2 شامل اشیایی از نوع کلاس first _ class هستند. تابع gettype ( ) به عنوان آرگومان نام یک متغیر را دریافت کرده و دنباله کاراکتری باز می‌گرداند که نوع داده آن متغیر را مشخص می‌نماید. در زبانهای برنامه‌نویسی چون PHP که قوانین سفت و سختی در مورد نوع داده‌ها وجود ندارد (به این‌گونه زبانها اصطلاحاً loosely _ typed گفته می‌شود)، توابعی چون gettype ( ) هنگام بررسی نوع داده آرگومانهای ارسالی به توابع بسیار مفید واقع می‌شوند. در قطعه کد فوق، تابع gettype ( ) دنباله کاراکتری "object" را باز می‌گرداند و مقدار مذکور آن چیزی است که بر روی صفحه مرورگر اینترنت به‌عنوان خروجی نمایش می‌یابد.

با مشاهده خروجی حاصل از اجرای این قطعه کد، قانع خواهید شد که دو عدد شئ ایجاد شده‌است. هرچند که این دو شئ احتمالاً فایده چندانی ندارند اما نکته مهمی را آشکار می‌کنند. می‌توان کلاس را به‌منزله قالبی تصور کرد که با به‌کارگیری آن می‌توان اشیای متعددی را به تعداد دلخواه به‌وجود آورد. اجازه دهید تا ویژگی‌های بیشتری را به ساختار کلاس‌هایمان اضافه کنیم. در این صورت می‌توانیم اشیایی با قابلیت عملکرد بهتر به‌دست آوریم.

خصوصیات شئ

اشیا به متغیرهای خاصی که آنها را خصوصیت می‌نامیم، دسترسی دارند. این‌گونه متغیرها را می‌توان در هر جایی از کلاس تعریف نمود، اما معمولاً جهت افزایش خوانایی و کد مربوطه آنها را در ابتدای هر کلاس تعریف می‌کنند. خصوصیت ممکن است یک متغیر، آرایه و حتی یک شئ دیگر باشد، به تعریف کلاس زیر توجه کنید:

```
Class  first _ class
    {
    var  $name = " harry " ;
}
```

دقت کنید که متغیر (خصوصیت) مورد نظرمان را با استفاده از واژه کلید var تعریف کرده‌ایم. این امر در مورد تعریف متغیرهای کلاس ضروری بوده و در صورت فراموش کردن آن با پیغام خطایی در همین رابطه مواجه خواهید شد. با تعریف فوق هر نمونه (شئ) از کلاس first _ class دارای خصوصیتی با نام $name و مقدار " harry "خواهد بود. دستیابی و حتی تغییر مقدار این متغیر از خارج شئ امکان‌پذیر است. قطعه کد زیر چگونگی انجام این فرآیند را نشان می‌دهد:

```
Class first _ class {
   Var  $name = " harry " ;
}
$obj 1 = new first_ class ( ) ;
$obj 2 = new first _ class ( ) ;
$obj 1 → name = " bob " ;
print  $obj 1 → name < BR >" ;
print  $obj 2  → name < BR >" ;
```

همان‌گونه که مشاهده می‌کنید، عملگر → امکان دستیابی و تغییر خصوصیتهای یک شئ را در اختیارمان قرار می‌دهد. با وجودی که مقدار خصوصیت $name هر دو شئ $obj 1 و $obj 2 در ابتدا برابر با " harry " بوده، با استفاده از عملگر مذکور توانستیم این مقدار را در مورد شئ $obj 1 با دنباله کاراکتری " bob " تعویض کنیم. در انتها مجدداً با استفاده از عملگر → مقدار خصوصیت هر دو شئ را بر روی صفحه مرورگر اینترنت به نمایش گذاشته‌ایم.

> زبان‌های شئ‌گرا مانند زبان برنامه‌نویسی Java همواره بر لزوم تعریف سطحی از دسترسی به خصوصیات و متدهای اشیا تاکید می‌کنند. این بدان معنی است که سطح دسترسی می‌تواند تنها به ویژگی‌های مورد نیاز جهت استفاده هرچه کارآمدتر اشیا محدود شود. در مورد خصوصیات، این بدان معناست که جهت امنیت بیشتر تنها در درون کلاس بتوان آنها را مورد دستیابی قرار داد. در زبان PHP چنین حفاظتی وجود ندارد. به‌عبارت دیگر، دستیابی به تمامی خصوصیات شئ کاملاً بدون محدودیت صورت می‌گیرد. این وضعیت در مواردی که مقادیر آنها به‌دقت دستخوش تغییر نشود، موجب مشکلات بزرگی می‌گردد.

از اشیا می‌توان جهت ذخیره اطلاعات نیز استفاده کرد. ذخیره اطلاعات در اشیا جالب توجه‌تر از ذخیره آنها در آرایه‌های انجمنی است. در قسمت بعد به بررسی متدهای یک شئ می‌پردازیم، یعنی بخشی از شئ که توانایی انجام عملیات مورد نظر را (که در کلاس مرجع تعریف شده است) دارد.

متدهای شئ

واژه متد به تابع تعریف شده در درون یک کلاس اطلاق می‌شود. تمامی اشیایی که از یک کلاس نمونه‌گیری می‌شوند، دارای متدها و بنابراین قابلیتهای کلاس مورد نظر خواهند بود. برنامه لیست ۱- ۸ در خط ۷، تعریف متدی را به تعریف کلاس اضافه کرده است.

```
 1: <html>
 2: <head>
 3: <title>Listing 8.1</title>
 4: <body>
 5: <?php
 6: class first_class {
 7:     function sayHello() {
 8:         print "hello";
 9:     }
10: }
11:
12: $obj1 = new first_class();
13: $obj1->sayHello();
14: // outputs "hello"
15: ?>
16: </body>
17: </html>
```

لیست ۱- ۸ تعریف کلاسی شامل یک متد

همان‌گونه که به احتمال زیاد متوجه شده‌اید متدها هم از لحاظ ظاهر و هم از لحاظ رفتار عملکردی مشابه عملکرد توابع معمولی دارند. با این حال جایگاه همیشگی متدها تنها در درون کلاسهاست. همانند خصوصیات، با استفاده از عملگر → می‌توان متدها را مورد دستیابی (فراخوانی) قرار داد. نکته مهم‌تر این است که متدها به متغیرها عضو (خصوصیات) کلاس دسترسی دارند. پیشتر مثالی را در مورد دستیابی به یک خصوصیت از خارج کلاس مشاهده کردید. اما سؤال این است که آیا می‌توان ترتیبی داد که اشیا خودشان را مورد دستیابی قرار دهند؟ لیست ۲- ۸ را در نظر بگیرید.

```
 1: <html>
 2: <head>
 3: <title>Listing 8.2</title>
 4: <body>
 5: <?php
 6: class first_class {
 7:     var $name="harry";
 8:     function sayHello() {
 9:         print "hello my name is $this->name<BR>";
10:     }
11: }
12:
13: $obj1 = new first_class();
```

لیست ۲- ۸ دستیابی به یک خصوصیت از درون یک متد

```
14: $obj1->sayHello();
15: // outputs "hello my name is harry"
16: ?>
17: </body>
18: </html>
```

دنباله لیست ۲- ۸

همان‌گونه که ملاحظه می‌کنید، با استفاده از متغیر ویژه‌ای با نام $this می‌توان از داخل یک شئ به خود آن اشاره نمود. این نوع دسترسی در خط ۹ از برنامه لیست فوق مشهود است. متغیر $this در دنیای برنامه نویسی را می‌توان به ضمایر شخصی در ادبیات تشبیه کرد. هرچند که برای دستیابی به یک شئ باید از نام متغیری که شئ مورد نظر به آن نسبت داده شده است، استفاده کرد (برای مثال $obj 1)، برای دستیابی یک شئ به خودش استفاده از متغیر ویژه $this ضروری است. با ترکیب متغیر $this و عملگر → می‌توان از درون یک شئ به تمام خصوصیات و متدهای آن شئ دسترسی پیدا کرد.

وضعیتی را تصور کنید که لازم است تا مقادیر مختلفی را به خصوصیت name نمونه‌های گرفته شده از کلاس مرجع first_class نسبت دهید. همان‌گونه که پیشتر نیز این کار را انجام دادید، یکی از روشها این است که به‌طور دستی مقدار خصوصیت name را مشخص کنیم. برنامه لیست ۳- ۸ روش دیگری را نشان می‌دهد که در آن جهت انجام فرآیند مورد نظر از یک متد استفاده کرده‌ایم (خط ۱۰ تا ۱۲ را ملاحظه کنید).

```
1: <html>
2: <head>
3: <title>Listing 8.3</title>
4: </head>
5: <body>
6: <?php
7: class first_class {
8:     var $name="harry";
9:
10:    function setName( $n ) {
11:        $this->name = $n;
12:    }
13:
14:    function sayHello() {
15:        print "hello my name is $this->name<BR>";
16:    }
17: }
18:
19:
20: $obj1 = new first_class();
21: $obj1->setName("william");
22: $obj1->sayHello();
23: // outputs "hello my name is william"
24: ?>
25: </body>
26: </html>
```

لیست ۳- ۸ تغییر مقدار یک خصوصیت از طریق متدی از همان شئ

همان‌گونه که مشاهده می‌کنید مقدار اولیه خصوصیت name در این برنامه برابر با دنباله کاراکتری " harry " است. اما با فراخوانی متد ( ) setName از شئ مورد نظر در خط ۱۰ این مقدار به دنباله کاراکتری " william " تغییر می‌کند. توجه کنید که شئ مورد نظر چگونه قادر به تعیین مقادیر خصوصیات خود است. همچنین توجه داشته باشید که دقیقاً مشابه آنچه که در مورد توابع عادی شاهد آن بودیم، می‌توانیم آرگومانهایی را جهت پردازش به متدهای یک شئ ارسال کنیم (مانند ارسال دنباله کاراکتری " william " به متد ( ) setName در خط ۲۱ برنامه).

توضیح یک نکته بسیار مهم را تا بدین جا به تعویق انداختیم؛ اگر کلاس شما دارای متدی همنام با خود آن کلاس باشد (مانند متد ( ) first _ class در کلاس first _ class)، هنگامی که با استفاده از واژه کلیدی new اقدام به ایجاد نمونه‌ای از کلاس می‌کنید آن متد به‌خودی خود فراخوانی می‌گردد. به این گونه متدها " سازنده" یا constructor گفته می‌شود. از آنجا که طبق قاعده کلی، هر متدی می‌تواند یک یا چند آرگومان داشته باشد، می‌توان متدهای سازنده را نیز به‌گونه‌ای طراحی کرد که مقادیری را به‌عنوان آرگومان دریافت کنند. بدین سان می‌توان ترتیبی داد که فرآیند نمونه‌گیری شئ از یک کلاس توأم با پردازش آن آرگومانها باشد. معمولاً مهم‌ترین فرآیندی که متدهای سازنده انجام می‌دهند تعیین مقادیر اولیه خصوصیات شئ مربوطه است. بدین ترتیب می‌توان چنین اظهار کرد که متدهای سازنده وضعیت ابتدایی اشیا را مشخص می‌کنند. برنامه لیست ۴- ۸ متد سازنده‌ای را به کلاس first _ class اضافه کرده است.

```
 1: <html>
 2: <head>
 3: <title>Listing 8.4</title>
 4: </head>
 5: <body>
 6: <?php
 7: class first_class {
 8:     var $name;
 9:     function first_class( $n="anon" ) {
10:         $this->name = $n;
11:     }
12:     function sayHello() {
13:         print "hello my name is $this->name<BR>";
14:     }
15: }
16:
17: $obj1 = new first_class("bob");
18: $obj2 = new first_class("harry");
19: $obj1->sayHello();
20: // outputs "hello my name is bob"
21: $obj2->sayHello();
22: // outputs "hello my name is harry"
23: ?>
24: </body>
25: </html>
```

لیست ۴- ۸ کلاسی شامل یک متد سازنده

متد سازنده first _ class ( ) در خط ۹ از این برنامه هنگام نمونه‌گیری از کلاس first _ class به‌طور خودکار فراخوانی می‌گردد. همان‌گونه که ملاحظه می‌کنید آرگومان متد سازنده فوق دارای مقدار پیش‌فرض می‌باشد. از این جهت در صورتی که هنگام نمونه‌گیری شئ از به‌کارگیری آرگومان متد سازنده خودداری کنیم مقدار پیش‌فرض، یا به عبارت دیگر دنباله کاراکتری "anon" مورد استفاده قرار خواهد گرفت.

بررسی یک مثال

در این قسمت از درس مثالی را بررسی می‌کنیم که در آن از تمام مطالب عنوان شده در این درس استفاده خواهیم کرد. این مثال نسبت به موارد مشابهی که تاکنون بررسی کردیم، مفیدتر خواهد بود. مثال ما شامل طراحی کلاسی است که جدولی از فیلدها را نگهداری کرده و آنها را بر مبنای نام ستونی که فیلد مورد نظر در زیر آن قرار می‌گیرد، مرتب می‌کند. این ساختار از اطلاعات باید بر مبنای روش سطر به سطر تشکیل شود. وجود متدی در این کلاس برای نوشتن مقادیر داده‌ها بر روی صفحه مرورگر اینترنت از ضروریات است. قالب‌بندی داده‌ها به شکل خوش‌طرح در این مرحله ضروری نیست.

تعریف خصوصیات کلاس

پیش از هرچیز باید تصمیم بگیریم که کلاس مورد نظر شامل چه خصوصیاتی خواهد بود. در طراحی خود مایلیم تا اسامی ستونها را در یک آرایه و داده‌های موجود در سطرها را در یک آرایه دو بعدی نگه‌داریم. همچنین از یک متغیر از نوع عددصحیح جهت اطلاع از تعداد ستونها استفاده خواهیم کرد. بدین ترتیب تعریف کلاس Table به شکل زیر خواهد بود:

```
class Table {
   var  $table_ array = array ( ) ;
   var  $headers = array ( ) ;
   var  $cols ;
}
```

ایجاد یک متد سازنده

باید به طریقی بتوانیم ستونهایی از جدول را که داده‌ها را در آن به نمایش خواهیم گذاشت، نام‌گذاری کنیم. این‌گونه کارها جزو آن دسته کارهایی است که متدهای سازنده جهت انجام آن طراحی می‌شوند. می‌توانیم آرایه‌ای از دنباله‌های کاراکتری را به یک متد سازنده از کلاس ارسال کنیم. با انجام این عمل متد مذکور می‌تواند تعداد ستونهای جدول (تعداد عناصر آرایه) را محاسبه کرده و نتیجه را در یک متغیر عددی نگه دارد. به تعریف متد سازنده Table ( ) توجه کنید:

```
function Table ( $headers ) {
```

```
    $this → headers = $headers ;
    $this → cols = count ($headers) ;
}
```

با فرض اینکه هنگام نمونه‌گیری یک شئ جدید از کلاس Table اطلاعات صحیحی به آن ارسال شده باشد، متد سازنده این کلاس قادر خواهد بود تا تعداد ستونهایی از جدول را که جهت ذخیره اطلاعات به آنها نیاز داریم و همچنین نام هر ستون از جدول را مورد دستیابی قرار دهد. به‌دلیل اینکه اطلاعات مذکور در قالب خصوصیات کلاس نگهداری می‌شوند، تمامی متدهای شئ مورد نظر قادر به دستیابی به آنها خواهند بود.

طراحی متد ( ) addRow

شئ نمونه‌گیری شده از کلاس Table هر یک از سطرهای شامل داده‌ها را در قالب یک آرایه می‌پذیرد. فرآیند فوق البته با این فرض انجام می‌شود که این اطلاعات منطبق بر ترتیب ستونها در اختیار قرار می‌گیرد. فرآیند فوق توسط متد ( ) addRow در شئ مورد نظر انجام می‌شود:

```
function addRow ( $row ) {
    If ( count ($row ) ! = $this → cols )
        Return false ;
    array _push ($this → table _ array, $row) ;
    return true ;
}
```

متد ( ) addRow جهت اجرا به آرگومانی نیاز دارد که در داخل متغیری با نام $row ذخیره شده است. همان‌گونه که ملاحظه می‌کنید، تعداد ستونهایی از جدول را که شئ مورد نظر قصد پردازش آن را دارد، در خصوصیت $cols از کلاس مربوطه ذخیره کرده‌ایم. بدین ترتیب با استفاده از تابع count ( ) می‌توانیم تعداد عناصر ذخیره شده در آرایه $row را تعیین نماییم. چنانچه تعداد این عناصر با تعداد ستونهای جدول معادل نباشد متد ( ) addRow مقدار false را باز می‌گرداند.

در صورتی که تعداد عناصر آرایه مورد بحث مطابق انتظار باشد، فراخوانی تابع سیستمی array_ ( ) push باعث می‌شود تا عناصر آرایه مذکور به خصوصیت table_ array اضافه شود. تابع سیستمی array_ push ( ) جهت اجرا به دو آرگومان نیاز دارد: آرایه‌ای جهت اضافه کردن و مقداری که آرایه فوق به آن اضافه می‌شود. اگر چنانچه آرگومان دوم خود یک آرایه باشد به‌عنوان یک عنصر منفرد به آرایه اول اضافه می‌شود و بدین ترتیب یک آرایه چند بعدی به دست می‌آید. بدین ترتیب می‌توان آرایه‌ای از آرایه‌ها را به‌سادگی ایجاد نمود.

طراحی متد ( ) addRowAssocArray

عملکرد متد ( ) addRow خوب و مطابق انتظار است منتها به این شرط که عناصر آرایه ارسالی

به آن دارای ترتیب شاخص‌گذاری صحیحی باشند. متد ( ) addRowAssocArray که در این قسمت معرفی می‌شود دارای انعطاف بیشتری است. این تابع به عنوان آرگومان یک آرایه انجمنی را دریافت می‌کند. نکته اینجاست که کلیدهای دسترسی هریک از مقادیر باید با اسامی تیترهای ذخیره شده در خصوصیت header شئ مربوطه معادل باشند، چه در غیر این‌صورت نتایج غیر قابل انتظاری به‌بار می‌آید. به تعریف این متد توجه کنید:

```
function addRowAssocArray ( $row_ assoc) {
   $row = array ( ) ;
   foreach ( $this → headers as $header) {
      if ( ! isset ($row_ assoc [$header] ) )
         $row_ assoc [$header] = " " ;
      $row [ ] =$row_ assoc [$header]  ;
   }
   array_ push ($this → table_ array, $row) ;
   return true ;
}
```

همان‌گونه که مشاهده می‌کنید آرایه انجمنی ارسال شده به متد ( ) addRowAssocArray در متغیری با عنوان $row_ assoc ذخیره می‌شود. جهت نگهداری مقادیری که در نهایت به خصوصیت table_ array اضافه خواهند شد، از یک آرایه تهی با نام $row استفاده شده است. همچنین به‌منظور اطلاع از یافتن معادل هر یک از دنباله‌های کاراکتری با مقادیر موجود در آرایه $row_assoc از یک حلقه تکرار استفاده شده است. برای انجام این کار تابع سیستمی ( ) isset در زبان PHP4 که متغیری را به‌عنوان آرگومان دریافت می‌کند ابزار مناسبی است. این تابع در صورتی که متغیر ارسالی به آن مقدار دهی شده باشد مقدار true و در غیر این‌صورت مقدار false را باز می‌گرداند. در هربار گذر از حلقه تکرار یکی از عناصر موجود در آرایه $row_ assoc که کلید دستیابی آن مقدار جاری خصوصیت headers از شئ مربوطه است به تابع سیستمی ( ) isset ارسال می‌گردد. اگر عنصری با کلید دستیابی فوق در آرایه $row- assoc یافت نشود برنامه عنصری را با این کلید دستیابی ایجاد کرده و مقدار آن‌را برابر با دنباله کاراکتری تهی قرار می‌دهد. به این ترتیب می‌توانیم کارمان را با ساخت آرایه $row، یعنی اضافه کردن عناصری از آرایه $row_ assoc به آن که شاخص آنها دنباله‌های کاراکتری موجود در آرایه headers می‌باشد، ادامه دهیم. پس از پایان عملیات کلیه حلقه‌ها، آرایه $row شامل یک کپی شاخص‌گذاری شده از عناصر $row_ assoc خواهد بود (در مواردی که مقدار معادل یافت نشده باشد از دنباله کاراکتری تهی استفاده خواهد شد).

هم‌اکنون جهت اضافه کردن سطرهای موجود از داده‌ها به خصوصیت table_ array از یک شئ Table دو روش ساده در اختیار داریم. آنچه اکنون بدان نیاز داریم روشی جهت نمایش داده‌ها برروی صفحه مرورگر اینترنت است.

متد output ( )

متد output ( ) به‌سادگی مقادیر آرایه های headers و table_ array را بر روی صفحه مرورگر به نمایش می‌گذارد. طراحی این متد با هدف تسهیل در اشکال زدایی صورت گرفته است. روش بهتر انجام این کار را بعداً در درس همین ساعت ملاحظه خواهید کرد. کد مربوط به این تابع چنین است:

```
function output ( ) {
   print "< pre >" ;
   foreach ( $this → headers as $headert)
      print "< B > $header < / B > " ;
   print "\ n " ;
   foreach ($this → table_ array as $y) {
      foreach ($y as $xcell)
         print " $xcell " ;
      print "\ n " ;
   }
   print "< /  pre >" ;
}
```

عملکرد این قطعه کد پس از مطالعه آن باید به سادگی برای شما روشن شده باشد. در این متد ابتدا با استفاده از یک ساختار تکرار مقادیر آرایه headers بر روی صفحه مرورگر به نمایش در می‌آید. همین فرآیند در مورد آرایه دیگر یعنی table_ array نیز تکرار می‌شود. از آنجا که آرایه مذکور یک آرایه دو بعدی است، لازم است تا جهت پردازش عناصر آن، یعنی نمایش آنها بر روی صفحه مرورگر از دو حلقه تودرتو استفاده کنیم.

ارائه برنامه کامل

برنامه موجود در لیست ۵- ۸ شامل کد کلاس Table و همچنین کد مربوط به نمونه‌گیری شئ از این کلاس و فراخوانی متدهای آن است.

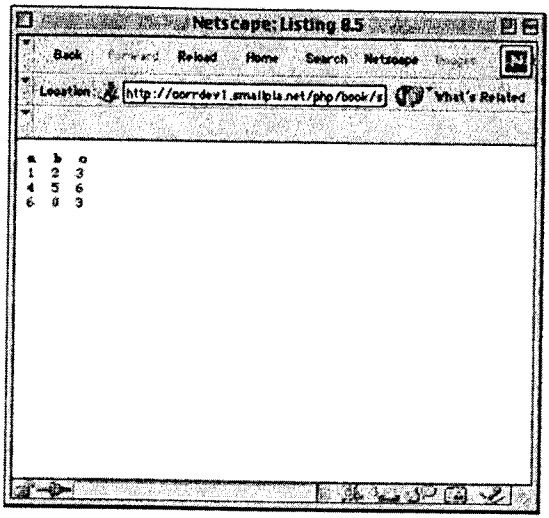
```
 1: <html>
 2: <head>
 3: <title>Listing 8.5</title>
 4: </head>
 5: <body>
 6: <?php
 7: class Table {
 8:     var $table_array = array();
 9:     var $headers = array();
10:     var $cols;
11:     function Table( $headers ) {
12:         $this->headers = $headers;
13:         $this->cols = count ( $headers );
14:     }
15:
```

لیس ۵- ۸ برنامه کامل کلاس Table

```
16:     function addRow( $row ) {
17:         if ( count ($row) != $this->cols )
18:             return false;
19:         array_push($this->table_array, $row);
20:         return true;
21:     }
22:
23:     function addRowAssocArray( $row_assoc ) {
24:         $row = array();
25:         foreach ( $this->headers as $header ) {
26:             if ( ! isset( $row_assoc[$header] ))
27:                 $row_assoc[$header] = "";
28:             $row[] = $row_assoc[$header];
29:         }
30:         array_push($this->table_array, $row);
31:         return true;
32:     }
33:
34:     function output() {
35:         print "<pre>";
36:         foreach ( $this->headers as $header )
37:             print "<B>$header</B>  ";
38:         print "\n";
39:         foreach ( $this->table_array as $y ) {
40:             foreach ( $y as $xcell )
41:                 print "$xcell  ";
42:             print "\n";
43:         }
44:         print "</pre>";
45:     }
46: }
47:
48: $test = new table( array("a","b","c") );
49: $test->addRow( array(1,2,3) );
50: $test->addRow( array(4,5,6) );
51: $test->addRowAssocArray( array ( b=>0, a=>6, c=>3 ) );
52: $test->output();
53: ?>
54: </body>
55: </html>
```

ادامه لیست ۵- ۸

خروجی حاصل از اجرای این برنامه در شکل ۱- ۸ مشاهده می‌شود.



شکل ۱- ۸ عملکرد قابلیتهای کلاس Table

به‌شرطی که طول دنباله‌های کاراکتری موجود در آرایه table_array با هم برابر باشند. شکل و شمایل خروجی کاملاً مناسب است. این وضعیت در صورتی که طول کاراکترها متفاوت باشند دوام خود را از دست می‌دهد.

بررسی کاستی‌های کلاس Table

با وجودی که کلاس Table از عملکرد مناسبی برخوردار است با داشتن فضا و زمان بیشتر شاید بتوان ویژگی‌های بیشتر و امنیت بهتری را نیز تامین نمود.

به‌دلیل اینکه در زبان PHP تعیین نوع داده‌ها مشمول قوانین سفت و سختی نمی‌باشند این وظیفه برنامه‌نویس است که از یکی بودن نوع داده‌های ارسالی به متدها و نوع داده‌هایی که متدها انتظار آنها را می‌کشند، اطمینان حاصل کند. به‌جهت تامین این هدف در درس ساعت شانزدهم، با عنوان "بهره‌گیری از داده‌ها" توابعی را جهت بررسی و کار با داده‌های مختلف معرفی خواهیم نمود. همچنین ممکن است جهت نیل به کارایی بیشتر بخواهید متدهای دیگری را مثلاً برای مرتب کردن عناصر موجود در سطرهای جدول بر مبنای مقادیر موجود در ستونها به امکانات کلاس مرجع Table اضافه کنید.

دلایل استفاده از کلاس

شاید تا به حال این پرسش به ذهن شما خطور کرده باشد که آیا به‌جای استفاده از یک شئ و تعریف کلاس مرجع آن که خود موجب کد سربار و اضافه‌ای جهت ذخیره و بازیابی اطلاعات مورد نظر

می‌شود، به‌سادگی از یک روش دیگر مثلاً از آرایه‌ها استفاده می‌کردیم؟

دلایل متعددی جهت استفاده از این شیوه در دست است. پیش از هر چیز کد ارائه شده از قابلیت استفاده مجدد در سایر برنامه‌ها برخوردار است. این دلیل بسیار روشن است؛ جهت آرایه داده‌ها به روش مشخص و مطلوب و همچنین استفاده از آن در هر برنامه‌ای که نیازمند ذخیره داده‌ها و نمایش آنها به شیوه فوق باشد، بهره‌گیری از کلاس Table کاملاً منطقی و عاقلانه است.

دلیل دوم این است که شئ نمونه‌گیری شده از کلاس Table کاملاً یک شئ فعال است، از این نظر که بدون نیاز به تشکیل یک ساختار تکرار بر روی آرایه table_ array (که خصوصیتی از کلاس مورد نظر است)، می‌توان به‌طور مستقیم از طریق متد تعبیه شده جهت همین کار داده‌های موجود در این آرایه را در قالب مناسبی (از نظر قابلیت خوانایی) بر روی صفحه مرورگر به نمایش درآورد.

دلیل سوم وجود رابطهای مناسب جهت بهره‌گیری از قابلیتهای شئ نمونه‌گیری شده از کلاس مرجع Table است. اگر چنانچه بعد ها تصمیم به بهینه سازی کد مربوط به این کلاس بگیریم، می‌توانیم این کار را بدون تأثیر گذاری بر روی سایر کدهای موجود در پروژه (سایر کلاس‌ها و برنامه‌ها) انجام دهیم به‌شرطی که اسامی متدها، نوع و تعداد آرگومانها و همچنین نوع داده مقادیر بازگشتی از متدهای موجود در کلاس را بدون تغییر رها کنیم.

دلیل آخر برای استفاده از کلاسها این است که می‌توانیم از فواید ارث‌بری کلاسها از یکدیگر، توسعه ساده برنامه‌ها و تعویض عملکردهای کلاس پدر در کلاسهای فرزند بهره‌مند شویم. این همه امکانات برنامه‌نویسی به شیوه شئ‌گرا را به شدت مورد توجه قرار می‌دهد.

ارث بری

برای ایجاد کلاسی که قادر باشد تا عملکردهایی را از کلاس پدر خود به ارث ببرد لازم است تا تغییرات اندکی در معرفی آن بدهیم. برنامه موجود در لیست ۶- ۸ یک نمونه از چنین کلاسی را نشان می‌دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 8.6</title>
 4: </head>
 5: <body>
 6: <?php
 7: class first_class {
 8:     var $name = "harry";
 9:     function first_class( $n ) {
10:         $this->name = $n;
11:     }
12:     function sayHello() {
```

لیست ۶- ۸ ایجاد کلاسی که قادر به ارث بری از کلاس پدر خود است

```
13:         print "Hello my name is $this->name<br>";
14:     }
15: }
16:
17: class second_class extends first_class {
18:
19: }
20:
21: $test = new second_class("son of harry");
22: $test->sayHello();
23: // outputs "Hello my name is son of harry"
24: ?>
25: </body>
26: </html>
```

ادامه لیست ۶- ۸

همان‌گونه که در این برنامه ملاحظه می‌کنید علاوه بر کلاس ساده frist _ class که در خط ۷ تعریف شده‌است، کلاس ساده‌تر دیگری در خط ۱۷ با عنوان second _ class معرفی شده است. به‌نحوه استفاده از واژه کلیدی extends در کلاس دوم توجه خاص داشته باشید. استفاده از این واژه بدان معنی است که اشیای نمونه‌گیری شده از کلاس second_ class از تمام قابلیتهای ذکر شده در کلاس - frist class برخوردار خواهند شد. بدین ترتیب شئ از نوع second _ class دارای متدی با نام ( ) sayHello وخصوصیتی‌با نام name خواهد بود، یعنی همان مشخصه‌هایی که در کلاس frist _ class ذکر شده‌اند.

اگر این وضعیت هنوز موجب تعجب شما نشده است، بنابراین به نکته دیگری در مورد برنامه لیست ۶- ۸ توجه کنید. همان‌گونه که مشاهده می‌کنید در کلاس second_class هیچ متد سازنده‌ای راتعریف‌نکرده‌ایم. بنابراین پرسش با مضمون زیر کاملاً منطقی است: چگونه مقدار پیش‌فرض خصوصیت name یعنی دنباله کاراکتری " harry " به دنباله کاراکتری ارسالی به کلاس second _ class یعنی دنباله کاراکتری " son of harry " تغییر کرده است؟ از آنجا که در کلاس second _ class هیچ متد سازنده‌ای تعریف نشده لذا در صورت لزوم متد سازنده کلاس پدر یعنی frist _ class فراخوانی می‌شود.

> چنانچه‌کلاس فرزندی فاقد تعریف متد سازنده باشد، هنگام نمونه‌گیری از این کلاس متد سازنده کلاس پدر به‌خودی خود فراخوانی خواهد شد. توجه داشته باشید که این ویژگی قابل توجه تنها در PHP4 پشتیبانی شده و نسخه‌های پیشین این زبان فاقد آن هستند.

رونویسی متدهای کلاس پدر در کلاس فرزند

همان‌گونه که در قسمت قبل مشاهده کردید، نمونه‌های ایجاد شده از کلاس second_ class دقیقاً رفتاری مشابه رفتار نمونه‌های ایجاد شده از کلاس first _ class دارند. در برنامه‌های ایجاد شده به شیوه شئ‌گرا کلاسهای فرزند می‌توانند متدهای به ارث رسیده از کلاس پدر را به نحو دلخواه

رونویسی کنند. در این صورت نمونه‌های ایجاد شده از کلاسهای فرزند رفتاری متفاوت با نمونه‌های کلاس پدر خواهند داشت. در برنامه لیست ۷- ۸ متد ( ) sayHello که از کلاس پدر first _ class به کلاس فرزند second _ class به ارث رسیده است، در کلاس فرزند دستخوش تغییر شده است.

```
 1: <html>
 2: <head>
 3: <title>Listing 8.7</title>
 4: </head>
 5: <body>
 6: <?php
 7: class first_class {
 8:     var $name = "harry";
 9:     function first_class( $n ) {
10:         $this->name = $n;
11:     }
12:     function sayHello() {
13:         print "Hello my name is $this->name<br>";
14:     }
15: }
16:
17: class second_class extends first_class {
18:     function sayHello() {
19:         print "I'm not going to tell you my name<br>";
20:     }
21: }
22:
23: $test = new second_class("son of harry");
24: $test->sayHello();
25: // outputs "I'm not going to tell you my name"
26: ?>
27: </body>
28: </html>
```

لیست ۷- ۸ رونویسی متدی از کلاس پدر در کلاس فرزند

همان‌گونه که مشاهده می‌کنید متد ( ) sayHello که در خط ۱۲ کلاس first_ class تعریف شده است، در خط ۱۸ برنامه در کلاس فرزند second _ class مورد رونویسی قرار گرفته است.

فراخوانی یک متد رونویسی شده

گاهی اوقات لازم است تا قابلیتی از کلاس پدر را که در کلاس فرزند آن‌را رونویسی کرده‌ایم در کلاس فرزند مورد استفاده قرار دهیم. در زبانهای برنامه نویسی شئ‌گرا چنین امکاناتی از قبل پیش‌بینی شده است. در برنامه لیست ۸- ۸ متد ( ) sayHello از کلاس فرزند second _ class متد همنام از کلاس پدر first_class را که مورد رونویسی قرار داده است، فراخوانی می‌کند.

```
 1: <html>
 2: <head>
 3: <title>Listing 8.8</title>
 4: </head>
 5: <body>
 6: <?php
 7: class first_class {
 8:     var $name = "harry";
 9:     function first_class( $n ) {
10:         $this->name = $n;
11:     }
12:     function sayHello() {
13:         print "Hello my name is $this->name<br>";
14:     }
15: }
16:
17: class second_class extends first_class {
18:     function sayHello() {
19:         print "I'm not going to tell you my name - ";
20:         first_class::sayHello();
21:     }
22: }
23:
24: $test = new second_class("son of harry");
25: $test->sayHello();
26: // outputs "I'm not going to tell you my name - Hello my name is son of
harry"
27: ?>
28: </body>
29: </html>
```

لیست ۸- ۸ فراخوانی یک متد رونویسی شده

همان‌گونه که مشاهده می‌کنید با استفاده از روش زیر:

Parentclassname : : methodname ( )

می‌توانیم متدهای رونویسی شده را در کلاس فرزند فراخوانی کنیم. این روش در خط ۲۰ برنامه مورد استفاده قرار گرفته است. دقت کنید که روش فوق مختص PHP4 بوده و بهره‌گیری از آن در PHP3 یا نسخه‌های پایین‌تر موجب بروز خطا خواهد شد.

بررسی نمونه‌ای از یک برنامه توارثی

تاکنون تکنیکهای مربوط به چگونگی ارث بردن مشخصه‌ها، رونویسی و همچنین توسعه قابلیتهای کلاس پدر در کلاس فرزند را مورد بررسی قرار دادیم. اکنون می‌توانیم برخی از این تکنیکها را جهت ایجاد کلاسی که مشخصه‌هایی را از کلاس Table که برنامه آن‌را در لیست ۵- ۸ مشاهده کردید به ارث می‌برد، به‌کار ببندیم. کلاس جدید که با نام HTMLTable شناخته خواهد شد در اصل به‌منظور غلبه بر کاستی‌های متد ( ) output کلاس Table مورد طراحی قرار می‌گیرد.

تعریف خصوصیات کلاس HTMLTable

کلاس HTMLTable با استفاده از یک جدول استاندارد HTML اقدام به قالب‌بندی داده‌های موجود خواهد کرد. در این برنامه امکاناتی تعبیه خواهد شد که با استفاده از آنها کاربر کلاس HTMLTable می‌تواند ویژگی‌های جدول HTML خروجی همچون CELLPADDING از نشانه TABLE و BGCOLOR از نشانه TD را تغییر دهد. در برنامه‌های واقعی البته امکان تغییرات بیشتری مورد نظر می‌باشد. به تعریف کلاس HTMLTable توجه کنید:

```
class HTMLTable extends Table {
   var $bgcolor ;
   var $cellpadding = " 2 " ;
}
```

همان‌گونه که مشاهده می‌کنید کلاس جدید HTMLTable به‌عنوان کلاس فرزند Table تعریف شده است (واژه کلیدی extends جهت ایجاد این رابطه نقش اساسی دارد). در این کلاس دو خصوصیت bgcolor و cellpading تعریف شده که خصوصیت اخیر دارای مقدار پیش‌فرض 2 می‌باشد.

ایجاد متد سازنده

پیشتر به این نکته اشاره شد که اگر طراحی کلاس فرزند فاقد تعریف متد سازنده باشد هنگام نمونه‌گیری از کلاس فرزند در برنامه، متد سازنده کلاس پدر به‌طور خودکار فراخوانی می‌شود. با این وجود در طرح کلاس HTMLTable ترجیح می‌دهید تا متد سازنده‌ای را به‌طور ویژه برای این کلاس تعریف کنیم. تعریف این متد به قرار زیر است:

```
function HTMLTable ( $headers, $bg = "# ffffff " ) {
   Table : : Table ($headers) ;
   $this → bgcolor = $bg ;
}
```

همان‌گونه که ملاحظه می‌کنید متد سازنده HTMLTable آرایه‌ای از اسامی ستونهای جدول و نیز یک دنباله کاراکتری را به‌عنوان آرگومان ورودی می‌پذیرد. آرگومان دوم که یک دنباله کاراکتری با مقدار پیش‌فرض " ffffff #" می‌باشد عهده‌دار تعیین مقدار خصوصیت bgcolor از کلاس جدید است. در ابتدا متد سازنده کلاس پدر یعنی Table جهت ارسال آرگومان $headers به‌منظور تعیین مقادیر اسامی ستونهای جدول فراخوانی می‌گردد. از آنجا که خلاصه‌نویسی امر پسندیده‌ای در برنامه‌نویسی است در این برنامه عملیات فوق به‌عهده متد سازنده کلاس Table گذاشته شده و بدین ترتیب نگرانی خود در مورد نحوه انجام این کار را از بین برده‌ایم. پس از این فرآیند خصوصیت bgcolor به‌عنوان بخشی از کد متد سازنده HTMLTable مقداردهی شده است.

توجه داشته باشید که اگر کلاس فرزند دارای متد سازنده باشد، در این صورت متد سازنده کلاس پدر به‌طور خودکار فراخوانی نمی‌شود. به‌عبارت دیگر، در صورت نیاز به فراخوانی متد سازنده کلاس پدر باید آن‌را به‌طور صریح فراخوانی کنیم.

طراحی متد ( ) setCellpadding

کلاس فرزند علاوه بر متدهایی که از کلاس پدر خود به ارث می‌برد، می‌تواند متدهایی را نیز تعریف کند. متدی که با عنوان ( ) setCelllpadding در این قسمت طراحی خواهیم کرد به کاربر اجازه می‌دهد تا خصوصیت Celllpadding را به‌طور دلخواه مقداردهی کند. البته در بهترین حالت ممکن باید بتوان ترتیبی داد که مقدار خصوصیت Celllpadding مستقیماً از خارج شئ قابل تنظیم باشد، اما در بیشتر موارد این حالت توصیه نمی‌شود. به‌عنوان یک قاعده سرانگشتی، همواره بهتر است تا جهت تغییر مقادیر خصوصیات، متدهایی را در درون کلاس تعریف کنیم. در نسخه پیچیده‌تر این کلاس ممکن است نیاز باشد تا متد ( ) setCellpadding جهت تنظیم مقدار خصوصیت Celllpadding اقدام به تغییر خصوصیات دیگری از کلاس نماید. متاسفانه باید به این نکته منفی اعتراف کنیم که در PHP4 هیچ روش مناسبی جهت کنترل دقیق این‌گونه دستیابی‌ها وجود ندارد. به تعریف این متد توجه کنید:

```
function setCellpadding ( $padding ) {
   $this → Celllpadding = $padding ;
}
```

طراحی متد ( ) output

متد ( ) output کلاس HTMLTable ، متد همنام خود از کلاس پدر Table را به‌طور کامل رونویسی خواهد کرد. عملیاتی که این متد انجام می‌دهد مشابه کاری است که متد همنام در کلاس پدر انجام می‌دهد منتها با این تفاوت که خروجی در قالب یک جدول HTML به نمایش در می‌آید. به تعریف این متد توجه کنید:

```
function output ( ) {
   print "< table cellpadding = \" $this → cellpadding \" border = 1 >" ;
   foreach ( $this → headers as $header)
    print "< td bgcolor = \" $this → bgcolor \" > < b > $header < / b > < / td >" ;
   foreach ( $this → table _ array as $row ⇒ $cells) {
    print "< tr >" ;
   foreach ( $cells as $cell)
    print "< td bgcolor = \" $this → bgcolor \" > $cell < / td >" ;
   print "< / tr >" ;
   }
print "< / table >" ;
}
```

چنانچه عملکرد متد مشابه از کلاس پدر Table را به‌خوبی درک کرده باشید، درک عملکرد این متد اکنون بسیار ساده خواهد بود. همان‌گونه که ملاحظه می‌کنید در ازای هربار گذر از حلقه‌های تکرار مقادیر آرایه‌های table _ array و header بر روی صفحه مرورگر به نمایش در می‌آید. واضح است که قالب‌بندی خانه‌های جدول و نیز رنگ زمینه آنها توسط خصوصیات cellpadding و bgcolor کنترل می‌شود.

ارائه برنامه کامل

برنامه موجود در لیست ۹-۸ کلاسهای Table و HTMLTable را در قالب یک اسکریپت PHP مورد استفاده قرار داده است. این برنامه ضمن نمونه‌گیری از کلاس فرزند HTMLTable اقدام به تغییر مقدار خصوصیت cellpadding کرده و پس از تعیین داده‌های جدول آنها را با استفاده از متد (output) این کلاس بر روی صفحه مرورگر اینترنت در قالب یک جدول HTML نمایش می‌دهد. در مثالهای واقعی‌تر داده‌های برنامه از طریق مقادیر ذخیره شده در یک بانک اطلاعاتی تأمین می‌شوند.

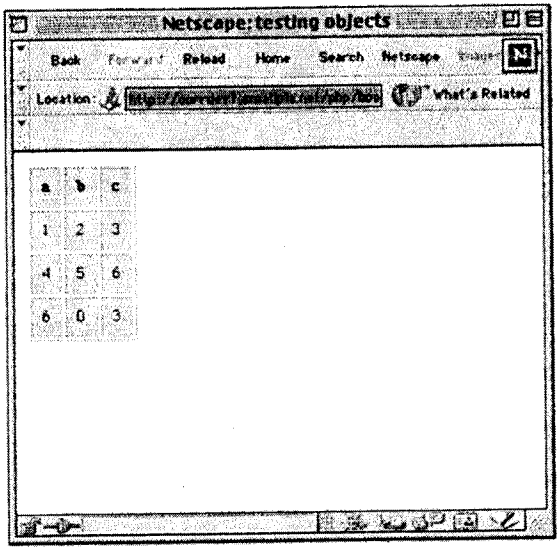
```
1: <html>
2: <head>
3: <title>testing objects</title>
4: </head>
5: <body>
6: <?php
7: class Table {
8:     var $table_array = array();
9:     var $headers = array();
10:    var $cols;
11:    function Table( $headers ) {
12:        $this->headers = $headers;
13:        $this->cols = count ( $headers );
14:    }
15:
16:    function addRow( $row ) {
17:        if ( count ($row) != $this->cols )
18:            return false;
19:        array_push($this->table_array, $row);
20:        return true;
21:    }
22:
23:    function addRowAssocArray( $row_assoc ) {
24:        if ( count ($row_assoc) != $this->cols )
25:            return false;
26:        $row = array();
27:        foreach ( $this->headers as $header ) {
28:            if ( ! isset( $row_assoc[$header] ))
29:                $row_assoc[$header] = " ";
30:            $row[] = $row_assoc[$header];
31:        }
32:        array_push($this->table_array, $row);
```

لیست ۹- ۸ برنامه کاملی که از کلاس پدر Table و کلاس فرزند HTMLTable استفاده می‌کند

```
33:     }
34:
35:     function output() {
36:         print "<pre>";
37:         foreach ( $this->headers as $header )
38:             print "<B>$header</B>  ";
39:         print "\n";
40:         foreach ( $this->table_array as $y ) {
41:             foreach ( $y as $xcell )
42:                 print "$xcell  ";
43:             print "\n";
44:         }
45:         print "</pre>";
46:     }
47: }
48:
49: class HTMLTable extends Table {
50:     var $bgcolor;
51:     var $cellpadding = "2";
52:     function HTMLTable( $headers, $bg="#ffffff" ) {
53:         Table::Table($headers);
54:         $this->bgcolor=$bg;
55:     }
56:
57:     function setCellpadding( $padding ) {
58:         $this->cellpadding = $padding;
59:     }
60:     function output() {
61:         print "<table cellpadding=\"$this->cellpadding\" border=1>";
62:         foreach ( $this->headers as $header )
63:             print "<td bgcolor=\"$this->bgcolor\"><b>$header</b></td>";
64:         foreach ( $this->table_array as $row=>$cells ) {
65:             print "<tr>";
66:             foreach ( $cells as $cell )
67:                 print "<td bgcolor=\"$this->bgcolor\">$cell</td>";
68:             print "</tr>";
69:         }
70:         print "</table>";
71:     }
72: }
73: $test = new HTMLTable( array("a","b","c"), "#00FF00");
74: $test->setCellpadding( 7 );
75: $test->addRow( array(1,2,3));
76: $test->addRow( array(4,5,6));
77: $test->addRowAssocArray( array ( b=>0, a=>6, c=>3 ));
78: $test->output();
79: ?>
80: </body>
81: </html>
```

دنباله لیست ۹- ۸

خروجی حاصل از اجرای این برنامه در شکل ۲- ۸ قابل ملاحظه است.



شکل ۲- ۸ خروجی برنامه لیست ۹- ۸

دلایل استفاده از ویژگی توارث

یک پرسش خوب می‌تواند در اینجا بدین صورت مطرح شود که دلیل جداسازی کلاس Table از HTMLTable چیست؟ آیا نمی‌توانستیم در فضا و زمان برنامه‌نویسی صرفه‌جویی کرده و قابلیت نمایش داده‌ها در قالب جدول HTML را در همان کلاس پدر Table تعبیه کنیم؟ پاسخ به‌طور خلاصه این است: " قابلیت انعطاف. "

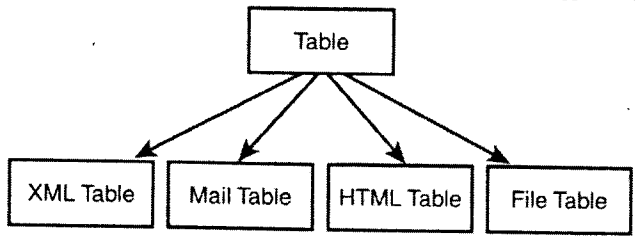
حالتی را تصور کنید که نیاز باشد تا کلاسی با قابلیت نگهداری جدولی از مقادیر که بر مبنای اسامی ستونها سازماندهی می‌شوند، ایجاد کنیم. اگر در چنین حالتی از قبل کلاسی در اختیارتان باشد که ضمن جمع‌آوری و ذخیره اطلاعات آنها را در قالب یک جدول HTML سازماندهی کرده و خروجی را بر روی صفحه مرورگر اینترنت نمایش دهد، نیازی به کلاس جدیدی که این قابلیت را پیاده‌سازی کند، نخواهید داشت.

حال وضعیت دیگری را درنظر بگیرید که مجبور باشید علاوه بر نمایش داده‌های موجود در قالب یک جدول HTML در مرورگر اینترنت، آنها را در یک فایل متن بر روی دیسک ذخیره نمایید. در این حالت به احتمال قوی لازم است تا متدها و خصوصیات دیگری را نیز به کلاس موجود اضافه کنید.

وضعیت هنگامی بغرنج می‌شود که به‌عنوان مثال پس از گذشت یک هفته مجبور باشید خروجی برنامه را علاوه بر نمایش آنها در جدول HTML و ذخیره آنها در یک فایل متنی، از طریق پست الکترونیکی ارسال نمایید. اگر اینترانت (شبکه داخلی) شرکت شما از طریق اسناد XML رابطه بین کارمندان را برقرار می‌کند، شاید لازم باشد تا این داده‌ها را علاوه بر اشکال فوق به شکل اسناد

XML نیز منتشر کنید. در چنین مواردی پیاده‌سازی همه این قابلیتها در یک کلاس تنها نشانه‌های آغاز یک دردسر بزرگ است چراکه لازم است تا برنامه مورد نظر کاملاً بازنویسی شود.

اجازه دهید تا مطلب فوق را با توجه به آنچه که در مورد کلاسهای Table و HTMLTable مشاهده کردیم، دقیق‌تر مورد بررسی قرار دهیم. تا بدین‌جای کار توانستیم فرآیند قالب‌بندی (نمایش) داده‌ها را از فرآیند تولید آنها تفکیک کنیم. هنگامی که نیاز به ذخیره داده‌ها در قالب یک فایل متن باشد تنها کاری که باید انجام دهیم این است که کلاس جدیدی ایجاد کنیم تا قابلیتهایی را از کلاس Table به ارث ببرد. فرض کنید نام این کلاس جدید FileTable باشد. برای ایجاد یک چنین کلاسی نیازی نیست تا کد موجود در کلاس Table را تغییر دهیم. این فرآیند می‌تواند در مورد کلاسهای جدید دیگری مثل MailTable و XMLTable نیز تکرار شود. شکل ۳- ۸ روابط موجود مابین کلاس پدر Table و کلاسهای فرزند را نشان می‌دهد.



شکل ۳- ۸ روابط پدر و فرزندی موجود بین کلاس‌های Table و سایر کلاس‌ها

آنچه که مسلم است این است که اشیای نمونه‌گیری شده از کلاسهای فرزند کلاس پدر Table همگی شامل متدی با نام ( ) output می‌باشند. از این جهت می‌توانیم آنها را در درون یک آرایه ذخیره نماییم.بدین‌ترتیب هنگام نیاز به انواع مختلف خروجی می‌توانیم متد ( ) output را در قالب یک ساختار تکرار به تعداد دفعات مورد نیاز و بدون توجه در مورد نحوه پیاده‌سازی این متد در کلاسهای مربوطه فراخوانی کنیم.به‌عبارت‌دیگر،بااستفاده‌ازآرایه‌ای‌از نمونه‌های کلاس Table می‌توانیم خروجی‌های مختلفی مانندe-mail، HTML ، XML و حتی یک متن ساده را تنها با فراخوانی متد ( ) output به دست آوریم.

کسب اطلاعات در مورد کلاسها و اشیا

تا کنون مشاهده کردید که می‌توان از توابعی همچون تابع ( ) gettype جهت پی بردن به نوع داده ذخیره شده در یک متغیر استفاده کرد. به‌واسطه این‌گونه توابع می‌توان اطمینان حاصل کرد که توابع مورد استفاده در برنامه با نوع آرگومان‌های مورد انتظارشان تغذیه خواهند شد.

همه اشیای موجود، از هر کلاسی که نمونه‌گیری شده باشند در نهایت از نوع داده object محسوب می‌شوند. با این حال در برخی موارد لازم است تا اطلاعات بیشتری در این مورد به دست آوریم.

تعیین نوع کلاس یک شئ

فرض کنید کلاسی داریم که به‌سادگی دنباله‌ای از کاراکترها را به خروجی می‌فرستد. با این وجود متدی از این کلاس که عهده‌دار انجام این کار است، مستلزم دریافت آرگومانی از نوع کلاس ObjectFilter است. به کد مربوط به این کلاس توجه کنید:

```
Class SayHello {
   Function print_ hello ( $filter _ object) {
      Print  $filter _ object → filter ( "hello you < br >" ) ;
   }
}
class outputFilter {
   function filter ( $txt ) {
      return "< b > $txt < / b >" ;
   }
}
$hello = new  SayHello ( ) ;
$hello → print_ hello (new outputFilter ( ) ) ;
```

همان‌گونه که مشاهده می‌کنید کلاس sayHello از این موضوع مطمئن است که شئ ارسال شده به متد print_ hello از آن به‌عنوان آرگومان از متدی با نام ( ) filter برخوردار است. واضح است که مسئله اطمینان در این قسمت مورد توجه خاص ما قرار گرفته است. با استفاده از تابع سیستمی ویژه‌ای با نام ( ) get _class می‌توان اطمینان پیدا کرد که متغیر ارسال شده به تابع print _ hello به‌عنوان آرگومان نمونه‌ای از کلاس مرجع outputFilter است. تابع( ) get _ class یک شئ را به‌عنوان آرگومان ورودی دریافت کرده و نام کلاس مرجع آن‌را (البته با حروف کوچک انگلیسی) باز می‌گرداند. به نسخه بهتری از کلاس SayHello توجه کنید:

```
class SayHello {
   function print _ hello ( $filter _ object ) {
      if (get _ class ( $filter _ object) ! = " outputfilter " ) reteurn false ;
      print $filter _ object → filter ( " hello you < br > " ) ;
   }
}
```

با این تغییر، تابع ( ) print_ hello همواره از این مطلب مطمئن خواهد بود که نمونه‌ای از کلاس outputFilter را به‌عنوان آرگومان در اختیار دارد.

تعیین نوع کلاس پدر یک شئ

در مبحث مطرح شده در قسمت قبل، شاید از این موضوع که فرآیند قالب‌بندی خروجی به شئ که‌ازخارج کلاس SayHello تامین شده بود، واگذار شد، تعجب‌کرده باشید. حال تصور کنید که چندین کلاس دیگر جهت قالب‌بندی خروجی در اختیار داریم که همگی از فرزندان کلاس outputFilter

هستند. به کد مربوط به این کلاسها توجه کنید:

```
Class ItalicFilter extends OutputFilter {
   Function filter ( $txt ) {
      return "< i > $txt < / i >" ;
   }
}
class UnderlineFilter extends OutputFilter {
   function filter ( $txt ) {
      return "< u > $txt  < / u >" ;
   }
}
class BlinkFilter extends OutputFilter {
   function filter ( $txt ) {
      return "< blink >$txt  < / blink  >" ;
   }
}
```

آن‌گونه که از ظواهر امر پیداست نمی‌توانیم نمونه‌ای از کلاس BlinkFilter را به‌عنوان آرگومان به متد ( ) print _ hello از کلاس SayHello ارسال کنیم. آیا بهتر نبود به روشی می‌توانستیم از دغدغه خاطر کلاس SayHello بکاهیم به‌گونه‌ای که متد( ) print_ hello از این کلاس قادر به پذیرفتن کلیه نمونه‌هایی از کلاسهای فرزند کلاس مرجع OutputFilter (یعنی کلاسهای UnderlinFilter, ItalicFilter و BlinkFilter) باشد؟ قدر مسلم اگر کلاسی از خانواده کلاس مرجع OutputFilter باشد، حتماً دارای متدی با عنوان ( ) filter خواهد بود.

تابع ( ) is _ subclass _ of پاسخ این معما است. این تابع نام یک شئ و همچنین نام کلاسی را که شئ باید از آن مشتق شده باشد، به‌عنوان آرگومان دریافت می‌کند. اگر این شئ نمونه‌ای از کلاسی باشد که فرزند کلاس ارسال شده به تابع ( ) is _ subclass _ of است، تابع مقدار true و در غیر این‌صورت مقدار false را باز می‌گرداند. بدین ترتیب متد ( ) print _ hello با اطمینان بیشتری می‌تواند عملیات خود را انجام دهد.

برنامه لیست ۱۰- ۸ تمامی قطعات را در قالب یک اسکریپت PHP آرایه می‌کند. به‌تعریف متد ( ) print _ hello در خط ۳ و چگونگی استفاده از متد ( ) is _ subclass _of در خط ۵ توجه کنید.

```
 1: <?php
 2: class SayHello {
 3:         function print_hello( $filter_object ) {
 4:                 if ( get_class( $filter_object ) != "outputfilter" &&
 5:                      ! is_subclass_of( $filter_object,"outputfilter")  )
 6:                                 return false;
 7:                 print $filter_object->filter( "hello you<br>" );
 8:         }
 9: }
10:
```

لیست ۱۰- ۸ بررسی کلاس و مسئله ارث بری

```
11: class OutputFilter {
12:         function filter( $txt ) {
13:                 return "<b>$txt</b>";
14:         }
15: }
16:
17: class ItalicFilter extends OutputFilter {
18:         function filter( $txt ) {
19:                 return "<i>$txt</i>";
20:         }
21: }
22:
23: class UnderlineFilter extends OutputFilter {
24:         function filter( $txt ) {
25:                 return "<u>$txt</u>";
26:         }
27: }
28:
29: class BlinkFilter extends OutputFilter {
30:         function filter( $txt ) {
31:                 return "<blink>$txt</blink>";
32:         }
33: }
34:
35: $hello = new SayHello();
36: $hello->print_hello( new OutputFilter() );
37: $hello->print_hello( new ItalicFilter() );
38: $hello->print_hello( new UnderlineFilter() );
39: $hello->print_hello( new BlinkFilter() );
40: ?>
```

ادامه لیست ۱۰-۸

بررسی وجود کلاس و متد

به موازات رشد کتابخانه‌ها، کلاسها نیز به طور فزاینده‌ای به یکدیگر وابسته می‌شوند. بدین ترتیب ممکن است در مواقعی یک کلاس قصد استفاده از امکانات کلاس دیگری را داشته باشد که در دسترس آن برنامه نباشد. از این‌رو در زبان PHP امکاناتی جهت بررسی وجود کلاسها و متدها پیش از به‌کارگیری آنها در برنامه پیش‌بینی شده است.

تابع سیستمی class _ exists ( ) نام کلاسی را در قالب یک دنباله کاراکتری به‌عنوان آرگومان دریافت می‌کند. چنانچه کلاس مورد نظر یافت شود تابع مقدار true و در غیر این‌صورت مقدار false را باز می‌گرداند. این متد به‌ویژه هنگامی مفید است که نام کلاس مورد نظر در یک دنباله کاراکتری ذخیره شده باشد. به‌نمونه‌ای از نحوه به‌کارگیری این تابع توجه کنید:

```
if ( class _ exists ( $class _ name) )
   $obj = new  $class _ name ( ) ;
```

تابع دیگر در همین رابطه method _ exists ( ) نام دارد. این تابع سیستمی جهت اجرا به دو آرگومان نیاز دارد. آرگومان اول نام یک شئ و آرگومان دوم نام متد مورد بررسی است. به نمونه‌ای از

به‌کارگیری این تابع توجه کنید:

```
if ( method _ exists ( $filter _ object, " filter ") )
   Print  $filter _ object → filter ( "hello you < br >" ) ;
```

در قطعه کد فوق چنانچه متدی با نام ( ) filter در شئ $filter _ object موجود باشد، تابع مورد بحث مقدار true و در غیر این‌صورت مقدار false را باز می‌گرداند.

ذخیره و بازیابی اشیا

معمولاً اشیا را از ترکیب داده‌های ذخیره شده در یک رسانه ذخیره‌سازی ایجاد می‌کنیم. به‌عبارت دیگر، از داده‌های موجود جهت ایجاد اشیا استفاده کرده و پس از انجام پردازشهای لازم داده‌ها را دوباره در محل مربوطه ذخیره می‌کنیم. با این حال گاهی از اوقات لازم است تا اشیا و داده‌ها را دست‌نخورده در جایی ثبت نماییم. در زبان PHP دو تابع سیستمی برای انجام این فرآیند پیش‌بینی شده است.

اولین تابع ( ) serialize نام دارد. از این تابع می‌توان جهت ثبت و نگهداری اشیا استفاده کرد. این تابع نام یک شئ از کلاس را به‌عنوان آرگومان پذیرفته و یک دنباله کاراکتری ایجاد می‌کند. این دنباله کاراکتری را می‌توان در یک فایل یا یک بانک اطلاعاتی ذخیره کرد و یا به برنامه دیگری ارسال نمود. به نمونه‌ای از به‌کارگیری این تابع توجه کنید:

```
Class apple {
   Var $flavor = "sweet" ;
}
$app = new apple ( ) ;
$stored = serialize ( $app ) ;
print $stored ;
// this print : "0 : 5 : "apple" : 1 : { s : 6 : "flavor" ; s : 5 : "sweet"; } "
```

دنباله کاراکتری تولید شده توسط تابع ( ) serialize را می‌توان با استفاده از تابع سیستمی دیگری با عنوان ( ) unserialize به یک شئ از همان نوع تبدیل کرد. اگر هنگام استفاده از تابع ( ) unserialize کلاس اصلی شئ مورد بحث در دسترس باشد، یک کپی دقیق از شئ اصلی ایجاد خواهد شد. به نمونه زیر توجه کنید:

```
$new_ app = unserialize ( $stored ) ;
print  $new_app → flavor ;
// prints "sweet"
```

در برخی موارد لازم است تا پیش از ذخیره شئ کارهای کوچک دیگری انجام شود. این فرآیند به‌ویژه هنگامی از اهمیت زیاد برخوردار است که شئ مورد نظر اتصالی را با یک بانک اطلاعاتی موجود باز کرده باشد یا پیش از این در حال کار با یک فایل بوده باشد. به‌همین ترتیب ممکن است لازم باشد تا پس از ساخته شدن مجدد شئ به نوعی درگیر فرآیند مقداردهی متغیرها شود. برای کنترل چنین

مواردی همواره می‌توان دو متد مخصوص را در اشیایی که احتمال می‌دهیم به ذخیره بازیابی آنها احتیاج پیدا خواهیم کرد، تعبیه کنیم. این دو متد عبارتند از ( ) sleep - - و ( ) wakeup - - .

متد ( ) sleep - - پیش از ذخیره شئ مورد نظر به خودی خود توسط تابع ( ) serialize فراخوانی می‌شود. این امکان خوبی را جهت انجام برخی کارهایی که پیش از ذخیره شئ لازم است صورت بگیرد، در اختیار می‌گذارد. برای اجرای موفقیت آمیز فرآیند ذخیره سازی، متد ( ) sleep - - باید آرایه‌ای از اسامی خصوصیاتی را بازگرداند که برنامه قصد ذخیره آنها را در قالب دنباله کاراکتری دارد، به نمونه‌ای از کاربرد این متد توجه کنید:

```
class apple {
   var  $flavor = "sweet" ;
   var  $frozen = 0 ;
   function sleep ( ) {
      $this → frozen + + ;
      // any clean up stuff goes here
      return array _keys (get _ object _ vars ( $this) ) ;
   }
}
$app = new apple ( ) ;
$stored = serialize ( $app) ;
print $stored ;
// prints "0 : 5 :"apple" : 2 : { s : 6 :"flavor" ; s : 5 :"sweet" ; s : 6 :"frozen" ; i : 1 ; } "
```

به روشی که در انتهای تعریف متد ( ) sleep - - جهت لیست نمودن اسامی تمامی خصوصیات موجود در شئ به‌کار بردیم، توجه کنید. در این روش از تابع سیستمی( ) get _object _ vars استفاده شده است. این تابع جهت اجرا به یک شئ به‌عنوان آرگومان نیاز دارد. مقدار بازگشتی این تابع یک آرایه انجمنی است که شامل تمامی خصوصیات آن شئ می‌باشد. همان‌گونه که مشاهده می‌کنید حاصل فراخوانی تابع ( ) get _object _ vars به‌عنوان آرگومان به تابع دیگری با نام ( ) array _ keys ارسال شده است. تابع ( ) array _ keys به عنوان آرگومان آرایه‌ای را (که معمولاً یک آرایه انجمنی است) دریافت کرده و آرایه دیگری را که شامل مقادیر کلید دستیابی آرگومان خود است، باز می‌گرداند. متد دیگر در این رابطه ( ) wakeup - - نام دارد. از این متد هنگام بازیابی استفاده می‌شود. اگر چنین متدی در کلاس تعریف شده باشد به‌طور خودکار توسط تابع ( ) unserialize فراخوانی می‌شود. از این متد می‌توان جهت برقراری اتصال با بانک اطلاعاتی یا هر نوع مقداردهی اولیه مورد نیاز شئ استفاده کرد. به نمونه‌ای از کاربرد این متد توجه کنید:

```
Function - - wakeup ( ) {
   Print  "This apple has been frozen " . $ this → frozen . "time (s) " ;
   // any initialization stuff goes here
}
```

حال با در دست داشتن تعریف این متد می‌توانیم تابع ( ) unserialize را فراخوانی کنیم:

```
$new _ app = unserialize ($stored) ;
```

```
// prints "This apple has been frozen 1 time (s) "
```

جمع بندی

غیر ممکن است بتوان تمامی ویژگی‌های برنامه نویسی به شیوه شئ‌گرا را در عرض تنها یک ساعت مورد بررسی قرار داد، اما با این حال امیدواریم که دست کم با بخشی از آن آشنا شده باشید.

وسعت استفاده از اشیا و کلاسها در پروژهای برنامه‌نویسی موضوعی است که باید با توجه به شرایط در مورد آن تصمیم‌گیری کرد. به احتمال قوی پروژه‌های برنامه‌نویسی که در آنها به میزان زیادی از ویژگی‌های برنامه‌نویسی به شیوه شئ‌گرا استفاده شده است نسبت به پروژه‌هایی که به شیوه سنتی پیاده‌سازی شده‌اند نیازمند منابع بیشتری جهت اجرا می‌باشند. با این وجود پیاده‌سازی هوشمندانه و مؤثر ویژگی‌های برنامه‌نویسی شئ‌گرا به میزان زیادی می‌تواند قابلیت عملکرد و انعطاف برنامه‌ها و همچنین سازماندهی آنها را افزایش دهد.

در درس این ساعت سعی شد تا آنجا که مقدور بود چگونگی ایجاد کلاس و نمونه‌گیری اشیایی از آن مورد توجه قرار بگیرد. در این درس چگونگی ایجاد خصوصیات و متدهای کلاس را مورد بررسی قرار داده و در مورد چگونگی ایجاد کلاسهای جدیدی که ویژگی‌ها و قابلیتهایی را از یک کلاس دیگر به ارث برده و بعضاً آنها را رونویسی می‌کنند، به بحث نشستیم.

در انتهای این ساعت نیز بحثی را درباره چگونگی تشخیص کلاس مرجع یک شئ و چگونگی اطلاع از این موضوع که آیا کلاس مورد نظر خود فرزند کلاس دیگری است یا خیر ترتیب دادیم.

اکنون که هسته اصلی زبان برنامه‌نویسی PHP را تا بدین جای کتاب مورد بررسی قرار دادیم وقت آن است که دانش خود را گسترش داده و به بحث در مورد سایر ویژگی‌ها و امکانات این زبان بپردازیم. در درس ساعت آینده به امکاناتی از زبان PHP که در رابطه با فرمهای HTML پیش‌بینی شده‌اند، خواهیم پرداخت.

پرسش و پاسخ

پرسش: در درس این ساعت، مباحث نا آشنایی مورد بررسی قرار گرفتند. آیا یک برنامه‌نویس خوب PHP لزوماً باید برنامه‌نویسی به شیوه شئ‌گرا را بداند؟

پاسخ: پاسخ در یک کلام منفی است. بیشتر برنامه‌های PHP به میزان اندک یا حتی هیچ از ویژگی‌های برنامه نویسی به شیوه شئ‌گرا بهره می‌گیرند. استفاده از این شیوه بدین معنی نیست که در صورت عدم استفاده از آن نتوان برنامه‌های مورد نظر را توسعه داد. فایده استفاده از این شیوه در چگونگی سازماندهی برنامه‌ها، قابلیت استفاده مجدد و قابلیت توسعه عملکردها نهفته است.

حتی در صورتی که نخواهید به این شیوه برنامه‌نویسی کنید، ممکن است حین توسعه به برنامه‌های دیگری که شامل کلاس هستند، نیاز پیدا کنید. مطالب این درس کمک می‌کنند تا این‌گونه برنامه‌ها را درک کنید.

پرسش: متغیر ویژه $this اندکی موجب سردرگمی است. ممکن است توضیح خلاصه‌ای در مورد آن بدهید؟

پاسخ: در داخل کلاس گاهی لازم است که متدهای همان کلاس فراخوانی شده و یا خصوصیات آن مورد دستیابی قرار بگیرند. با استفاده از متغیر $this و به‌دنبال آن عملگر → می‌توان هر دوموردفوق را انجام داد. متغیر $this را می‌توان به‌سادگی ابزاری دانست که هنگام تعریف (نمونه‌گیری) اشیای کلاس ایجاد شده و امکان دستیابی به اشیا را از درون خود آنها در اختیار قرار دهد.

تمرینها

هدف از این بخش دوره مطالب فراگیری شده در قالب آزمون است. پاسخ بخش آزمون بلافاصله بعد از آن آمده است. بخش فعالیتها شامل تمرینهایی است که به‌منظور افزایش قابلیت برنامه‌نویسی خواننده طراحی شده و البته فاقد پاسخ است.

آزمون

۱- چگونه می‌توان یک کلاس تهی با نام emptyClass ایجاد کرد به‌گونه‌ای که شامل هیچ متد یا خصوصیتی نباشد؟

۲- با فرض در اختیار داشتن کلاس emptyClass در تمرین قبل چگونه می‌توان از این کلاس نمونه‌گیری کرد؟

۳- چگونه می‌توان خصوصیتی را در درون یک کلاس معرفی کرد؟

۴- چگونه می‌توان نام متد سازنده یک کلاس را انتخاب کرد؟

۵- چگونه می‌توان متد سازنده یک کلاس را ایجاد کرد؟

۶- چگونه می‌توان یک متد معمولی در یک کلاس تعریف کرد؟

۷- چگونه می‌توان دستیابی و تنظیم مقادیر خصوصیتها و دستیابی به متدها را در درون یک کلاس صورت داد؟

۸- چگونه می‌توان خصوصیات و متدهای یک کلاس را از خارج آن کلاس مورد دستیابی قرار داد؟

۹- برای اینکه کلاسی بتواند قابلیتهایی را از یک کلاس دیگر به ارث ببرد چه چیزی باید در تعریف آن کلاس تغییر کند؟

پاسخ آزمون

۱- تعریف کلاس با استفاده از واژه کلیدی class صورت می‌گیرد:

```
Class emptyClass {
}
```

۲- جهت نمونه‌گیری از کلاس همواره عملگر new مورد استفاده قرار می‌گیرد:

```
$obj = new emptyClass ( ) ;
```

۳- با استفاده از واژه کلیدی var می‌توان خصوصیتی را در یک کلاس تعریف کرد:

```
Class point {
   // properties
   var $x = 0 ;
   var $y = 0 ;
```

۴- نام متد سازنده انتخابی نیست. متدهای سازنده یک کلاس همواره همنام آن کلاس هستند.

۵- با تعریف متدی که نام آن با نام کلاس معادل است می‌توان یک متد سازنده برای آن کلاس ایجاد کرد. متد سازنده هنگام ایجاد نمونه‌هایی آن کلاس به‌طور خودکار فراخوانی می‌شوند:

```
Class Point {
   // properties
   var $x = 0 ;
   var $y = 0 ;
   // constructor
   function Point ( $x, $y ) {
      // set up code gose here
   }
}
```

۶- متدهای معمولی توابعی از کلاس هستند که با اسامی آنها با نام کلاس معادل نیست:

```
Class Point {
   // properties
   var $x = 0 ;
   var $y = 0 ;
   // constructor
   function Point ( $x, $y ) {
      // set up code gose here
   }
   // method .
   function moveTo ( $x, $y ) {
   }
}
```

۷- با ترکیب متغیر ویژه‌ای با نام $this و عملگر → می‌توان خصوصیات و متدهای کلاس را از درون خود کلاس مورد دستیابی قرار داد:

```
Class Point {
   // properties
```

```
    var $x = 0 ;
    var $y = 0 ;
    // constructor
    function Point ( $x, $y ) {
        // calling a method
        $this → moveTo ( $x, $y ) ;
    }
    // method
    function moveTo ( $x, $y ) {
        // setting properties
        $this → x = $x ;
        $this → y = $y ;
    }
}
```

۸- با ذکر نام شئ (که معمولاً در درون یک متغیر ذخیره می‌شود) و به دنبال عملگر → می‌توان خصوصیات و متدهای آن شئ را از خارج آن شئ مورد دستیابی قرار داد:

```
// instantiating an object
$p = new Point ( 40 , 60 ) ;

// calling an object's method
$p → moveTo ( 20 , 200 ) ;

// accessing an object's property
print $p → x ;
```

۹- واژه کلیدی extends ابزاری است که با استفاده از آن می‌توان رابطه توارث را مابین دو کلاس تعریف کرد. در تعریف کلاسی که در این رابطه شرکت می‌کند این واژه بعد از نام کلاس فرزند و قبل از نام کلاس پدر (در تعریف کلاس فرزند) واقع می‌شود:

```
Class FunkyPoint extends Point {
}
```

فعالیتها

۱- کلاسی با نام baseCalc تعریف کنید به‌گونه‌ای که قادر به ذخیره دو عدد صحیح در قالب دو خصوصیت باشد. در این کلاس متدی با نام ( ) calculate تعریف کنید که اعداد مذکور را بر روی صفحه مرور گر نمایش دهد.

۲- کلاسی با نام addCalc تعریف کنید به صورتی که قابلیتهایی را از کلاس baseCalc در تمرین قبل به ارث ببرد، متد ( ) calculate کلاس پدر را به‌گونه‌ای رونویسی کنید که مجموع مقادیر خصوصیات کلاس را بر روی صفحه مرورگر نمایش دهد.

۳- تمرین قبل را در مورد کلاس دیگری با نام minusCalc تکرار کنید. متد ( ) calculate کلاس پدر را به گونه‌ای رونویسی کنید که حاصل تفاضل مقادیر خصوصیات کلاس را بر روی صفحه مرورگر نمایش دهد.

« بخش سوم: بهره‌گیری از PHP »

- ساعت نهم: بهره‌گیری از فرم‌ها
- ساعت دهم: بهره‌گیری از فایل‌ها
- ساعت یازدهم: بهره‌گیری از توابع DBA
- ساعت دوازدهم: ارتباط با بانکهای اطلاعاتی- بهره‌گیری از MySQL
- ساعت سیزدهم: بررسی عملیات سمت سرور
- ساعت چهاردهم: گرافیک و PHP
- ساعت پانزدهم: بهره‌گیری از تاریخ و ساعت
- ساعت شانزدهم: بهره‌گیری از داده‌ها
- ساعت هفدهم: بهره‌گیری از دنباله‌های کاراکتری

- ساعت هجدهم: بهره‌گیری از عبارات منظم
- ساعت نوزدهم: ثبت وضعیت با استفاده از کوکی‌ها و رشته‌های پرس‌و جو
- ساعت بیستم: ثبت وضعیت با استفاده از توابع مربوط به ثبت جلسات
- ساعت بیست و یکم: بهره‌گیری از محیط سرور
- ساعت بیست و دوم: PHP و XML

ساعت نهم

بهره‌گیری از فرم‌ها

تا بدین‌جای کتاب در مورد تمامی مثالها یک بعد بسیار مهم و اساسی را نادیده گرفتیم. اکنون شما می‌توانید متغیرها و آرایه‌ها را مقدار دهی کنید، توابع مورد نیازتان را تعریف کرده و آنها را فراخوانی نمایید و با اشیای مختلفی که از کلاسهای مورد نظرتان نمونه‌گیری کردید، کار کنید. تمامی اینها در صورتی که قادر به نمایش اطلاعات مورد نظر به کاربر نباشیم از ارزش چندانی برخوردار نخواهند بود. در درس این ساعت به‌دنبال راهی هستیم تا بتوانیم اطلاعات مورد نیاز کاربر را به او عرضه کرده و از اطلاعاتی که وی وارد سیستم می‌کند، در برنامه استفاده کنیم.

در وب جهان گستر (World Wide Web)، فرم‌های HTML از معانی اساسی بوده و به این مفهوم است که حجمی از اطلاعات را می‌توان از کاربری به سرور ارسال کرد. زبان برنامه‌نویسی PHP به‌گونه‌ای طراحی شده که امکاناتی را جهت دریافت اطلاعات و بهره‌گیری از اطلاعات دریافتی از طریق فرم‌های HTML در اختیار برنامه‌نویس و کاربر وب قرار می‌دهد.

در درس این ساعت با این مطالب آشنا می‌شوید:

- **چگونگی دستیابی و استفاده از متغیرهای سیستمی**
- **چگونگی دستیابی به اطلاعات وارد شده در فیلدهای یک فرم HTML**
- **چگونگی کار با عناصری از فرم که امکان انتخاب چندین گزینه را در اختیار قرار می‌دهند**
- **چگونگی ایجاد سندی که هم شامل یک فرم HTML و هم شامل یک برنامه PHP جهت ارسال مقادیر وارد شده در عناصر فرم است**
- **چگونگی ثبت وضعیت با استفاده از فیلدهای پنهان فرم**
- **چگونگی تغییر مسیر کاربر به یک صفحه جدید**
- **چگونگی ایجاد فرم‌های HTML جهت بارگذاری فایل‌ها و بررسی برنامه PHP کنترل‌کننده آن**

در ادامه به بررسی موارد فوق می‌پردازیم.

متغیرهای سیستمی

پیش از اقدام عملی جهت ایجاد یک فرم و استفاده از آن برای جمع‌آوری داده‌ها، لازم است تا اندکی تأمل کرده و نگاهی دوباره به متغیرهای سراسری بیندازیم. اگر خاطرتان باشد اولین برخوردمان با این‌گونه متغیرها در درس ساعت ششم با عنوان " توابع " بود. متغیر سراسری متغیری است که در بالاترین سطح یک برنامه معرفی شده باشد، به بیان دیگر هر متغیری که خارج از تابع تعریف شده باشد، یک متغیر سراسری است. تمامی متغیرهای برنامه در یک آرایه انجمنی سیستمی با نام $GLOBALS ثبت می‌شوند. این وضعیت در برنامه‌هایی چون لیست ۱- ۹ مفید است، چراکه امکان دستیابی به آنها به‌راحتی از طریق یک ساختار تکرار فراهم می‌گردد.

```
 1: <html>
 2: <head>
 3: <title>Listing 9.1 Looping through the $GLOBALS array</title>
 4: </head>
 5: <body>
 6: <?php
 7: $user1 = "Bob";
 8: $user2 = "Harry";
 9: $user3 = "Mary";
10: foreach ( $GLOBALS as $key=>$value ) {
11:    print "\$GLOBALS[\"$key\"] == $value<br>";
12: }
13: ?>
14: </body>
15: </html>
```

لیست ۱- ۹ دستیابی به عناصر آرایه $GLOBALS

همان‌گونه که مشاهده می‌کنید در خطوط ۷ تا ۹ از این برنامه سه متغیر سراسری را معرفی کرده و در داخل یک ساختار تکرار در خطوط ۱۰ و ۱۱ عناصر آرایه انجمنی $GLOBALS را مورد دستیابی قرار داده‌ایم. تابع سیستمی Print در خط ۱۱ به‌سادگی کلیدهای دستیابی و مقادیر مربوطه را بر روی صفحه مرورگر نمایش می‌دهد. در خروجی برنامه قادر خواهیم بود تا متغیرهای تعریف شده را به راحتی تشخیص دهیم. اما مطلب به همین‌جا ختم نمی‌شود. PHP به‌طور خودکار متغیرهای سراسری را تعریف می‌کند؛ این متغیرها محیطهای سرور و کلاینت را توصیف می‌کنند و بسته به سیستم عامل مورد استفاده‌تان، و همچنین نوع سرور (وب سرور) و تنظیمات مربوطه، امکان دستیابی به این متغیرها تفاوت خواهد داشت. با این حال در مورد مفید بودن آنها شک نداشته باشید. جدول ۱- ۹ اسامی و جزئیات برخی از مهم‌ترین متغیرهای سیستمی را که البته از نوع متغیرهای سراسری

هستند، نشان می‌دهد. تمامی این متغیرها از طریق آرایه سیستمی $GLOBALS در دسترس شماست.

جدول ۱- ۹ اسامی برخی از متغیرهای سیستمی

نام متغیر	محتوا	مثال کاربردی
$HTTP _ USER _ AGENT	نام و نسخه مرورگر اینترنت مورد استفاده کاربر (کلاینت)	Mozila / 4.6 (X11; I; Linux2.2.6 - 15 apmac ppc)
$REMOTE _ ADDR	آدرس IP کلاینت	158.152.55.35
$REMOTE _ METHOD	نوع درخواست: یکی از دو مورد GET یا POST	POST
$QUERY _ STRING	در مورد درخواستهای نوع GET، داده‌های کدبندی شده‌ای که به انتهای URL ضمیمه می‌شوند	Name = matt & address = Unknown
$REQUEST _ URI	آدرس کامل درخواست شامل رشته پرس‌و جو (مورد قبل)	/ matt / php - book / forms / eg 9.14. html ? name = matt & address = Unknown
$HTTP _ REFERER	آدرس صفحه‌ای که درخواست از آنجا تنظیم شده است.	http: // www. Test. com / a _ page. html

علاوه بر موارد ذکر شده در جدول فوق، PHP متغیرهای سراسری دیگری را نیز در اختیار برنامه‌نویسان قرار می‌دهد. برای مثال متغیر $GLOBALS [" PHP- SELF "] مسیر دقیق برنامه در حال اجرا را در اختیار می‌گذارد. در یک سیستم نمونه این مسیر می‌تواند به شکل زیر باشد:

/ dev / php 24 / ch 9 / listing 9.1. php

متغیر مذکور را می‌توان مستقیماً با نام $PHP_ SELF نیز مورد دستیابی قرار داد.

برنامه‌ای جهت جمع‌آوری داده‌های ورودی

در حال حاضر سعی ما این است که کد مربوط به فرم HTML را از کد برنامه PHP تفکیک کنیم. برنامه لیست ۲- ۹ یک فرم ساده HTML را نشان می‌دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 9.2 A simple HTML form</title>
 4: </head>
 5: <body>
 6: <form action="listing9.3.php">
 7: <input type="text" name="user">
 8: <br>
 9: <textarea name="address" rows="5" cols="40">
10: </textarea>
11: <br>
12: <input type="submit" value="hit it!">
13: </form>
14: </body>
15: </html>
```

لیست ۲- ۹ یک فرم ساده HTML

همان‌گونه که مشاهده می‌کنید این فرم شامل فیلد متنی با نام " User " است که در خط ۷ تعریف شده است. عناصر دیگر این فرم عبارتند از یک منطقه متنی (text area) با نام " address " و یک دکمه ارسال که به ترتیب تعریف آنها در خطوط ۹ و ۱۲ آمده است. قصد ما در اینجا بررسی جزئیات مربوط به HTML نیست. اگر علاقمند به فراگیری این جزئیات و نحوه استفاده از زبان نشانه‌گذاری HTML هستید، پیشنهاد می‌کنیم کتاب " خودآموز HTML در ۲۴ ساعت " یا یکی از چندین هزار سند online مربوط به جزییات این زبان نشانه‌گذاری را مطالعه کنید. در لیست فوق ملاحظه می‌کنید که آرگومان ACTTON از عنصر FORM به فایلی با نام listing 9.3. php که وظیفه پردازش اطلاعات فرم را به‌عهده دارد، اشاره می‌نماید. از آنجا که غیر از نام این فایل اطلاعات دیگری را در اختیار آرگومان ACTION قرار نداده‌ایم لذا فایل listing 9.3. php باید در همان فهرستی از کامپیوتر سرور که شامل سند HTML است، واقع باشد.

برنامه PHP موجود در لیست ۳- ۹ مقادیری را که کاربر در فرم HTML لیست قبل وارد کرده است، نمایش می‌دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 9.3 Reading input from the form in Listing 9.2</title>
 4: </head>
 5: <body>
 6: <?php
 7: print "Welcome <b>$user</b><P>\n\n";
 8: print "Your address is:<P>\n\n<b>$address</b>";
 9: ?>
10: </body>
11: </html>
```

لیست ۳- ۹ بازیابی و نمایش مقادیر وارد شده توسط کاربر در فرم تعریف شده در لیست ۲- ۹

این اولین برنامه اسکریپت PHP موجود در کتاب است که به‌واسطه کلیک بر روی یک پیوند (Link) یا تایپ مستقیم نام آن در فیلد آدرس مرورگر اینترنت اجرا نمی‌شود. ما برنامه موجود در

لیست مذکور را در قالب فایلی با عنوان listing 9.3. php ذخیره کرده‌ایم. این برنامه هنگامی فراخوانی می‌شود که کاربر پس از واردکردن اطلاعات در فرم HTML لیست ۲- ۹ دکمه موجود بر روی آن فرم را جهت ارسال کلیک نماید.

همان‌گونه که ملاحظه می‌کنید در این برنامه دو متغیر با اسامی $user و $address مورد دستیابی قرار گرفته‌اند. اکنون نباید از این موضوع تعجب کنید که این متغیرها شامل مقادیری هستند که کاربر در فیلدهای متن " user " و " address " از فرم HTML لیست ۲- ۹ وارد کرده است. فرم‌های HTML در برنامه‌های PHP به‌راحتی مورد پردازش قرار می‌گیرند. هرگونه اطلاعاتی که کاربر در قالب فرم‌های HTML ارسال می‌کند در متغیرهای سراسری همنام با فیلدهای موجود بر روی آن فرم‌ها در اختیار برنامه‌های PHP مربوطه قرار خواهند گرفت.

دستیابی به اطلاعات یک فرم با استفاده از آرایه‌ها

تا بدین جا چگونگی دریافت اطلاعات وارد شده در فیلدهایی از فرم HTML را که شامل تنها یک مقدار هستند (مانند فیلدهای متن)، مشاهده نمودید. در مواقعی لازم است تا داده‌های موجود در عناصری از فرم‌های HTML را که شامل چندین مقدار هستند مانند عنصر SELECT مورد پردازش قرار دهید. عنصر SELECT این امکان را در اختیار کاربر قرار می‌دهد تا از میان گزینه‌های متعدد چند گزینه مورد نظر خود را انتخاب کند. چنانچه عنصر SELECT موجود در فرم HTML را به‌سادگی با یک دنباله کاراکتری به صورت زیر نام‌گذاری کرده باشیم:

< Select name = "products" multiple >

آن‌گاه برنامه PHP که این اطلاعات را دریافت می‌کند تنها به یک مقدار ساده که معادل با نام این عنصر است (در اینجا "$products")، دسترسی خواهد داشت. با این حال می‌توانیم با درج جفت علامت [ ] پس از نام این عنصر در فرم HTML مربوطه این رفتار را تغییر دهیم. آنچه که در این حالت به‌دست می‌آید آرایه‌ای از مقادیر خواهد بود. این فرآیند در برنامه لیست ۴- ۹ قابل مشاهده است.

```
 1: <html>
 2: <head>
 3: <title>Listing 9.4 An HTML form including a SELECT element</title>
 4: </head>
 5: <body>
 6: <form action="listing9.5.php" method="POST">
 7: <input type="text" name="user">
 8: <br>
 9: <textarea name="address" rows="5" cols="40">
10: </textarea>
11: <br>
12: <select name="products[]" multiple>
13: <option>Sonic Screwdriver
14: <option>Tricorder
```

لیست ۴- ۹ یک فرم HTML که شامل عنصری از نوع SELECT می‌باشد.

```
15: <option>ORAC AI
16: <option>HAL 2000
17: </select>
18: <br>
19: <input type="submit" value="hit it!">
20: </form>
21: </body>
22: </html>
```

ادامه لیست ۴- ۹

اکنون با ایجاد تغییر فوق در کد مربوط به فرم HTML، برنامه اسکریپتی که مقادیر وارد شده در این فرم را پردازش می‌کند، شامل آرایه‌ای خواهد بود که با نام عنصر SELECT فرم HTML (در اینجا products [ ]) همنام است (در این مورد آرایه مذکور در متغیری با نام $products ذخیره می‌شود). این آرایه معادل عنصر SELECT فرم HTML است و شامل عناصری می‌باشد که در خطوط ۱۳ و ۱۶ این لیست مشخص شده است. برنامه موجود در لیست ۵- ۹ بیانگر این است که مقادیر تعریف شده در عنصر SELECT از فرم HTML فوق در قالب آرایه‌ای در اختیار برنامه PHP مربوطه قرار خواهند گرفت.

```
1: <html>
2: <head>
3: <title>Listing 9.5 Reading input from the form in Listing 9.4</title>
4: </head>
5: <body>
6: <?php
7: print "Welcome <b>$user</b><p>\n\n";
8: print "Your address is:<p>\n\n<b>$address</b><p>\n\n";
9: print "Your product choices are:<p>\n\n";
10: if ( ! empty( $products ) ) {
11:     print "<ul>\n\n";
12:     foreach ( $products as $value ) {
13:        print "<li>$value<br>\n";
14:     }
15:     print "</ul>";
16: }
17: ?>
18: </body>
19: </html>
```

لیست ۵- ۹ بازیابی مقادیر فرم HTML لیست ۴- ۹

همان‌گونه که مشاهده می‌کنید در خط ۷ از این برنامه متغیر $user که معادل عنصری از فرم HTML با نام user است، مورد دستیابی قرار گرفته است. در خط ۱۰ برنامه متغیر (آرایه) $products مورد بررسی قرار گرفته است. اگر این متغیر دارای مقدار باشد (به‌عبارت دیگر آرایه تهی از مقدار نباشد)، توسط یک ساختار تکرار در خط ۱۲ مقادیر موجود در آن بازیابی شده و در خط ۱۳ با فراخوانی تابع سیستمی Print به‌نمایش در می‌آیند.

هرچند که این تکنیک به‌طور خاص جهت دستیابی به مقادیر عنصر SELECT مورد استفاده

قرار می‌گیرد، اما حقیقت این است که با روش فوق می‌توان مقادیر هر نوع عنصری از فرم‌های HTML را مورد دستیابی قرار داد. برای مثال با منسوب کردن یک نام واحد به مجموعه‌ای از کادرهای علامت (check box) می‌توان این امکان را در برابر کاربر قرار داد تا از یک فیلد ساده چندین مقدار را انتخاب نماید. به‌شرطی که در انتهای نام فوق از جفت علامت [ ] استفاده شود، PHP مقادیر انتخاب شده توسط کاربر را در آرایه‌ای با همین نام ذخیره خواهد کرد. با جایگزین کردن عنصر SELECT (خطوط ۱۲ تا ۱۷ لیست ۴- ۹) با چندین کادر علامت به‌صورت زیر می‌توان مجدداً همین نتیجه را ملاحظه کرد:

```
< input type = "checkbox"  name = "products [ ] "
   value = "sonic screwdriver" > sonic screwdriver < br >
< input type = "checkbox" name = "products [ ]"
   value = "Tricorder" > Tricorder < br >
< input type = "checkbox"  name = "products [ ]"
   value = "ORAC AI" > ORAC AI < br >
< input type = "checkbox" name = "products [ ]"
   value = "HAL 2000" > HAL 2000 < br >
```

به این نکته توجه کنید که بهره‌گیری از این روش به آرایه‌هایی که به روش عددی شاخص‌گذاری شده‌اند، محدود نمی‌شود. می‌توان اطلاعات وارد شده در فرم ورودی را در یک آرایه انجمنی و یا حتی در آرایه‌های چندبعدی نیز ذخیره کرد. برای مثال جهت سازماندهی بهتر داده‌ها می‌توان اطلاعات وارد شده در فرم ورودی را در یک آرایه انجمنی مثلاً با نام $form ذخیره کرد. این عمل به‌سادگی با نام‌گذاری اسامی فیلدهای فرم معادل با کلیدهای دستیابی آرایه انجمنی (البته بدون ذکر علامت دلار) امکان‌پذیر است. قطعه کد زیر روند عملیات را نشان می‌دهد:

```
< input type = " text " name = " form [user] " > < br >
< textarea name = "form [address]" rows = "5" cols = "40" >
< / textarea >
```

پس از ارسال فرم مربوطه جهت پردازش می‌توان عناصر ' user ' و ' address ' از فرم HTML را از طریق آرایه $form در برنامه PHP مورد دستیابی قرار داد.

همچنین جهت استفاده از یک آرایه چندبعدی در این روش می‌توان به‌سادگی فرآیند مربوط به نام‌گذاری را که در پاراگراف قبل در مورد آرایه‌های انجمنی مشاهده کردید، اندکی توسعه داد. روند عملیات بدین ترتیب است:

```
< input type = "checkbox" name = " form [products] [ ]"
   value = " Sonic Screwdriver " > Sonic Screwdriver < br >
< input type = "checkbox" name = " form [products] [ ]"
   value = " Tricorder " > Tricorder < br >
< input type = "checkbox" name = " from [products] [ ]"
   value = " ORCA AI " > ORCA AI < br >
< input type = "checkbox" name = " form [products] [ ]"
   value = " HAL 2000 " > HAL 2000 < br >
```

هنگام ارسال این فرم، عنصر $form [products] باید شامل یک آرایه شاخص‌گذاری شده عددی باشد که عناصر آن موارد انتخاب شده از کادرهای علامت فرم HTML می‌باشند.

دستیابی به اطلاعات وارد شده در فرمهای HTML با استفاده از آرایه‌های سیستمی

تکنیکهایی که تا بدین‌جا در مورد دستیابی به اطلاعات وارد شده در فرم‌های HTML بررسی کردیم ازعملکردخوبی برخوردارند، اما به‌واسطه وجود متغیرهای سراسری ممکن است موجب سردرگمی شوند. جهت محدود کردن تعداد متغیرهای سراسری در برنامه می‌توان گزینه register _ globals از فایل php. ini را با مقدار off تنظیم نمود. این کار باعث جلوگیری از ایجاد متغیرهای سراسری به ازای فیلدهای موجود در یک فرم HTML خواهد شد. در درس ساعت دوم با عنوان " نصب PHP بر روی کامپیوتر" در مورد فایل php. ini و قابلیتهای مربوطه صحبت کردیم.

این عمل موجب جلوگیری از سردرگمی ناشی از وجود تعداد زیادی متغیر سراسری در برنامه می‌شود، اما پرسش این است که چگونه می‌توان به عناصر فرم و اطلاعات آنها دسترسی پیدا کرد؟

پاسخ این پرسش متغیرهای سراسری است که PHP4 آنها را در اختیارمان قرار می‌دهد. بسته به اینکه فرم مورد نظر از کدام‌یک از دو روش ارسال، یعنی POST یا GET استفاده می‌کند به یکی از دو متغیر سراسری $HTTP_ GET_ VARS یا $HTTP_ POST_ VARS و یا هر دو دسترسی خواهید داشت. متغیرهای مذکور در واقع آرایه‌های انجمنی هستند که کلیدهای دسترسی آنها را اسامی فیلدهای HTML و مقادیر مربوطه را اطلاعات وارد شده در این فیلدها تشکیل می‌دهند. برنامه لیست ۶- ۹ از مزایای این روش جهت نمایش مقادیر یک فرم ارسال شده به روش GET بهره‌گرفته است.

```
 1: <html>
 2: <head>
 3: <title>Listing 9.6 Reading input from any form using the $HTTP_GET_VARS
➥array</title>
 4: </head>
 5: <body>
 6: <?php
 7: foreach ( $HTTP_GET_VARS as $key=>$value ) {
 8:    print "$key == $value<BR>\n";
 9: }
10: ?>
11: </body>
12: </html>
```

لیست ۶- ۹ دستیابی به مقادیر فرم با استفاده از آرایه سیستمی $HTTP_ GET_ VARS

برنامه فوق لیستی از اسامی و مقادیر تمام پارامترهای ارسالی به آن از طریق روش ارسال GET را نمایش می‌دهد. فرآیند مشابه با استفاده از آرایه $HTTP_ POST_ VARS و روش ارسال POST عملی است.

تشخیص نحوه ارسال فرم

به‌جهت انعطاف بیشتر، برنامه‌های اسکریپت پس از دریافت داده‌ها در مورد اینکه داده‌های دریافتی را از کدام دو آرایه $HTTP_ GET_ VARS یا $HTTP_ POST_ VARS بازیابی کنند، باید تصمیم‌گیری نمایند. به‌عبارت دیگر این‌گونه برنامه‌ها باید قادر به تشخیص نحوه ارسال داده‌ها (یکی از دو روش ممکن GET یا POST) باشند. بر روی بیشتر سیستم‌ها می‌توان شیوه ارسال را با دستیابی به مقدار متغیر سیستمی $REQUEST_METHOD تشخیص داد. مقدار این متغیر همواره یکی از دنباله‌های کاراکتری " get " یا " post " (که بیانگر نوع ارسال GET و POST می‌باشند) است. با این وجود جهت اطمینان قطعی از قابلیت حمل برنامه همواره می‌توان هر دو آرایه نامبرده را مورد بررسی قرار داد.

برنامه لیست ۷- ۹ جهت اطمینان از این موضوع روش خاصی را مورد استفاده قرار می‌دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 9.7 Extracting parameters from
 4:       either a GET or POST request</title>
 5: </head>
 6: <body>
 7: <?php
 8: $PARAMS = ( count( $HTTP_POST_VARS ) )
 9:         ? $HTTP_POST_VARS : $HTTP_GET_VARS;
10:
11: foreach ( $PARAMS as $key=>$value ) {
12:    print "$key == $value<BR>\n";
13: }
14:
15: ?>
16: </body>
17: </html>
```

لیست ۷- ۹ بازیابی پارامترهای مربوط به درخواست GET یا POST

همان‌گونه که مشاهده می‌کنید در خط ۸ از این برنامه از عملگر سه عملوندی : ? جهت مقداردهی متغیر $PARAMS استفاده کرده‌ایم. با استفاده از تابع سیستمی ( ) count ، ابتدا وجود مقادیر در آرایه $HTTP_ POST_ VARS مورد بررسی قرار می‌گیرد. اگر چنانچه آرایه نامبرده تهی نباشد، متغیر $PARAMS با مقادیر این آرایه و در غیر این‌صورت با مقادیر موجود در آرایه $HTTP_

GET_ VARS مقداردهی می‌شود. به‌هرحال پس از مقداردهی متغیر $PARAMS از طریق یکی از دو آرایه ذکر شده بدون توجه به طریقه ارسال (GET یا POST) برنامه به کار خود ادامه می‌دهد.

ترکیب کد HTML و PHP در یک سند واحد

گاهی اوقات ممکن است مایل باشیم تا کد بررسی‌کننده داده‌های یک فرم HTML را در خود سند HTML تعبیه کنیم. چنین ترکیبی از کدهای HTML و PHP معمولاً هنگامی مفید است که بخواهیم فرم مورد نظر را بیش از یک مرتبه برای کاربر نمایش دهیم. البته در صورتی که بتوانیم کدنویسی کل سند را به‌صورتی پویا انجام دهیم فواید بیشتری از نقطه‌نظر قابلیت انعطاف و عملکرد عایدمان خواهد شد، اما این فرآیند به‌بهای از دست‌دادن یکی از قدرتمندترین ویژگی‌های PHP به دست خواهد آمد. هرچه حجم استفاده از کد HTML استاندارد در یک سند بیشتر باشد کار طراحی و توسعه برنامه‌های اسکریپتی که با داده‌های موجود در این فرم‌ها کار می‌کنند نیز به‌مراتب راحت‌تر و سرراست‌تر خواهد بود. با این وجود به‌عنوان یک قاعده مهم و مؤثر این نکته را به‌خاطر بسپارید که همواره باید از پراکندگی کد PHP در سطح اسنادتان جلوگیری به‌عمل آورید. هرجا که ممکن باشد توابعی را جهت فراخوانی از درون کد HTML طراحی کرده و مورد استفاده قرار دهید؛ به‌خاطر داشته باشید که همواره می‌توانید از این توابع در سایر پروژه‌ها نیز بهره‌مند شوید.

اما در مورد مثالی که در ادامه خواهید دید، فرض کنید در حال ایجاد وب سایتی هستیم که وظیفه آن در نهایت آموزش درس ریاضیات در مقطع پیش‌دبستانی است. به‌عنوان بخشی از این وب سایت لازم است یک برنامه PHP توسعه دهیم. این برنامه باید قادر به دریافت عددی که کاربر در یک فیلد از فرم وارد کرده باشد و پس از مقایسه آن عدد با یک عدد از پیش تعریف شده، باید نتیجه مقایسه را اعلام نماید.

برنامه لیست ۸- ۹ شامل یک فرم HTML است. هرچند که در این مثال تنها به یک فیلد متن جهت ورود اطلاعات نیاز داریم، با این حال به برنامه‌ای هرچند کوچک برای پردازش اطلاعات نیاز خواهیم داشت.

```
 1: <html>
 2: <head>
 3: <title>Listing 9.8 An HTML form that calls itself</title>
 4: </head>
 5: <body>
 6: <form method="POST">
 7: Type your guess here: <input type="text" name="guess">
 8: </form>
 9: </body>
10: </html>
```

لیست ۸- ۹ یک فرم ساده HTML شامل تنها یک فیلد متن جهت ورود داده‌ها

بدون توجه به نامی که به سند شامل این فرم می‌دهید، این حقیقت که از صفت action در عنصر form استفاده شده است، بدین معنی است که فرم مذکور اطلاعات را به سوی خودش ارسال می‌کند. به‌عبارت بهتر، این فرم جهت پردازش اطلاعات وارد شده خودش را فراخوانی می‌کند.

> تقریباً تمامی مرورگرهای اینترنت در صورتی که عنصر form از یک فرم HTML فاقد صفت action باشد آن فرم را جهت پردازش به صفحه جاری تحویل می‌دهند. با این وجود می‌توان به‌طور صریح مرورگرهای اینترنت را مجبور کرد تا فرم را به سند مربوطه (صفحه جاری) تحویل دهند. این عمل با استفاده از متغیر سیستمی $PHP_ SELF به صورت زیر قابل پیاده‌سازی است:
>
> < form action = " < ? php print $PHP_ SELF ? > " >

برنامه لیست ۸- ۹ بدین ترتیب هیچ‌گونه خروجی تولید نمی‌کند، در عوض برنامه موجود در لیست ۹- ۹ کد PHP مورد نیاز جهت پردازش این فرم را به سند اضافه کرده است. پیش از هر چیز لازم است تا عددی را که کاربر وارد کرده مورد دستیابی قرار دهیم. در جامع‌ترین حالت برنامه می‌توانیم ترتیبی دهیم تا عدد مورد نظر به طور تصادفی تولید شود اما اکنون قصد نداریم در آن سطح پیچیده خود را درگیر کنیم. همان‌گونه که ملاحظه می‌کنید در خط ۲ از برنامه، عدد صحیح 42 را به متغیر $num _ to _ guess نسبت داده‌ایم. گام بعدی این است که از ارسال فرم اطمینان حاصل کنیم، چه در غیر این‌صورت قصد دستیابی به متغیری را خواهیم داشت که هنوز در دسترس ما قرار نگرفته است. جهت بررسی ارسال فرم می‌توانیم وجود متغیر $guess را مورد بررسی قرار دهیم. چنانچه پارامتری با نام guess به برنامه اسکریپت ارسال شده باشد متغیر سراسری $guess در اختیار برنامه خواهد بود. در صورتی‌که چنین متغیری در برنامه موجود نباشد می‌توان این‌گونه فرض کرد که کاربر هنوز اقدامی در رابطه با ارسال فرم صورت نداده است. اما در صورتی‌که متغیر سراسری $guess موجود باشد، می‌توانیم مقدار آن را جهت پردازشهای بعدی مورد دستیابی قرار دهیم. همان‌گونه که مشاهده می‌کنید بررسی وجود متغیر سراسری $guess در خط ۴ از برنامه لیست ۹- ۹ انجام شده است.

```
1: <?php
2: $num_to_guess = 42;
3: $message = "";
4: if ( ! isset( $guess ) )
5:    $message = "Welcome to the guessing machine!";
6: elseif   ( $guess > $num_to_guess )
7:    $message = "$guess is too big! Try a smaller number";
8: elseif   ( $guess < $num_to_guess )
9:    $message = "$guess is too small! Try a larger number";
10: else // must be equivalent
11:    $message = "Well done!";
```

لیست ۹- ۹ برنامه PHP مورد نیاز فرم لیست ۸- ۹

```
12:
13: ?>
14: <html>
15: <head>
16: <title>Listing 9.9 A PHP number guessing script</title>
17: </head>
18: <body>
19: <h1>
20: <?php print $message ?>
21: </h1>
22: <form method="POST">
23: Type your guess here: <input type="text" name="guess">
24: </form>
25: </body>
26: </html>
```

ادامه لیست ۹- ۹

کاملاً واضح است که قسمت اعظم این برنامه متشکل از یک ساختار تصمیم‌گیری if است که مقدار متغیر $message را که یک دنباله کاراکتری است، تعیین می‌کند. در صورتی‌که متغیر $guess فاقد مقدار باشد، چنین فرض می‌شود که کاربر اولین باری است که صفحه مورد نظر را مشاهده می‌کند. چنین وضعیتی باعث می‌شود در خط ۵ از برنامه یک دنباله کاراکتری شامل پیغام خوش‌آمد گویی به متغیر $messege نسبت داده شود.

از طرف دیگر، اگر متغیر $guess شامل یک مقدار عددی باشد این مقدار با مقدار ذخیره شده در متغیر $num _ to _ guess مقایسه شده و دنباله کاراکتری مناسبی بسته به نتیجه به متغیر $message نسبت داده می‌شود. بررسی این مطلب که مقدار ذخیره شده در متغیر $guess بزرگتر از مقدار متغیر $num _to_guess است یا کوچک‌تر از آن، به ترتیب در خطوط ۶ و ۸ از برنامه فوق انجام می‌شود. اگر حاصل این مقایسه‌ها نشان داد که مقدار متغیر $guess نه بزرگ‌تر و نه کوچک‌تر از مقدار متغیر $num _ to _ guess می‌باشد، می‌توان چنین نتیجه گرفت که مقدار این دو متغیر با یکدیگر برابر است. در چنین حالتی پیغام تبریکی مبنی بر درست بودن حدس کاربر به نمایش در می‌آید. پس از قطعی شدن مقدار متغیر $message تنها کار مورد نیاز نمایش این دنباله کاراکتری (به‌عنوان بدنه سند HTML) بر روی صفحه مرورگر اینترنت است.

کارهای بیشتری را می‌توان در رابطه با این برنامه انجام داد اما آنچه مسلم است این است که می‌توان به‌سادگی این فرم را در اختیار یک طراح صفحات HTML قرار داد. وی می‌تواند بدون هیچ تاثیری بر روی برنامه یا برنامه نویسی اقداماتی حرفه‌ای جهت تزئین آن انجام دهد.

بهره‌گیری از فیلدهای مخفی جهت ثبت وضعیت

در برنامه لیست ۹- ۹ هیچ تدبیری برای اطلاع از اینکه کاربر چند بار اقدام به وارد کردن عدد

در فرم نموده، پیش‌بینی نشده است. برای اطلاع از این موضوع می‌توان از یک فیلد مخفی استفاده نمود. رفتار فیلدهای مخفی دقیقاً مشابه فیلدهای معمولی است با این تفاوت که کاربر قادر به مشاهده آن فیلدها نمی‌باشد (مگر آنکه کد HTML مربوط به آنها را مورد بررسی قرار دهد). برنامه موجود در لیست ۱۰- ۹ شامل همان برنامه لیست قبلی است با این تفاوت که از یک فیلد مخفی و اندکی نیز کد PHP جهت بهره‌گیری از آن استفاده می‌کند.

```
 1: <?php
 2: $num_to_guess = 42;
 3: $num_tries = ( isset( $num_tries ) ) ? ++$num_tries : 0;
 4: $message = "";
 5: if ( ! isset( $guess ) )
 6:    $message = "Welcome to the guessing machine!";
 7: elseif   ( $guess > $num_to_guess )
 8:    $message = "$guess is too big! Try a smaller number";
 9: elseif   ( $guess < $num_to_guess )
10:    $message = "$guess is too small! Try a larger number";
11: else // must be equivalent
12:    $message = "Well done!";
13:
14: $guess = (int) $guess;
15: ?>
16: <html>
17: <head>
18: <title>Listing 9.10 Saving state with a hidden field</title>
19: </head>
20: <body>
21: <h1>
22: <?php print $message ?>
23: </h1>
24: Guess number: <?php print $num_tries?>
25: <form method="POST">
26: Type your guess here:
27: <input type="text" name="guess" value="<?php print $guess?>">
28: <input type="hidden" name="num_tries" value="<?php print $num_tries?>">
29: </form>
30: </body>
31: </html>
```

لیست ۱۰- ۹ ثبت وضعیت با استفاده از یک فیلد مخفی

همان‌گونه که مشاهده می‌کنید نام فیلد مخفی در این برنامه num _ tries است که در خط ۲۸ تعریف شده است. در این برنامه جهت نمایش مقدار این فیلد از PHP استفاده کرده‌ایم. علاوه براین در خط ۲۷ از فیلد، متن ساده‌ای جهت نمایش آخرین مقدار ورودی بهره گرفته‌ایم، به‌گونه‌ای که کاربر بتواند همواره مقدار قبلی این فیلد را مشاهده کند. این روش در مواقعی مفید است که قصد تجزیه و تحلیل ورودی را داشته باشیم. بدین ترتیب اگر به‌دلایلی از ارسال فرم صرف‌نظر کنیم، می‌توانیم شانس تغییر مقدار آخرین ورودی را در اختیار کاربر قرار دهیم.

هنگام نیاز به نمایش مقدار یک عبارت بر روی صفحه مرورگر اینترنت می‌توانید از تابع سیستمی ( ) print یا ( ) echo برای این‌کار استفاده کنید. اگر قصد آن را دارید تا از کد PHP تعبیه شده در سند HTML تنها برای نمایش خروجی بر روی صفحه مرورگر اینترنت استفاده کنید، می‌توانید از شکل خاصی از علامت شروع کد PHP استفاده نمایید. همان‌گونه که می‌دانید علامت ? > شروع کد PHP را در سند HTML مشخص می‌کند. حال اگر یک علامت تساوی به انتهای آن اضافه کنید، علامت حاصل یعنی = ? > را می‌توانید به‌منظور نمایش مقدار مورد نظرتان مورد استفاده قرار دهید. به بیان دیگر عبارت زیر:

```
< ? print  $test ; ? >
```

معادل این عبارت است:

```
< ? = $test ? >
```

همان‌گونه که مشاهده می‌کنید در بخش اصلی از کد PHP این برنامه جهت افزایش مقدار متغیر $num - tries از عملگر : ? استفاده شده است. بدین ترتیب اگر متغیر $num _ tries حاوی مقدار باشد، یک واحد به آن اضافه شده و حاصل مجدداً به همین متغیر منسوب می‌گردد. در صورتی‌که متغیر $num _ tries حاوی مقدار نباشد مقدار عددی صفر را به آن نسبت می‌دهیم. از این طریق در بدنه سند HTML می‌توانیم تعداد دفعاتی که کاربر اقدام به وارد کردن عدد در فیلد guess نموده را مشخص کنیم.

هرگز به فیلدهای مخفی اطمینان کامل نکنید. به خاطر داشته باشید که هیچ اطلاعی در مورد جزئیات آنها نداریم. این نکته البته به معنی عدم استفاده از آنها نیست، بلکه اخطاری است که باید از آن آگاه باشید: اگر کاربران شما به کد HTML مربوطه دسترسی داشته باشند، با کمی دستکاری در تغییر مقادیر این‌گونه فیلدها ممکن است عملکرد برنامه‌هایتان را تهدید کنند.

تغییر مسیر کاربر

برنامه اسکریپت کوچک ما هنوز یک نقطه ضعف آشکار دارد و آن این است که در صورت حدس درست یا نادرست در مورد مقدار فیلد guess ، فرم HTML مربوطه جهت نمایش مجدداً بازنویسی می‌شود. حقیقت این است که اجتناب از بازنویسی اسناد HTML شامل کد اسکریپت به‌سختی امکان‌پذیر است. با این وجود می‌توان ترتیبی داد تا فرآیند موجب تغییر مسیر کاربر شود.

هنگامی که یک برنامه سرور (مثلاً یک اسکریپت PHP) با سمت کلاینت ارتباط برقرار می‌کند ابتدا اطلاعاتی را در مورد سند مربوطه (به‌منظور تسهیل در ردیابی آن) ارسال می‌کند. معمولاً PHP این فرآیند را خود به خود انجام می‌دهد اما در صورت تمایل برنامه‌نویس می‌تواند با استفاده از تابع سیستمی، ( ) header اطلاعات مورد نظر خود را به‌جای اطلاعات پیش‌فرض PHP به سمت کلاینت ارسال کند.

جهت فراخوانی تابع ( ) header ، ابتدا باید اطمینان حاصل کنید که هیچ‌گونه خروجی بر روی مرورگر اینترنت کاربر ارسال نشده باشد. اولین باری که محتوای مورد نظر بر روی مرورگر نامبرده ارسال می‌شود PHP اطلاعات مورد بحث را که به هِدر معروف است، ارسال خواهد کرد؛ بدین ترتیب زمان مورد نظر برای ارسال از دست خواهد رفت. توجه کنید که ارسال کم‌ترین اطلاعات بر روی مرورگر (حتی یک علامت خط جدید (\n) یا یک فضای خالی " ") خارج از نشانه‌های شروع و پایان برنامه PHP موجب ارسال بی چون و چرای هِدر خواهد شد. چنان‌چه قصد استفاده از تابع سیستمی ( ) header را در برنامه PHP خود دارید، باید مطمئن شوید که پیش از فراخوانی آن کدی را که موجب ارسال اطلاعات بر روی مرورگر اینترنت می‌شود، به‌کار نبرید. حتی در این مورد باید مراقب استفاده از توابع کتابخانه‌ای نیز باشید و پیش از آنها باید امکان ارسال اطلاعات توسط آنها را دقیقاً مورد بحث قرار دهید. برنامه موجود در لیست ۱۱- ۹ اطلاعاتی را نشان می‌دهد که در قالب یک هِدر نمونه توسط PHP به مرورگر اینترنت مورد استفاده کاربر ارسال می‌شود.

```
1: HEAD /dev/php24/ch9/listing9.1.php HTTP/1.0
2:
3: HTTP/1.1 200 OK
4: Date: Mon, 24 Sep 2001 14:32:28 GMT
5: Server: Apache/1.3.12 Cobalt (Unix) PHP/4.0.6 mod_perl/1.24
6: X-Powered-By: PHP/4.0.6
7: Connection: close
8: Content-Type: text/html
```

لیست ۱۱- ۹ یک هِدر نمونه که توسط برنامه PHP ارسال می‌شود

به کمک یک برنامه telnet می‌توان هِدر ارسال شده برای کلاینت را مشاهده نمود. برای این کار با استفاده از برنامه فوق از طریق پورت شماره ۸۰ به کامپیوتر میزبان متصل شده و فرمان زیر را صادر کنید :

HEAD / path / to / file. html HTTP / 1.0

بدین ترتیب می‌توانید اطلاعات هِدر ارسالی را از طریق این برنامه مشاهده نمایید.

با تعیین موقعیت مورد نظر خود در تابع سیستمی ( ) header به جای استفاده از موقعیت پیش‌فرض PHP، به صورت زیر می‌توان موجب تغییر مسیر کاربر به یک صفحه جدید شد:

header ("Location : http: // www. Corrosive. Co. uk ") ;

برنامه موجود در لیست ۱۲- ۹ از این روش استفاده کرده و در صورتی که حدس کاربر درباره عدد مورد نظر صحیح باشد موجب تغییر مسیر وی به سندی با نام congrats. html می‌شود (سند نامبرده باید در فهرست جاری موجود باشد، در غیر این‌صورت هیچ تلاشی جهت یافتن آن صورت نخواهد گرفت).

```
 1: <?php
 2: $num_to_guess = 42;
 3: $message = "";
 4: $num_tries = ( isset( $num_tries ) ) ? ++$num_tries : 0;
 5: if ( ! isset( $guess ) )
 6:    $message = "Welcome to the guessing machine!";
 7: elseif   ( $guess > $num_to_guess )
 8:    $message = "$num is too big! Try a smaller number";
 9: elseif   ( $guess < $num_to_guess )
10:    $message = "$num is too small! Try a larger number";
11: else { // must be equivalent
12:    header( "Location: congrats.html" );
13:    exit;
14: }
15: $guess = (int)$guess;
16: ?>
17: <html>
18: <head>
19: <title>Listing 9.12 Using header() to send raw headers</title>
20: </head>
21: <body>
22: <h1>
23: <?php print $message ?>
24: </h1>
25: Guess number: <?php print $num_tries?>
26: <form method="POST">
27: Type your guess here:
28: <input type="text" name="guess" value="<?php print $guess?>">
29: <input type="hidden" name="num_tries"
30:     value="<?php print $num_tries ?>">
31: </form>
32: </body>
33: </html>
```

لیست ۱۲- ۹ بهره‌گیری از تابع سیستمی ( ) header جهت ارسال هدر

همان‌گونه که مشاهده می‌کنید، بخش else از عبارت تصمیم‌گیری if در خط ۱۱ برنامه موجب درخواست سند congrats. html جهت نمایش می‌گردد. با وجود عبارت exit در خط ۱۳ از این برنامه، نمایش هرگونه خروجی برنامه نادیده گرفته می‌شود. به‌عبارت دیگر، عبارت exit بلافاصله موجب خاتمه اجرای برنامه و جلوگیری از نمایش خروجی (ارسال اطلاعات برای کلاینت) خواهد شد.

فرم‌ها و برنامه‌های مخصوص بارگذاری فایل

تا بدین‌جا برنامه‌هایی را بررسی کردیم که به فرآیند ساده ورود اطلاعات در فرم‌ها محدود

می‌شدند. مرورگر اینترنت Netscape 2 و نسخه‌های بالاتر، همچنین مرورگر اینترنت Internet Explorer 4 و نسخه‌های بالاتر همگی از ویژگی بارگذاری (upload) فایل‌ها پشتیبانی می‌کنند. البته زبان برنامه‌نویسی PHP نیز از این فرآیند پشتیبانی به‌عمل می‌آورد. در این قسمت از درس قصد داریم تا امکانات این زبان برنامه‌نویسی را در این مورد خاص بررسی کنیم.

پیش از هرچیز لازم است تا یک فرم HTML به همین منظور ایجاد کنیم. آن دسته از فرم‌های HTML که شامل فیلدهایی جهت بارگذاری فایل هستند، لازم است که از آرگومان ENCRYPT به صورت زیر استفاده کنند:

ENCRYPT = " multipart / form - data "

PHP علاوه بر این قادر است تا از مزایای یک فیلد مخفی اختیاری که می‌توان آن را قبل از فیلد بارگذاری فایل درج نمود، بهره بگیرد. نام این فیلد مخفی در صورت استفاده باید MAX _ FILE _ SIZE باشد. مقدار این فیلد مخفی بیانگر حداکثر اندازه فایلی بر حسب بایت است که جهت بارگذاری مورد استفاده قرار خواهد گرفت. توجه داشته باشید که مقدار این فیلد نمی‌تواند از مقدار گزینه upload _ max _ filesize موجود در فایل php. ini که اندازه پیش‌فرض آن ۲ مگابایت است، متجاوز شود. مقدار فیلد MAX _ FILE _ SIZE بستگی به مرورگر اینترنت مورد استفاده دارد. از این‌رو بهتر است تا به تنظیم مقدار گزینه upload _ max _ filesize جهت اجتناب از موارد غیر متعارف در بارگذاری فایل‌ها اعتماد کنید. پس از تنظیم فیلد مذکور با مقدار صلاحدید، اکنون می‌توانید فیلد بارگذاری را به فرم HTML اضافه کنید. این کار با تعریف یک المان INPUT که آرگومان TYPE آن‌را با مقدار " file " تنظیم خواهد کرد، میسر است. انتخاب نام این المان به شما بستگی دارد. برنامه موجود در لیست ۱۳- ۹ شکل تکمیل شده این برنامه را نشان می‌دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 9.13 A simple file upload form</title>
 4: </head>
 5: <body>
 6: <form enctype="multipart/form-data" method="POST">
 7: <input type="hidden" name="MAX_FILE_SIZE" value="51200">
 8: <input type="file" name="fupload"><br>
 9: <input type="submit" value="upload!">
10: </form>
11: </body>
12: </html>
```

لیست ۱۳- ۹ فرم ساده‌ای جهت بارگذاری فایل

بار دیگر توجه شما را به این نکته جلب می‌کنیم که این فرم صفحه‌ای را که در آن قرار دارد، فراخوانی می‌کند. این بدان دلیل است که به‌زودی یک برنامه PHP جهت کنترل فرآیند بارگذاری به این صفحه اضافه خواهیم کرد. همان‌گونه که مشاهده می‌کنید در خط ۷ از این برنامه حداکثر اندازه فایل قابل بارگذاری را برابر با ۵۰ کیلوبایت تعریف کرده‌ایم. همچنین در خط ۸ فیلد بارگذاری را با نام

"fupload" معرفی کرده‌ایم. همان‌گونه که احتمالاً حدس زده‌اید، این نام‌گذاری به‌زودی اهمیت خود را نشان خواهد داد.

پس از بارگذاری موفقیت آمیز یک فایل، نامی منحصر به‌فرد به آن تخصیص یافته و در یک فهرست موقت ذخیره می‌شود. در مورد سیستم عامل UNIX فهرست موقت پیش‌فرض tmp / است. با این حال از طریق تنظیم گزینه‌ای از فایل php. ini با نام upload _ tmp _ dir می‌توان این فهرست را به‌دلخواه تغییر داد.

از طریق آرایه‌ای با نام $HTTP_ POST_ FILES می‌توان به اطلاعات مفیدی درباره فایل‌های بارگذاری شده، دسترسی پیدا کرد. این آرایه انجمنی همواره شامل مقادیری است که خود از نوع آرایه انجمنی هستند (بدین ترتیب یک آرایه چند بعدی است). کلیدهای دستیابی این آرایه انجمنی اسامی فیلدهای بارگذاری موجود در فرم HTML هستند. جزئیات مربوط به این آرایه در جدول ۲- ۹ آمده است.

با در اختیار داشتن این جدول اکنون می‌توان برنامه‌ای نوشت که اطلاعاتی را در مورد فایل‌های بارگذاری شده در اختیارمان قرار دهد. اگر فایل بارگذاری شده از نوع GIF باشد، برنامه مذکور حتی سعی خواهد کرد تا آن‌را نمایش دهد. این برنامه در لیست ۱۴- ۹ قابل بررسی است.

جدول ۲- ۹ متغیرهای سراسری شامل اطلاعات فایل بارگذاری شده

مثال	محتوا	المان
test. gif	نام فایل بارگذاری شده	$HTTP_ POST_ FILES [′ fupload ′] [′ name ′]
/tmp/ phprDfZvN	مسیر فهرست موقت	$HTTP_ POST_ FILES[′ fupload ′] [′ tmp- name ′]
6835	اندازه فایل بارگذاری شده برحسب بایت	$HTTP_ POST_ FILES[′ fupload ′] [′ size ′]
image / gif	نوع فایل بارگذاری شده	$HTTP_ POST_ FILES[′ fupload ′] [′ type ′]

```
 1: <html>
 2: <head>
 3: <title>Listing 9.14 A file upload script</title>
 4: </head>
 5: <?php
 6: $file_dir = "/home/corrdev/htdocs/php24/scrap/uploads";
 7: $file_url = "http://corros.colo.hosteurope.com/dev/php24/scrap/uploads";
 8:
 9: foreach( $HTTP_POST_FILES as $file_name => $file_array ) {
10:     print "path: ".$file_array['tmp_name']."<br>\n";
11:     print "name: ".$file_array['name']."<br>\n";
12:     print "type: ".$file_array['type']."<br>\n";
13:     print "size: ".$file_array['size']."<br>\n";
14:
15:     if ( is_uploaded_file( $file_array['tmp_name'] )
16:         && $file_array['type'] == "image/gif" ) {
17:         move_uploaded_file( $file_array['tmp_name'], "$file_dir/$file_name")
18:             or die ("Couldn't copy");
19:         print "<img src=\"$file_url/$file_name\"><p>\n\n";
20:     }
21: }
22:
23: ?>
24: <body>
25: <form enctype="multipart/form-data" method="POST">
26: <input type="hidden" name="MAX_FILE_SIZE" value="51200">
27: <input type="file" name="fupload"><br>
28: <input type="submit" value="Send file!">
29: </form>
30: </body>
31: </html>
```

لیست ۱۴- ۹ برنامه بارگذاری فایل

همان‌گونه که مشاهده می‌کنید ابتدا در خطوط ۶ و ۷ دو متغیر با نامهای $file_dir و $file_url جهت ذخیره اطلاعات مربوط به مسیر فایل مورد نظر معرفی و مقداردهی شده‌اند. در خط ۹ از برنامه، به ازای هر یک از عناصر آرایه $HTTP_POST_FILES با استفاده از ساختار تکرار foreach یک حلقه تکرار تشکیل شده است. از آنجا که به‌ازای اولین بارگذاری صفحه این آرایه تهی می‌باشد، لذا هیچ یک از دستورالعملهای بدنه حلقه به اجرا درنیامده و برنامه هیچ‌گونه مشخصاتی را به خروجی نمی‌فرستد.

به محض بارگذاری فایل (توسط کلیک دکمه مربوطه) آرایه $HTTP_POST_FILES مقداردهی می‌شود. از آنجا که همواره بر قابلیت انعطاف برنامه‌ها تاکید می‌کنیم در اینجا نیز به‌جای استفاده از دستور if ترجیح داده‌ایم تا از حلقه تکرار استفاده کنیم. بدین ترتیب می‌توانیم امکان بارگذاری چندین فایل مختلف را از طریق یک صفحه واحد در اختیار کاربران قرار دهیم. همان‌گونه که ملاحظه می‌کنید، در خط ۹ از برنامه نام فایل بارگذاری شده در متغیری با نام $file_name و اطلاعات مربوط به آن فایل در متغیر دیگری با نام $file_array در قالب ساختار تکرار foreach ذخیره می‌شوند.

بدین ترتیب می‌توانیم اطلاعات مذکور را در داخل حلقه در اختیار کاربر قرار دهیم.

جهت انتقال فایل بارگذاری شده به فهرستی بر روی وب سرور مورد نظر، ابتدا لازم است تا بررسی‌هایی را در این مورد انجام دهیم. از آن‌جا که در مثال این برنامه خاص تنها با فایل‌هایی از نوع GIF سر و کار داریم، لذا نوع آن‌را در خط ۱۶ مورد ارزیابی قرار می‌دهیم.

همچنین در خط ۱۵ به‌منظور بررسی صحت فایل مورد نظر از یک تابع جدید با نام is_ uploaded_ file ( ) استفاده کرده‌ایم. این تابع در PHP4. 03 مجددا معرفی شده است. تابع مورد بحث مسیر فایل مورد نظر را به‌عنوان آرگومان ورودی پذیرفته و در صورتی که این فایل مجاز به بارگذاری باشد، مقدار true را باز می‌گرداند. از این جهت می‌توان گفت که تابع فوق به امنیت برنامه‌ها (و وب سرور) کمک زیادی می‌کند.

با فرض اینکه همه چیز به خوبی پیش برود، در خط ۱۷ برنامه فایل مورد نظر از فهرست موقت خودبه یک فهرست جدید کپی می‌شود. ما برای این منظور از تابع دیگری با عنوان move_ uploaded_ file ( ) استفاده کرده‌ایم. این تابع قادر است تا فایلی را از یک محل به محل دیگر کپی نماید. به خاطر داشته باشید که تابع مذکور پیش از انجام این فرآیند، تابع ( )is_uploaded_file را به طور ضمنی فراخوانی خواهد کرد. تابه( )move_uploaded_file جهت انجام‌عملیات خود به آدرسهای مبدأ و مقصد نیاز دارد. در صورتی که عملیات با موفقیت انجام شود تابع فوق مقدار true و در صورتی که تابع مجاز به بارگذاری نبوده و یا تلاش جهت دستیابی به آن با ناکامی مواجه شود، مقدار false را باز می‌گرداند.

همواره مراقب اسامی فایل‌های بارگذاری شده باشید. سیستم عاملهایی چون MacOS و ویندوز در رابطه با نام‌گذاری فایل‌ها دست شما را کاملاً باز می‌گذارند. از این‌رو اسامی فایل‌های موجود تحت این سیستمهای عامل ممکن است شامل کاراکترهای بسیار متنوعی مثل فضای خالی، علامت کوتیشن، و عموماً هر نوع کاراکتر دیگری باشد. بنابراین بهتر است تا از مکانیزمی جهت فیلتر کردن اسامی فایل‌ها استفاده نمایید. در درس ساعت هجدهم با عنوان " بهره‌گیری از عبارات منظم " مطالب مربوط به بررسی دنباله‌های کاراکتری را مورد توجه قرار خواهیم داد.

همان‌گونه که در برنامه لیست ۱۴- ۹ مشاهده می‌کنید مشابه آنچه که در خط ۶ متغیر $file_ dir جهت ذخیره مسیر فایل بارگذاری شده تعریف کرده‌ایم، در خط ۷ نیز آدرس فهرست مربوطه را به شکل متداول URL در متغیر دیگری با نام $file_ url ذخیره کرده‌ایم. در برنامه فوق با استفاده از المان img سعی کرده‌ایم تا فایلی از نوع gif را بارگذاری نماییم.

جمع‌بندی

اگر تا کنون با ما همراه بوده‌اید اکنون باید تأیید کنید که مطالب جالب توجهی را فرا گرفته‌اید. با دانشی که به‌همراه دارید از ابزارهای کارآمدی جهت ایجاد برنامه‌های محاوره‌ای برخوردارید. البته مطالب دیگری را نیز در این راستا باید فرا بگیرید. اکنون که می‌توانید اطلاعاتی را از کاربر دریافت کنید، بسیار مؤثر خواهد بود اگر بتوانید عملیات مفیدی بر روی آنها انجام دهید. برای نمونه نوشتن آن اطلاعات در یک فایل بر روی دیسک می‌تواند عملیات مفیدی باشد که آن‌را در درس ساعت بعد مورد بحث و بررسی قرار می‌دهیم.

در درس این ساعت چگونگی بهره‌گیری از آرایه انجمنی $GLOBALS، دستیابی به متغیرهای سراسری، دستیابی مقادیر وارد شده در فرم‌های HTML توسط کاربر و همچنین بارگذاری فایل‌ها با استفاده از تسهیلاتی که متغیرهای سراسری در اختیارمان قرار می‌دهند را فراگرفتید. علاوه بر این با نحوه ارسال اطلاعات هِدر بر روی مرورگر اینترنت مورد استفاده کاربر و نیز نحوه تغییر مسیر کاربر آشنا شدید. در انتهای درس نیز چگونگی ثبت و ارسال اطلاعات از یک فراخوانی برنامه اسکریپت به فراخوانی دیگر را با استفاده از فیلدهای مخفی فراگرفتید.

پرسش و پاسخ

پرسش: آیا می‌توان آرایه‌ای ایجاد کرد که مقادیر آن اطلاعات وارد شده در المانی غیر از لیستها و کادرهای علامت باشند؟

پاسخ: بله، در حقیقت هر المانی که نام آن در فرم HTML به یک جفت علامت [ ] ختم شود، هنگام ارسال فرم در یکی از خانه‌های آرایه مربوطه قرار خواهد گرفت. از این واقعیت می‌توان برای گروه‌بندی مقادیر ارسال شده از چندین فیلد با انواع مختلف در یک آرایه بهره‌گرفت.

پرسش: تابع ( ) header به ظاهر تابع کارآمدی است. آیا در مورد هِدرها در درسهای آینده مطالب بیشتری عنوان خواهد شد؟

پاسخ: بحث کاملی درباره پروتکل انتقال فرامتن یا HTTP (Hypertext Transfer Protocol) را که شامل هدرها نیز می‌شود در درس ساعت سیزدهم عنوان خواهیم کرد.

پرسش: فرآیند تبدیل خودکار اسامی المان‌های یک فرم HTML به متغیرها به‌نظر کمی خطرناک می‌آید. آیا می‌توان این قاعده را در PHP غیر فعال کرد؟

پاسخ: بله. با تغییر مقدار فعلی گزینه register_ globals به " off " در فایل php. ini می‌توانید مطمئن باشید که اسامی المان‌های یک فرم HTML پس از ارسال طی یک فرآیند خودکار به

متغیرهای سراسری تبدیل نخواهند شد.

تمرینها

هدف از این بخش دوره مطالب فراگیری شده در قالب آزمون است. پاسخ بخش آزمون بلافاصله بعد از آن آمده است. بخش فعالیتها شامل تمرینهایی است که به منظور افزایش قابلیت برنامه‌نویسی خواننده طراحی شده و البته فاقد پاسخ است.

آزمون

۱- با استفاده از کدام متغیر سیستمی می‌توان آدرس IP کاربر را تشخیص داد؟

۲- با استفاده از کدام متغیر سیستمی می‌توان اطلاعاتی راجع به مرورگر مورد استفاده کاربر (مرورگری که سند HTML شامل برنامه PHP را درخواست می‌کند) به‌دست آورد؟

۳- در صورتی که بخواهیم مقادیر فیلدهای یک فرم HTML پس از ارسال آن در آرایه‌ای با نام $form_ array ذخیره شود، فیلدها را چگونه باید نام‌گذاری کنیم؟

۴- کدام آرایه انجمنی سیستمی شامل تمامی مقادیر یک فرم HTML است که از طریق درخواست GET به وب سرور ارسال شده است؟

۵- کدام آرایه انجمنی سیستمی شامل تمامی مقادیر یک فرم HTML است که از طریق درخواست POST به وب سرور ارسال شده است؟

۶- از کدام تابع سیستمی می‌توان جهت تغییر مسیر کاربر به یک صفحه جدید استفاده کرد؟ چه دنباله کاراکتری را باید به این تابع به عنوان آرگومان ارسال کرد؟

۷- چگونه می‌توان اندازه فایلی را که کاربر می‌تواند از طریق یک فرم HTML بخصوص و به‌واسطه یک برنامه PHP آن‌را بارگذاری کند، محدود کرد؟

۸- چگونه می‌توان اندازه فایل‌هایی را که کاربر می‌تواند از طریق فرم‌های مختلف HTML و به‌واسطه برنامه‌های مختلف PHP آنها را بارگذاری کند، محدود کرد؟

پاسخ آزمون

۱- آدرس IP کاربر همواره از طریق متغیر $REMOTE_ ADDR قابل دسترسی است.

۲- اطلاعات مربوط به مرورگر اینترنت مورد استفاده کاربر شامل نوع و شماره نسخه و همچنین سیستم عامل مورد استفاده وی معمولاً از طریق متغیر $HTTP_ USER_ AGENT قابل دسترسی هستند.

۳- ایجادفیلدهای مختلف با نام واحد form_ array [ ] موجب ایجاد آرایه‌ای با نام $form_ array خواهد شد که مقادیر آن‌را مقادیر فیلدهای یاد شده تشکیل می‌دهند.

۴- آرایه سیستمی $HTTP_ GET_ VARS ، شامل تمامی مقادیری است که به‌عنوان درخواستی از نوع GET به وب سرور ارسال شده‌اند.

۵- آرایه سیستمی $HTTP_ POST_ VARS، شامل تمامی مقادیری است که به‌عنوان درخواستی از نوع POST به وب سرور ارسال شده‌اند.

۶- تابع سیستمی ( ) header را می‌توان جهت تغییر مسیر کاربر به یک صفحه جدید مورد استفاده قرار داد. دنباله کاراکتری ارسالی به این تابع به‌عنوان آرگومان باید شامل واژه LOCATION و به‌دنبال آن نام صفحه مورد نظر باشد:

```
Header (" LOCATION: anotherpage. html ");
```

۷- هنگام ایجاد فرم‌هایی با PHP4 که قابلیت بارگذاری فایل‌ها را داشته باشند می‌توان با تعریف یک فیلد مخفی با نام MAX_ FIELD_ SIZE به‌صورت زیر اندازه فایل‌های قابل بارگذاری را محدود کرد:

```
< INPUT TYPE = " hidden " NAME = " MAX_ FIELD_ SIZE "
  VALUE = " 51200 " >
```

۸- با تنظیم گزینه upload_ max_ filesize در فایل php. ini می‌توان اندازه فایل‌های قابل بارگذاری از طریق تمامی فرم‌های HTML را محدود کرد. محدودیت پیش‌فرض در این مورد ۲ مگابایت است.

فعالیتها

۱- برنامه‌ای جهت شبیه‌سازی عملکرد ماشین‌حساب بنویسید. برنامه باید بتواند دو عدد و نوع عملیات مورد نظر را که یکی از چهار عمل اصلی است از ورودی دریافت کند.

۲- با استفاده از یک فیلد مخفی ترتیبی دهید تا کاربر از تعداد دفعاتی که در تمرین قبل از اقدام به محاسبه کرده است، اطلاع حاصل نمایید.

ساعت دهم

بهره‌گیری از فایل‌ها

عملیاتی چون بررسی، بازخوانی و باز نویسی فایل‌ها از جمله عملیات اساسی است که هر زبان برنامه‌نویسی باید تسهیلاتی را در این رابطه فراهم نماید و زبان برنامه‌نویسی PHP نیز در این مورد استثنا نمی‌باشد. در حقیقت زبان PHP توابعی را در اختیار برنامه‌نویس قرار می‌دهد که فرآیند کار با فایل‌ها را بسیار ساده و سرراست می‌کند. در درس این ساعت مطالب زیر را در مورد فایل‌ها مورد بررسی قرار می‌دهیم:

- **چگونگی شامل کردن فایل‌ها در اسناد PHP**
- **چگونگی کسب اطلاعات مختلف در مورد فایل‌ها و فهرستها**
- **چگونگی باز کردن یک فایل پیش از کار با آن**
- **چگونگی خواندن اطلاعات از یک فایل**
- **چگونگی نوشتن اطلاعات در یک فایل**
- **چگونگی قفل کردن یک فایل**
- **چگونگی کار با فهرستها**

در ادامه به بررسی این مطالب خواهیم پرداخت.

شامل‌کردن فایل‌ها در اسناد PHP با استفاده از تابع سیستمی include ( )

با استفاده از تابع سیستمی include ( ) می‌توان فایل‌های مورد نظر را در اسناد PHP درج نمود. کد PHP موجود در این‌گونه فایل‌ها دقیقاً مشابه کد اصلی اجرا خواهد شد. این ویژگی جهت استفاده از امکانات کتابخانه‌های مختلف در برنامه‌های PHP بسیار مفید است.

تابعی را تصور کنید که کارایی آن فوق‌العاده است. چنانچه بخواهید از این تابع در چندین برنامه PHP بهره‌مند شوید تنها گزینه‌ای که تا کنون به ذهنتان می‌رسد احتمالاً کپی کردن کد این تابع در برنامه‌های مختلف PHP است. حال وضعیتی را در نظر بگیرید که پس از گذشت مدتی وجود یک اشکال کوچک در عملکرد این تابع برای شما آشکار شده باشد، یا اینکه مایل باشید تا ویژگی جدیدی به عملکرد این تابع اضافه کنید. در این صورت مجبورید تا این تغییر را در تمامی برنامه‌های PHP که از این تابع استفاده می‌کنند، اعمال نمایید. تابع سیستمی include ( ) در این‌گونه مواقع می‌تواند زحمت چنین تغییر گسترده‌ای را کاهش دهد. به کمک این قابلیت می‌توانید به هنگام اجرای یک برنامه PHP فایل دیگری را که آن هم شامل کد PHP است، وارد بازی کنید. تابع سیستمی include( ) جهت اجرا تنها به یک آرگومان نیاز دارد. این آرگومان موقعیت فایل PHP دیگری را مشخص می‌کند که مایلیم تا آن را وارد برنامه PHP اصلی کنیم. برنامه موجود در لیست ۱۰- ۱۰ یک اسکریپت PHP ساده را که با استفاده از تابع سیستمی include ( ) فایل PHP دیگری را مورد استفاده قرار داده است، نشان می‌دهد.

```
1: <html>
2: <head>
3: <title>Listing 10.1 Using include()</title>
4: </head>
5: <body>
6: <?php
7: include("listing10.2.php");
8: ?>
9: </body>
10: </html>
```

لیست ۱- ۱۰ چگونگی استفاده از تابع سیستمی include ( )

همان‌گونه که مشاهده می‌کنید، تابع include ( ) در خط ۷ از این برنامه سند دیگری از نوع PHP با عنوان listing 10.2. php را که محتوای آن در لیست ۲- ۱۰ آمده، مورد استفاده قرار می‌دهد. با اجرای برنامه لیست ۱- ۱۰ دنباله کاراکتری " I have been included !! " بر روی صفحه به نمایش در می‌آید. این‌گونه استفاده از تابع include ( )، یعنی شامل کردن یک فایل دیگر PHP جهت نمایش یک پیغام ساده بر روی صفحه، معمولاً استفاده‌ای نیست که به‌دنبال آن باشیم. اما این مثال در نوع

خود جهت نشان دادن قابلیت تابع مورد بحث کافی است. در این‌جا ممکن است پرسشی مطرح شود: آیا نمایش متن ساده فوق موجب هیچ خطایی در برنامه اصلی نمی‌شود؟ حقیقت این است که بنا به پیش‌فرض، محتوای فایل شامل شده به‌عنوان یک سند HTML در خروجی برنامه اصلی به نمایش در می‌آید.

اگر بخواهیم کد PHP موجود در برنامه فرعی را به اجرا درآوریم، لازم است تا محتوای آن را نیز توسط علایم شروع و پایان کد PHP نشانه‌گذاری کنیم.

```
1: I have been included!!
```

لیست ۲- ۱۰ فایل شامل شده در برنامه PHP لیست ۱- ۱۰

در برنامه لیست ۳- ۱۰ همان برنامه لیست ۱- ۱۰ را مجددا تکرار کردیم، با این تفاوت که فایل PHP شامل شده این بار در لیست ۴- ۱۰ به‌گونه‌ای توسعه یافته که در برنامه اصلی اجرا شود.

```
1: <html>
2: <head>
3: <title>Listing 10.3 Using include to execute PHP in another file</title>
4: </head>
5: <body>
6: <?php
7: include("listing10.4.php");
8: ?>
9: </body>
10: </html>
```

لیست ۳- ۱۰ استفاده از تابع ( ) include جهت اجرای کد PHP یک فایل مجزا

```
1: <?php
2: print "I have been included!!<BR>";
3: print "But now I can add up...  4 + 4 = ".(4 + 4);
4: ?>
```

لیست ۴- ۱۰ فایل PHP شامل شده در برنامه لیست ۳- ۱۰ که حاوی کد PHP می‌باشد

دستیابی به مقدار بازگشتی از یک سند شامل شده در برنامه اصلی

در زبان PHP4 فایل‌های شامل شده در برنامه‌های PHP به مانند توابع می‌توانند مقادیری را به‌عنوان نتیجه عملیات به برنامه اصلی بازگردانند. همان‌گونه که در مورد توابع نیز شاهد آن بودیم، دستورالعمل return قادر است تا به اجرای برنامه‌های شامل شده در برنامه اصلی خاتمه دهد. جالب اینکه در این فرآیند کد HTML دخیل نمی‌باشد. برنامه موجود در لیست ۵- ۱۰ فایل موجود در لیست ۶- ۱۰ را که شامل بخش HTML نیز می‌باشد، مورد استفاده قرار می‌دهد. با اجرای دستورالعمل return در فایل ۶- ۱۰ خاتمه آن فرا رسیده و مابقی سند که شامل کد HTML است، نادیده گرفته می‌شود.

```
 1: <html>
 2: <head>
 3: <title>Listing 10.5 Using include() to execute PHP and assign the return
➥value</title>
 4: </head>
 5: <body>
 6: <?php
 7: $addResult = include("listing10.6.php");
 8: print "The include file returned $addResult";
 9: ?>
10: </body>
11: </html>
```

لیست ۵- ۱۰ بهره‌گیری از تابع ( ) include جهت اجرای برنامه PHP لیست ۶- ۱۰ و نسبت‌دهی نتیجه به یک متغیر

```
1: <?php
2: $retval = ( 4 + 4 );
3: return $retval;
4: ?>
5: This HTML should never be displayed because it comes after a return
➥statement!
```

لیست ۶- ۱۰ کد PHP مورد استفاده لیست ۵- ۱۰ که مقداری را نیز به‌عنوان نتیجه عملیات باز می‌گرداند

توجه کنید که مقدار بازگشتی از فایل‌های شامل شده در برنامه اصلی در زبان PHP3 تنها هنگامی به برنامه مذکور باز می‌گردد که دستورالعمل return مورد استفاده در فایل شامل شده در درون تابع مورد استفاده قرار بگیرد. به‌عبارت دیگر، برنامه PHP لیست ۶- ۱۰ در صورت استفاده از نسخه PHP3 موجب یک خطا خواهد شد.

استفاده از تابع سیستمی ( ) include در درون ساختارهای کنترلی

از تابع سیستمی include ( ) می‌توان در درون ساختار تصمیم‌گیری نیز استفاده کرد. بدین ترتیب فایل شامل شده تنها در صورتی مورد بازخوانی و استفاده برنامه اصلی قرار می‌گیرد که عبارت شرطی ساختار تصمیم‌گیری true ارزیابی شود. برای نمونه، فایل شامل شده در قطعه کد زیر هرگز مورد توجه برنامه اصلی قرار نخواهد گرفت:

```
$test = false ;
if ( $test ) {
   include ( "a _ file. txt") ; // won't be included
}
```

از تابع سیستمی ( ) include در ساختارهای تکرار نیز می‌توان استفاده کرد. در این‌صورت فراخوانی این تابع در هر بار گذر از حلقه یا فایل شامل شده توسط این تابع جایگزین خواهد شد. بدین ترتیب دستورالعملهای فایل شامل شده به ازای هر بار گذر از حلقه اجرا می‌شود. در برنامه لیست ۷- ۱۰، تابع سیستمی ( ) include در یک ساختار تکرار for مورد استفاده قرار گرفته است. واضح است که تابع مذکور در هر فراخوانی فایلی را در برنامه شامل می‌کند که با سایر فراخوانی‌ها متفاوت است.

```
 1: <html>
 2: <head>
 3: <title>Listing 10.7 Using include() within a loop</title>
 4: </head>
 5: <body>
 6: <?php
 7: for ( $x = 1; $x<=3; $x++ ) {
 8:    $incfile = "incfile$x".".txt";
 9:    print "Attempting include $incfile<br>";
10:    include( "$incfile" );
11:    print "<p>";
12: }
13: ?>
14: </body>
15: </html>
```

لیست ۷- ۱۰ بهره‌گیری از تابع ( ) include در درون ساختار تکرار

همان‌گونه که مشاهده می‌کنید، با اجرای این برنامه محتوای سه فایل مختلف incfile1. txt، incfile2. txt و incfile3. txt در برنامه اصلی شامل خواهند شد. با فرض اینکه هریک از این فایل‌ها شامل متن ساده‌ای باشند که نام خود را مشخص می‌کنند، خروجی برنامه به شکل زیر خواهد بود:

```
Attempting include incfile1. txt
This is incfile1. txt

Attempting include incfile2. txt
This is incfile2. txt

Attempting include incfile3. txt
This is incfile3. txt
```

تابع سیستمی include _ once ( )

یکی از مشکلات ناشی از استفاده از چندین کتابخانه مختلف در برنامه هنگامی برملا می‌شود که تابع ( ) include به تعداد دو بار در یک برنامه فراخوانی شود. این وضعیت در پروژه‌های برنامه‌نویسی بزرگ و معمولاً هنگامی که فایل‌های کتابخانه‌ای مختلف تابع ( ) include را جهت شامل کردن یک فایل واحد فراخوانی می‌کنند، نمایان می‌شود. شامل کردن دوباره یک فایل معمولاً باعث تکرار معرفی توابع و کلاسهای موجود در آن فایل می‌شود. موضوعی که موتور PHP را آشکارا به زحمت می‌اندازد.

این وضعیت با استفاده از تابع سیستمی دیگری با نام ( ) include _ once به‌خوبی قابل حل و فصل است. این تابع جهت اجرا مسیر فایل مورد نظر را به‌عنوان آرگومان پذیرفته و اول باری که فراخوانی می‌شود رفتاری مشابه تابع سیستمی ( ) include ارائه می‌دهد. این در حالی‌است که اگر این تابع برای دومین بار به‌منظور شامل کردن همان فایل قبلی فراخوانی شود، فایل مذکور برای بار دوم در برنامه اصلی (برنامه فراخواننده تابع ( ) include _ once) شامل نخواهد شد.

این شرایط تابع ( ) include _ once را به ابزار مناسبی برای ایجاد کتابخانه‌ها تبدیل کرده است.

گزینه include _ path

بهره‌گیری از توابع سیستمی ( ) include و ( ) include _ once جهت دستیابی به کتابخانه‌ها به میزان قابل توجهی به قابلیت انعطاف و استفاده مجدد برنامه‌ها کمک می‌کند. با این حال مسائل دیگری وجود دارد که باید از وجود آنها آگاه باشید. در این میان به‌ویژه مسأله قابلیت حمل در صورت ذکر مسیر فایل‌ها در برنامه می‌تواند، مسأله‌ساز باشد. فرض کنید که فهرستی با نام Lib ایجاد کرده و فایل یا فایل‌های موجود در آن را در سراسر برنامه‌تان شامل کرده‌اید. نمونه زیر می‌تواند اقدامی برای این کار تصور شود:

```
include _ once ( "/ home / user / bob / htdocs / project4 / lib / mylib. inc. php") ;
```

با این‌حال هنگامی که برنامه خود را به سرور جدیدی منتقل می‌کنید خود را در وضعیتی می‌یابید که باید صدها و بلکه تعداد بیشتری از آدرس فایل یا فایل‌های شامل در برنامه اصلی را به آدرس جدید تغییر دهید. با تنظیم مناسب گزینه include _ path در فایل php. ini می‌توانید از دردسر تغییر آدرس خلاص شوید :

```
include _ path . : / home / user / bob / htdocs / project4 / lib /
```

مقدار گزینه include _ path را با هر تعداد فهرست که با علامت کولون (:) از یکدیگر جدا شده باشند، می‌توانید تنظیم کنید (علامت جداکننده در سیستم عامل ویندوز سمی کولون است). با انجام چنین تنظیمی اکنون می‌توانید فایل کتابخانه‌ای خود را تنها با ذکر نام آن در تابع ( ) include یا include _ once مورد استفاده قرار دهید:

```
include _ once ( "mylib. inc. php ") ;
```

اکنون هنگام انتقال برنامه‌تان بر روی یک سرور جدید تنها کافی است تا مقدار این گزینه را باردیگر تنظیم نمایید.

در زبان PHP4 تابع سیستمی دیگری با نام ( ) require معرفی شده که عملکردی مشابه تابع سیستمی ( ) include دارد. همچنین تابع سیستمی دیگری معرفی شده که عملکرد آن مشابه تابع سیستمی ( ) include _ once است. نام این تابع ( ) require_once است.
نکته قابل توجه در مورد تابع ( ) require ، این است که اجرای آن هیچ‌گونه ارتباطی با روند اجرای برنامه و شرایط اجرایی ندارد. از این‌رو معمولاً از تابع فوق در داخل ساختارهای تصمیم‌گیری یا تکرار استفاده نمی‌شود.
به‌خاطر داشته باشید که فایلی که به‌عنوان نتیجه فراخوانی تابع ( ) require در برنامه اصلی شامل شده باشد، نمی‌تواند مقداری را به آن برنامه باز گرداند. به عدم وجود این محدودیت در مورد تابع ( ) include توجه نمایید.

بررسی فایل‌ها

پیش از هرگونه اقدامی جهت کار با یک فایل یا فهرست، بهتر است اطلاعات بیشتری در مورد آن داشته باشیم. زبان PHP4 توابعی را در اختیار برنامه‌نویس قرار داده که وی با استفاده از آنها می‌تواند اطلاعات مفیدی در مورد فایل‌های موجود بر روی سیستم به‌دست آورد. در این قسمت به‌طور خلاصه به بررسی مهم‌ترین آنها می‌پردازیم.

بررسی وجودی فایل‌ها با استفاده از تابع ( ) file _ exists

با استفاده از تابع سیستمی ( ) file _ exists ، می‌توان از وجود یک فایل اطمینان حاصل کرد. آرگومان این تابع یک دنباله کاراکتری است که بیانگر مسیر مطلق یا نسبی فایل مورد نظر می‌باشد. چنانچه فایل در محل مذکور یافت شود، این تابع مقدار true را به برنامه فراخواننده باز می‌گرداند. در غیر این‌صورت مقدار بازگشتی false خواهد بود. به نمونه‌ای از کاربرد این تابع توجه کنید:

```
if ( file _ exists ( "test. txt" ) )
   print "The file exists !" ;
```

تشخیص فایل از فهرست

با استفاده از تابع ( ) file _ is در برنامه‌های PHP می‌توانید فایل‌ها را از فهرستها تشخیص دهید. تابع فوق مسیر فایل یا فهرست مورد بررسی را به‌عنوان آرگومان پذیرفته و یکی از دو مقدار true

یا false را باز می‌گرداند. به نمونه‌ای از کاربرد این تابع توجه کنید:

```
if ( is _ file ("/ test.txt") )
   print " test.txt is a file!" ;
```

در قطعه برنامه فوق جهت بررسی اینکه آیا دنباله کارکتری test.txt یک فایل است یا خیر از تابع is_file استفاده شده است . تابع مشابهی به نام ( )is_dir وجود دارد که به کمک آن می‌توان همین بررسی را در مورد فهرستها نیز انجام داد . ایت تابع نیز مسیر فایل یا فهرست مورد بررسی را به عنوان آرگومان پذیرفته و یکی از دو مقدار true یا false را باز می‌گرداند. نمونه کد زیر نحوه استفاده از این تابع را نشان می‌دهد:

```
If(is_dir("/tmp"))
Print "/tmp is a dirctory";
```

بررسی وضعیت یک فایل

پس از اطمینان از وجود فایل و نیز اطمینان از ماهیت فایل بودن آن با استفاده از توابعی که در قسمتهای قبل بررسی شده می‌توان اطلاعات بیشتری را نیز در مورد وضعیت آن به‌دست آورد. معمولاً در مورد فایل‌ها یکی از سه عملیات خواندن، نوشتن و اجرا کردن مورد نظر می‌باشد. در زبان PHP توابعی برای بررسی امکان انجام هر یک از عملیات فوق بر روی یک فایل پیش‌بینی شده است.

با استفاده از تابع ( ) is _ readable می‌توان تشخیص داد که آیا فایل مورد بررسی قابل خواندن است یا خیر. در سیستمهای UNIX ممکن است بتوان فایلی را مشاهده کرد اما خواندن آن چیزی است که بستگی به مجوزهای اعطایی از جانب مالک فایل دارد. تابع سیستمی ( ) is _ readable مسیر فایل مورد بررسی را در قالب یک دنباله کاراکتری پذیرفته و یکی از دو مقدار true یا false را باز می‌گرداند. به نمونه‌ای از کاربرد آن توجه کنید:

```
if (is _ readable ("test. txt") )
   print "test. txt is readable" ;
```

به‌طور مشابه، تابع سیستمی ( ) is _ writable امکان نوشتن در یک فایل را مورد بررسی قرار می‌دهد. بار دیگر، تابع فوق مسیر فایل مورد بررسی را در قالب یک دنباله کاراکتری پذیرفته و یکی از دو مقدار true یا false را باز می‌گرداند. به نمونه‌ای از کاربرد این فایل توجه کنید:

```
if (is _ writable ("test. txt") )
   print "test. txt is writable" ;
```

با استفاده از تابع سیستمی دیگری با نام ( ) is _ executable می‌توان امکان قابل اجرا بودن فایل مورد بررسی را با توجه به مجوز اجرایی آن یا امکان اجرا کردن آن بر روی محیط عامل تشخیص داد. این تابع سیستمی نیز به مانند دو تابع سیستمی فوق مسیر فایل مورد بررسی را در قالب یک دنباله کاراکتری دریافت کرده و یکی از دو مقدار true یا false را باز می‌گرداند. به نمونه‌ای از کاربرد

این فایل توجه کنید:

```
if (is _ executable ("test. txt") )
   print "test. txt is executable" ;
```

تشخیص اندازه فایل با استفاده از تابع ( ) filesize

تابع ( ) filesize مسیر فایلی را به‌عنوان آرگومان پذیرفته و اندازه آن را برحسب بایت بازمی‌گرداند. در صورت بروز مشکل، این تابع مقدار false را باز می‌گرداند. به نمونه‌ای از کاربرد آن توجه کنید:

```
print "The size of test. txt is ... " ;
print filesize (" test. txt") ;
```

دستیابی به اطلاعات بیشتر درباره فایل‌ها

گاهی لازم است تا از زمان آخرین دستیابی یا نوشتن در یک فایل اطلاع داشته باشیم. در زبان PHP توابعی پیش‌بینی شده که امکان انجام این‌گونه بررسی‌ها را در اختیارمان قرار می‌دهد.

با استفاده از تابع ( ) file a time، می‌توانیم زمان آخرین دستیابی به یک فایل را مشخص کنیم. این تابع مسیر فایل مورد نظر را به‌عنوان آرگومان پذیرفته و تاریخی را که آخرین بار فایل مذکور مورد دستیابی قرار گرفته، باز می‌گرداند. توجه کنید که دستیابی فایل به معنی خواندن یا نوشتن فایل می‌باشد. تاریخ بازگردانده شده از تمامی این توابع قالب خاصی با عنوان " UNIX epoch format " دارد. این قالب خاص تاریخ را برحسب تعداد ثانیه‌های سپری شده از نیمه شب اولین روز ژانویه سال ۱۹۷۰ میلادی بیان می‌کند. تابع سیستمی ( ) date قادر است تا این قالب خاص را به تاریخ متداولی که برای ما آشناست، تبدیل کند. در درس ساعت پانزدهم با عنوان " بهره‌گیری از تاریخ و ساعت " مطالب بیشتری درباره این تابع فرا می‌گیرید. اکنون به نمونه‌ای از کاربرد تابع ( ) file a time توجه کنید:

```
$atime = file atime ("test. txt") ;
print "test. txt was last accessed on " ;
print date ( "D d M Y g : I A", $atime) ;
// sampleoutput : The 13 Jan 2000 2 : 26 PM
```

با بهره‌گیری از تابع ( ) file m time، می‌توان تاریخ دستکاری یک فایل را تغییر داد. برای انجام این کار علاوه بر نام (مسیر) تابع، تاریخ مورد نظر را نیز در قالب UNIX باید به‌عنوان آرگومان این تابع مورد استفاده قرار دهید. دستکاری فایل معمولاً به معنی تغییر محتوای آن می‌باشد. به نمونه زیر توجه کنید:

```
$mtime = filemtime (" test. txt ") ;
print " test. txt was last modified on " ;
print date ( "D d M Y g : i A ", $mtime) ;
// sample output : Thu 13 Jan 2000 2 : 26 PM
```

علاوه بر موارد فوق با استفاده از تابع ( )Filectime می‌توان تاریخ تغییر یک سند را مشخص نمود. در سیستمهای UNIX تاریخ تغییر یک سند هنگامی تنظیم می‌شود که محتوای فایل مربوطه را تغییر دهد و یا مالکیت یا مجوزهای اعطایی در رابطه با آن دستخوش تغییر شوند. در محیطهای عامل دیگر، مقدار بازگشتی از این تابع تاریخ ایجاد فایل مورد بررسی را مشخص می‌کند به نمونه‌ای از این کاربرد این تابع توجه کنید:

```
$ctime=filectime("test.txt"):
print "test. Txt was last changed on" ;
print date ("D d M Y g : "A" , $ctime) ;
//sample output : thu 13 jan 2000 2:26 PM]
```

ایجاد تابعی برای انجام چندین بررسی بر روی فایل

برنامه موجود در لیست ۸- ۱۰ تابعی را تعریف می‌کند که قابلیت انجام بررسی‌های مختلفی بر روی یک فایل را به‌طور یک‌جا در اختیارمان قرار می‌دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 10.8 A function to output the results of multiple file
➥tests</title>
 4: </head>
 5: <body>
 6: <?php
 7: $file = "test.txt";
 8: outputFileTestInfo( $file );
 9:
10: function outputFileTestInfo( $f ) {
11:     if ( ! file_exists( $f ) ) {
12:         print "$f does not exist<BR>";
13:         return;
14:     }
15:     print "$f is ".(is_file( $f )?"":"not ")."a file<br>";
16:     print "$f is ".(is_dir( $f )?"":"not ")."a directory<br>";
17:     print "$f is ".(is_readable( $f )?"":"not ")."readable<br>";
18:     print "$f is ".(is_writable( $f )?"":"not ")."writable<br>";
19:     print "$f is ".(is_executable( $f )?"":"not ")."executable<br>";
20:     print "$f is ".(filesize($f))." bytes<br>";
21:     print "$f was accessed on ".date( "D d M Y g:i A", fileatime( $f )
➥)."<br>";
22:     print "$f was modified on ".date( "D d M Y g:i A", filemtime( $f )
)."<br>";
23:     print "$f was changed on ".date( "D d M Y g:i A", filectime( $f )
➥)."<br>";
24: }
25:
26: ?>
27: </body>
28: </html>
```

لیست ۸- ۱۰ تابعی که خروجی آن نتیجه بررسی‌های مختلف بر روی یک فایل است

همان‌گونه که ملاحظه می‌کنید در این برنامه به‌منظور خلاصه‌نویسی سعی کرده‌ایم تا از عملگر : ? جهت بررسی‌های مورد نیاز استفاده کنیم. اجازه دهید تا به یکی از این بررسی‌ها که در خط ۱۵ آمده است، نگاه دقیق‌تری بیندازیم:

```
print "$f is " . (is _ file ($f) ? " " : "not" ) . "a file < br > " ;
```

در عبارت فوق از تابع ( ) is _ file به‌عنوان عبارت شرطی عملگر : ? استفاده کرده‌ایم. چنانچه این تابع مقدار true را بازگرداند، یک دنباله کاراکتری تهی (" ") و در صورتی که مقدار false را بازگرداند، دنباله کاراکتری " not " مورد استفاده تابع print قرار خواهد گرفت. مقدار حاصل از عملگر:? (یکی از دو دنباله کاراکتری نامبرده) با استفاده از عملگر ترکیب دنباله‌های کاراکتری (.) به دنباله کاراکتری مورد استفاده در تابع print اضافه می‌شود. عبارت فوق را می‌توان با وضوح بیشتری اما به بهای از دست رفتن خلاصه نویسی به صورت زیر نیز نوشت:

```
$is _ it = is _ file ($f) ? " " : "not" ;
print "$f is $is  it a file " ;
```

حتی می‌توان خلاصه نویسی را به‌کل فراموش کرده و عبارت فوق را در قالب یک ساختار تصمیم‌گیری if / else در نهایت وضوح به‌صورت زیر بازنویسی کرد:

```
if (is _ file ($f) )
   print "$fi is a file < br > " ;
else
   print "$f is not a file < br > " ;
```

از آنجا که تاثیر هر سه روش فوق در نهایت یکسان می‌باشد، اینکه از کدام‌یک از آنها بهتر است استفاده شود، تصمیمی است که به برنامه‌نویس بستگی دارد.

ایجاد و حذف فایل‌ها

چنانچه فایلی در حال حاضر موجود نباشد با استفاده از تابع سیستمی ( ) touch می‌توان آن‌را ایجاد نمود. تابع فوق با دریافت مسیر فایل مورد نظر (که نام آن را نیز شامل می‌شود) فایلی تهی و همنام با آنچه در آرگون خود دریافت کرده است، ایجاد می‌کند. چنانچه فایل مذکور در حال حاضر در مسیر تعیین شده موجود باشد محتوای آن دست‌نخورده باقی مانده، اما تاریخ دستکاری آن به تاریخ فعلی تغییر خواهد کرد. به نمونه‌ای از کاربرد این تابع توجه کنید:

```
Touch (" myfile. txt ") ;
```

عملکرد معکوس، یعنی حذف یک فایل موجود با استفاده از تابع سیستمی ( ) unlike امکان‌پذیر است. بار دیگر مسیر فایل مورد نظر به‌عنوان آرگومان مورد استفاده تابع قرار می‌گیرد. به نمونه زیر توجه کنید:

```
unlike(" myfile. txt ") ;
```

توجه داشته باشید که در سیستم UNIX عملیات ایجاد، حذف، خواندن، نوشتن و تغییر دادن فایل‌ها و فهرستها مستلزم در دست داشتن مجوزهای مربوطه است.

بازکردن فایل جهت خواندن، نوشتن یا اضافه کردن محتوا

پیش از کار با هر فایلی ابتدا لازم است تا آن‌را باز کنید. در زبان برنامه‌نویسی PHP تابعی با عنوان ( ) fopen امکان باز کردن فایل را پیش از نوشتن یا خواندن آن در اختیار برنامه‌نویس قرار می‌دهد. این تابع مستلزم دو آرگومان است. آرگومان اول فایل مورد نظر جهت بازکردن را مشخص می‌کند. آرگومان دوم عملیاتی را مشخص می‌نماید که پس از باز شدن فایل می‌توان آن‌را در مورد فایل اجرا کرد. سه نوع عملیات موجود در این رابطه عبارتند از خواندن، نوشتن و اضافه کردن محتوا به فایل که به ترتیب با دنباله‌های کاراکتری ' r ', ' w ' و ' a ' مشخص می‌شوند. لازم به یادآوری است که هر دو آرگومان تابع ( ) fopen از نوع دنباله کاراکتری هستند. در صورت موفقیت‌آمیز بودن عملیات، تابع ( ) fopen مرجعی را به فایل باز شده بازمی‌گرداند که می‌توان از آن در برنامه جهت اشاره به فایل نامبرده بهره گرفت. عبارت زیر چگونگی بازکردن فایل نمونه‌ای را جهت خواندن محتوای آن نشان می‌دهد:

```
$fp = fopen ( "test. txt" , ' r ' ) ;
```

عبارت زیر چگونگی بازکردن فایل نمونه را جهت نوشتن (تغییر) محتوای آن نشان می‌دهد:

```
$fp = fopen ( "test. txt" , ' w ' ) ;
```

و بالاخره عبارت زیر چگونگی باز کردن فایل مورد بحث را جهت اضافه کردن محتوای جدید به محتوای موجود فایل نشان می‌دهد:

```
$fp = fopen ( "test. txt" , ' a ' ) ;
```

اگر به هر دلیلی عملیات بازکردن فایل با شکست مواجه شود، تابع ( ) fopen مقدار false را باز می‌گرداند. از این‌جهت همواره بهتر است پیش از آغاز به کار بر روی فایل مورد نظر، موفقیت آمیز بودن عملیات تابع ( ) fopen مورد بررسی قرار بگیرد. این‌کار با استفاده از ساختار شرطی if به صورت زیر قابل پیاده سازی است:

```
if ($fp = fopen ("test. txt" , " w " ) ) {
   // do something with $fp
}
```

همچنین می‌توان به‌صورت زیر در حالتی که عملیات تابع ( ) fopen موفقیت آمیز نباشد، به اجرای برنامه خاتمه داد:

```
($fp = fopen ("test. txt", " w " ) ) or die ("couldn't open file, sorry") ;
```

در عبارت فوق، اگر تابع( ) fopen مقدار true را بازگرداند، باقیمانده عبارت از or به بعد مورد ارزیابی قرار نگرفته و بدین ترتیب تابع سیستمی ( ) die که موجب نمایش یک پیغام در پنجره مرورگر

خاتمه دادن به برنامه می‌شود، اجرا نخواهد شد. در غیر این‌صورت باقیمانده عبارت فوق از or به بعد اجرا شده و تابع ( ) die فراخوانی خواهد شد که این به معنی خاتمه اجرای برنامه است. با فرض اینکه کل عملیات، شامل بازکردن فایل و کار با آن با موفقیت انجام شده باشد، نباید فراموش کنید که فایل را در انتهای کار ببندید. این فرآیند با استفاده از تابع سیستمی ( ) fclose قابل پیاده‌سازی است. آرگومان این تابع مرجعی است که تابع ( ) fopen در صورت موفقیت آمیز بودن عملیات باز می‌گرداند:

```
Fclose ($fp) ;
```

خواندن محتوای فایل

در زبان PHP توابع سیستمی متعددی جهت خواندن محتوای فایل‌ها پیش‌بینی شده است. با استفاده از این توابع می‌توانیم داده‌ها را برمبنای بایت، کاراکتر و یا خط به خط از فایل‌ها بخوانیم.

خواندن خط به خط فایل با استفاده از توابع ( ) fgets و ( ) feof

پس از بازکردن یک فایل جهت خواندن، اغلب لازم است تا به صورت خط به خط آن‌را مورد دستیابی قرار دهیم. با استفاده از تابع سیستمی ( ) fgets می‌توان خطی از یک فایل باز شده را خواند. آرگومان مورد نیاز این تابع مرجع بازگشتی از تابع ( ) fopen است که جهت بازکردن فایل آن‌را پیشتر فراخوانی کرده‌ایم. دومین آرگومان این تابع یک عدد صحیح است. این عدد صحیح تعداد بایت‌هایی را که تابع ( ) fgets باید از فایل مورد نظر بخواند، مشخص می‌کند البته به‌شرطی که به انتهای خط یا به انتهای فایل نرسد. تابع ( ) fgets تاجایی به خواندن محتوای فایل ادامه می‌دهد که با نشانه خط جدید ("\n") مواجه شود یا اینکه تعداد بایت‌هایی را که به‌عنوان آرگومان دوم مشخص شده است، مورد بازخوانی قرار دهد و یا اینکه تابع مذکور به انتهای فایل برسد. به نمونه‌ای از کاربرد این تابع توجه کنید:

```
$line = fgets ($fp, 1024) ; // where $fp is the file resource
                            // returned by fopen ( )
```

با وجودی که می‌توان خط به خط فایل‌ها را با استفاده از تابع ( ) fopen خواند، به روشی نیاز است تا زمانی که به انتهای فایل می‌رسیم فرآیند خواندن را متوقف کنیم. تابع سیستمی ( ) feof برای این منظور معرفی شده است. تابع مذکور در صورتی که فرآیند خواندن به انتهای فایل می‌رسد مقدار true و در غیر این‌صورت مقدار false را باز می‌گرداند. باردیگر توجه داشته باشید که آرگومان این تابع مرجع فایلی است که تابع ( ) fopen آن را باز می‌گرداند:

```
Feof ($fp); // where $fp is the file resource returned by fopen ( )
```

برنامه‌موجود در لیست ۹- ۱۰ چگونگی بازخوانی خط به خط فایلی را به این روش نشان می‌دهد.

```
1: <html>
2: <head>
3: <title>Listing 10.9 Opening and reading a file line by line</title>
4: </head>
5: <body>
6: <?php
7: $filename = "test.txt";
8: $fp = fopen( $filename, "r" ) or die("Couldn't open $filename");
9: while ( ! feof( $fp ) ) {
10:     $line = fgets( $fp, 1024 );
11:     print "$line<br>";
12: }
13: ?>
14: </body>
15: </html>
```

لیست ۹- ۱۰ بازکردن فایلی جهت بازخوانی خط به خط

همان‌گونه که مشاهده می‌کنید، در خط ۸ از برنامه با استفاده از تابع سیستمی ( ) fopen فایلی به‌منظور خواندن باز شده است. استفاده از عملگر or در عبارت مربوطه این اطمینان را می‌دهد که در صورت عدم موفقیت تابع مذکور در بازکردن فایل مورد نظر اجرای برنامه پایان می‌یابد. این وضعیت معمولاً هنگامی اتفاق می‌افتد که فایل موجود نباشد و یا اینکه (در سیستمهای UNIX) مجوز لازم جهت خواندن فایل در دسترس کاربری که برنامه را اجرا می‌کند، نباشد. فرآیند اصلی خواندن در خط ۹ و در ساختار تکرار while انجام می‌گیرد. به ازای هرگذر از این ساختار تکرار عبارت شرطی اقدام به فراخوانی تابع ( ) feof می‌کند. چنان‌چه حاصل این فراخوانی مقدار true باشد، اجرای عملیات ساختار تکرار پایان می‌پذیرد. به‌عبارت دیگر، اجرای عملیات ساختار تکرار تاجایی ادامه پیدا می‌کند که فرآیند خواندن به انتهای فایل موردنظر برسد. به‌عبارت دیگر، اجرای عملیات ساختار تکرار تابع سیستمی ( ) fgets جهت استخراج یک خط (یا ۱۰۲۴ بایت) از فایل مورد نظر فراخوانی می‌شود. نتیجه این فراخوانی به متغیر $line منسوب شده و سپس در خط ۱۱ بر روی پنجره مرورگر اینترنت به نمایش درمی‌آید. ضمن اینکه انتهای هر خط خروجی در مرورگر با نشانه < BR > جهت افزایش میزان خوانایی همراه شده است.

خواندن حجم دلخواهی از اطلاعات درون یک فایل با استفاده از تابع ( ) fread

به‌جای خواندن خط به خط محتوای فایل، می‌توان آن‌را در اندازه‌های دلخواه مورد بازخوانی قرار داد. در زبان PHP تابع ( ) fread این وظیفه را انجام می‌دهد. تابع مذکور مرجع یک فایل و همچنین تعداد بایت‌های مورد نیاز جهت خواندن را به‌عنوان آرگومان دریافت می‌کند. این تابع تعداد بایت‌های بازخوانی شده از فایل را باز می‌گرداند. این تعداد در حالت عادی با تعداد بایت‌های درخواستی جهت خواندن (آرگومان دوم تابع) برابر است، مگر آنکه فرآیند خواندن به انتهای فایل برسد:

```
$chunk = fread ($fp, 16) ;
```

برنامه موجود در لیست ۱۰- ۱۰ مشابه برنامه لیست ۹- ۱۰ است با این تفاوت که به‌جای بازخوانی خط به خط فایل آن‌را در قطعات ۱۶ بایتی می‌خواند.

```
 1: <html>
 2: <head>
 3: <title>Listing 10.10 Reading a file with fread()</title>
 4: </head>
 5: <body>
 6: <?php
 7: $filename = "test.txt";
 8: $fp = fopen( $filename, "r" ) or die("Couldn't open $filename");
 9: while ( ! feof( $fp ) ) {
10:     $chunk = fread( $fp, 16 );
11:     print "$chunk<br>";
12: }
13: ?>
14: </body>
15: </html>
```

لیست ۱۰- ۱۰ خواندن یک فایل با استفاده از تابع ( ) fread

هرچند که بازخوانی فایل در قطعاتی با اندازه دلخواه از مزایای تابع ( ) fread است، این تابع هیچ‌گونه امکانی در رابطه با تعیین موقعیتی که عملیات خواندن باید از آنجا آغاز شود، در اختیار قرار نمی‌دهد. در عوض PHP تابع سیستمی دیگری را با عنوان ( ) fseek جهت تعیین نقطه آغاز فرآیند خواندن فایل در اختیار می‌گذارد. به‌کمک این تابع می‌توان نقطه شروع خواندن فایل را به‌دلخواه تعیین کرد. آرگومان اول این تابع مرجع فایل مورد نظر است که از طریق فراخوانی تابع ( ) fopen به‌دست می‌آید. آرگومان دوم تابع مذکور عدد صحیحی است که فاصله نقطه شروع خواندن را از ابتدای فایل برحسب بایت مشخص می‌کند:

```
Fseek ($fp, 64) ;
```

برنامه موجود در لیست ۱۱- ۱۰ با استفاده از توابع سیستمی ( ) fseek و ( ) fread نیمه دوم فایل دلخواهی را بر روی پنجره مرورگر به تصویر می‌کشد.

```
 1: <html>
 2: <head>
 3: <title>Listing 10.11 Moving around a file with fseek()</title>
 4: </head>
 5: <body>
 6: <?php
 7: $filename = "test.txt";
 8: $fp = fopen( $filename, "r" ) or die("Couldn't open $filename");
 9: $fsize = filesize($filename);
10: $halfway = (int)( $fsize / 2 );
11: print "Halfway point: $halfway <BR>\n";
12: fseek( $fp, $halfway );
13: $chunk = fread( $fp, ($fsize - $halfway) );
14: print $chunk;
15: ?>
16: </body>
17: </html>
```

لیست ۱۱- ۱۰ بهره‌گیری از تابع ( ) fseek

همان‌گونه که مشاهده می‌کنید، در خط ۱۰ از برنامه ابتدا موقعیت وسط فایل با تقسیم مقدار بازگشتی از تابع ( ) filesize محاسبه شده است. در خط ۱۲ از این مقدار به‌دست آمده به عنوان آرگومان دوم تابع ( ) fseek جهت تنظیم موقعیت شروع فرآیند خواندن استفاده شده است. در نهایت در خط ۱۳ با استفاده از تابع ( ) fread نیمه دوم فایل مورد بازخوانی قرارگرفته و نتیجه بر روی صفحه مرورگر اینترنت به نمایش درآمده است.

خواندن کاراکتر به کاراکتر اطلاعات درون فایل‌ها با استفاده از تابع ( ) fgetc

عملکرد تابع ( ) fgetc مشابه تابع ( ) fgets است با این تفاوت که با هربار فراخوانی به‌جای بازخوانی یک خط تنها یک کاراکتر از فایل را مورد بازخوانی قرار می‌دهد. از آنجا که اندازه کاراکتر همیشه برابر با یک بایت است تابع مذکور به آرگومان دومی که اندازه مذکور را تعیین کند، نیازی ندارد. آرگومان اول این تابع مرجع فایل حاصل از فراخوانی تابع ( ) fopen است:

```
$char = fgetc ($fp) ;
```

برنامه لیست ۱۲- ۱۰ شامل یک ساختار تکرار است که فایل test. txt را کاراکتر به کاراکتر در هربار عبور از حلقه خوانده و آنها را در پنجره مرورگر اینترنت نمایش می‌دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 10.12</title>
 4: </head>
 5: <body>
 6: <?php
 7: $filename = "test.txt";
 8: $fp = fopen( $filename, "r" ) or die("Couldn't open $filename");
 9: while ( ! feof( $fp ) ) {
10:    $char = fgetc( $fp );
11:    print "$char<BR>";
12: }
13: ?>
14: </body>
15: </html>
```

لیست ۱۲- ۱۰ استفاده از تابع ( ) fseek جهت بازخوانی فایل

نوشتن یا اضافه کردن محتوا به یک فایل

فرآیند نوشتن در فایل مشابه فرآیند اضافه کردن محتوا (اطلاعات) به یک فایل است. تنها تفاوت بین این‌دو در نحوه فراخوانی تابع ( ) fopen است. جهت نوشتن در یک فایل هنگام بازکردن آن با استفاده از تابع ( ) fopen آرگومان دوم را دنباله کاراکتری ' w ' تشکیل می‌دهد:

```
$fp = fopen ("test. txt", " w " ) ;
```

در این حالت هر تلاشی جهت نوشتن در فایل از نقطه شروع آن فایل آغاز خواهد شد. اگر فایل در این حالت موجود نباشد، ابتدا فایل مورد نظر ایجاد می‌شود و در صورتی که فایل موجود باشد هرگونه اطلاعاتی که از پیش در آن ذخیره شده باشد با نوشتن داده‌های جدید در آن از بین می‌رود.

آرگومان دوم تابع ( ) fopen را هنگام اضافه کردن محتوا به یک فایل دنباله کاراکتری ' a ' تشکیل می‌دهد:

```
$fp = fopen ("test. txt", " a " ) ;
```

در این حالت هرگونه تلاشی جهت نوشتن اطلاعات جدید در فایل موجب اضافه شدن آن به محتوای موجود در فایل خواهد شد.

نوشتن در فایل با استفاده از تابع ( ) fwrite و ( ) fputs

فرآیند نوشتن در فایل با استفاده از توابع ( ) fwrite و ( ) fputs امکان‌پذیر است. هردو تابع عملکرد مشابهی داشته و آرگومان‌های یکسانی را دریافت می‌کنند. آرگومان اول این توابع مرجع یک فایل و آرگومان دوم آنها یک دنباله کاراکتری است. هر دو تابع دنباله کاراکتری آرگومان دوم خود را در تابع مورد نظر می‌نویسند:

```
Fwrite ($fp, "hello world") ;
Fputs ($fp, "hello world") ;
```

نوشتن دریک فایل به همین راحتی است. برنامه موجود در لیست ۱۳- ۱۰ ابتدا با استفاده از تابع ( ) Fwrite فایلی را مورد بازنویسی قرار داده و سپس با بهره‌گیری از تابع ( ) fputs محتوایی را به آن اضافه می‌کند.

```
 1: <html>
 2: <head>
 3: <title>Listing 10.13 Writing and appending to a file</title>
 4: </head>
 5: <body>
 6: <?php
 7: $filename = "test.txt";
 8: print "Writing to $filename<br>";
 9: $fp = fopen( $filename, "w" ) or die("Couldn't open $filename");
10: fwrite( $fp, "Hello world\n" );
11: fclose( $fp );
12: print "Appending to $filename<br>";
13: $fp = fopen( $filename, "a" ) or die("Couldn't open $filename");
14: fputs( $fp, "And another thing\n" );
15: fclose( $fp );
16: ?>
17: </body>
18: </html>
```

لیست ۱۳- ۱۰ نوشتن و اضافه کردن محتوا به یک فایل

قفل‌کردن فایل‌ها با استفاده از تابع ( ) flock

تکنیکهایی که تا بدین‌جا در رابطه با نوشتن و خواندن فایل‌ها فراگرفتید درصورتی که تنها یک کاربر اقدام به اجرای برنامه‌های مربوطه کند، به‌خوبی پاسخ‌گوی نیازهایتان خواهند بود. با این حال در دنیای واقعی اوضاع اندکی متفاوت است، بدین معنی که معمولاً چندین کاربر به‌طور هم زمان اقدام به دستیابی و اجرای این برنامه‌ها خواهند کرد. وضعیتی را در نظر بگیرید که طی آن چندین کاربر برنامه‌ای را به اجرا درآورده‌اند که شامل قطعه کدی برای نوشتن در یک فایل مشخص باشد. در این صورت به‌واسطه نوشتن هم‌زمان در یک فایل داده‌های موجود به‌راحتی تحریف خواهد شد.

برای جلوگیری از این وضعیت PHP4 تابعی را ارائه کرده که نام آن ( ) Flock است. این تابع می‌تواند دسترسی کاربران را به یک فایل مشخص در صورتی که کاربر دیگری مشغول کار با آن باشد به‌طور مؤثری محدود نماید (منظور از دسترسی در اینجا خواندن یا نوشتن در فایل است). تابع ( )Flock دارای دو آرگومان است. آرگومان اول این تابع یک مرجع فایل معتبر است. آرگومان دوم عدد صحیحی است که نوع قفل‌کردن فایل را مشخص می‌کند. PHP4 چندین مقدار ثابت سیستمی را جهت تعیین نوع قفل فایل از پیش‌تعریف کرده است. بدین ترتیب می‌توانید به‌جای اعداد صحیح مذکور از این ثابتهای از پیش‌تعریف شده، استفاده کنید. جدول ۱- ۱۰ سه نوع قفل قابل استفاده را نشان می‌دهد.

جدول ۱- ۱۰ ثابت‌های سیستمی موجود جهت تعیین نوع قفل در تابع ( ) Flock

نام ثابت سیستمی	مقدار عددی معادل	نوع قفل	توضیح
LOCK _ SH	1	اشتراکی	این قفل اجازه خواندن را در اختیار سایر فرآیندها قرار می‌دهد اما از نوشتن جلوگیری می‌کند (از این قفل معمولاً هنگام خواندن یک فایل استفاده می‌شود)
LOCK_ EX	2	انحصاری	این قفل هم اجازه خواندن و هم اجازه نوشتن در فایل‌ها را از سایر فرآیندها سلب می‌کند (از این قفل معمولاً هنگام نوشتن در یک فایل استفاده می‌شود)
LOCK _ UN	3	باز	موجب باز شدن دو نوع قفل بالا می‌شود

به هنگام اجرای هم زمان برنامه توسط چندین کاربر مختلف فراخوانی تابع ( ) Flock درست پس از فراخوانی تابع ( ) fopen (جهت قفل کردن فایل) و فراخوانی مجدد آن درست قبل از فراخوانی تابع ( ) Fclose (جهت باز کردن قفل) اطمینان‌بخش است:

```
$fp = fopen (" test. txt ", " a "  ) ; or die (" couldn't open ") ;
Flock ($fp, LOCK _ EX) ; // exclusive lock
// write to the file
flock ($fp, LOCK _ UN) ; // release the lock
fclose ($fp) ;
```

قفل کردن فایل‌ها با استفاده از تابع سیستمی ( ) flock خاصیت منحصر به فردی دارد و آن این‌است که تنها سایر برنامه‌هایی که از تابع مذکور استفاده کرده‌اند، فرآیند قفل‌کردن را محترم می‌شمارند.

کار بر روی فهرستها

اکنون که چگونگی بررسی وجودی فایل‌ها و نحوه خواندن، نوشتن و قفل‌کردن آنها را فراگرفتید، وقت آن است که توجه خود را به فهرستها معطوف کنیم. زبان PHP توابع مختلفی را جهت کار بر روی فهرستها تدارک دیده است. در این قسمت قصد داریم تا چگونگی ایجاد، حذف و خواندن محتوای فهرستها را مورد توجه قرار دهیم.

ایجاد فهرستها با استفاده از تابع ( ) mkdir

به کمک تابع سیستمی ( ) mkdir می‌توان یک فهرست جدید ایجاد کرد. این تابع جهت اجرا به دو آرگومان نیاز دارد. اولین آرگومان یک دنباله کاراکتری است که مسیر فهرست مورد نظر را مشخص می‌کند. آرگومان دوم عددی در مبنای هشت است که مجوزهای اعطایی به فهرست حاصل را تعیین می‌کند. طبق قرارداد، اعداد مبنای هشت در زبان PHP با رقم صفر آغاز می‌شوند. آرگومان دوم تابع ( ) mkdir تنها در سیستم عامل UNIX تاثیرگذار است. عددی که بیانگر مجوزهای اعطایی به فهرست ایجاد شده است، باید دارای سه رقم صفر تا ۷ باشد (منهای صفر اول که بیانگر مبنای هشت است). هریک از این ارقام از چپ به راست به ترتیب بیانگر مجوزهای اعطایی به مالک فهرست (کسی که اقدام به ایجاد کردن آن نموده است)، گروهی که مالک فایل در آن واقع است (در سیستم عامل UNIX هرکاربری در یکی از گروههای موجود قرار می‌گیرد) و بقیه کاربران می‌باشد.چنانچه فهرست مورد نظر با موفقیت ایجاد شود، تابع ( ) mkdir مقدار true و در غیر این‌صورت مقدار false را بازمی‌گرداند. چنانچه عملیات تابع فوق در ایجاد فهرست مورد نظر با شکست مواجه شود، معمولاً می‌توان چنین نتیجه گرفت که مجوز نوشتن لازم در اختیار کاربری که برنامه را به اجرا درآورده است، نمی‌باشد. در صورتی که تنظیم مجوزهای فهرست در سیستم عامل UNIX خوشایند شما نباشد، شاید یکی از مثالهای زیر همواره به‌کارتان بیاید. غیر از مواردی که نوشتن در فهرست مورد نظر توسط همه کاربران

مشکل خاصی برای شما ایجاد نمی‌کند (این حالت معمولاً بعید است !)، همواره بهتر است تا آرگومان دوم تابع را به صورت 0755 تنظیم کنید. این بدان معنی است که مجوز خواندن را در اختیار همه کاربران قرار داده‌اید اما مجوز نوشتن را از آنان سلب کرده‌اید:

```
Mkdir ( "testdir ", 0777) ; // global read/ write/ execute
                              permissions
Mkdir ( "testdir ", 0755) ; // world and group : read / execute
                            // only, owner : read / write / execute
```

حذف یک فهرست با استفاده از تابع ( ) rmdir

در صورتی که مجوزهای لازم در دست بوده و فهرست مورد نظر خالی باشد، با استفاده از تابع سیستمی ( ) rmdir در برنامه می‌توان اقدام به حذف فهرست مذکور از سیستم نمود. این تابع تنها به آرگومانی نیاز دارد که مسیر فهرست مورد نظر را جهت حذف مشخص می‌کند:

```
rmdir (" testdir ") ;
```

بازکردن فهرستی جهت خواندن با استفاده از تابع ( ) opendir

پیش از هر اقدامی جهت خواندن محتوای یک فهرست لازم است تا مرجعی به آن فهرست در دسترس ما باشد. این عمل با استفاده از تابع سیستمی ( ) opendir قابل انجام است. آرگومان این تابع مسیر فهرستی را مشخص می‌کند که قصد بازکردن آن را داریم. چنانچه فهرست مورد نظر موجود بوده و مجوز لازم جهت خواندن آن نیز در دسترس باشد، تابع مذکور مرجعی را به آن بازمی‌گرداند. در غیر این‌صورت مقدار false چیزی است که این تابع باز می‌گرداند:

```
$dh = opendir ( "testdir ") ;
```

خواندن محتوای یک فهرست با استفاده از تابع ( ) readdir

همان‌گونه که تابع ( ) gets خطی از یک فایل را می‌خواند، تابع سیستمی ( ) readdir نیز نام فایل یا فهرست دیگری را از یک فهرست می‌خواند. آرگومان مورد نیاز این تابع مرجع یک فهرست است. آنچه که این تابع بازمی‌گرداند نام فایل یا فهرستی است که تابع مورد بحث آن را مورد بازخوانی قرار داده‌است. چنانچه فرآیند خواندن به انتهای فهرست برسد تابع مقدار false را باز می‌گرداند. توجه کنید که تابع ( ) readdir بدون ذکر نام مسیر، تنها نام فایل یا فهرست را بازمی‌گرداند. برنامه موجود در لیست ۱۴- ۱۰ محتوای یک فهرست را نمایش می‌دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 10.14 Listing the contents
 4: of a directory with readdir()</title>
 5: </head>
 6: <body>
 7: <?php
 8: $dirname = ".";
 9: $dh = opendir( $dirname ) or die("couldn't open directory");
10:
11: while ( ! ( ( $file = readdir( $dh ) ) === false ) ) {
12:     if ( is_dir( "$dirname/$file" ) )
13:         print " ";
14:     print "$file<br>";
15: }
16: closedir( $dh );
17: ?>
18: </body>
19: </html>
```

لیست ۱۴- ۱۰ نمایش محتوای یک فهرست با استفاده از تابع ( ) readdir

همان‌گونه که در این برنامه مشاهده می‌کنید، ابتدا در خط ۹ با استفاده از تابع ( ) opendir فهرستی را جهت خواندن بازکرده و سپس با استفاده از یک ساختار تکرار در خط ۱۱ عناصر موجود در آن را بازیابی کرده‌ایم. ملاحظه کنید که فراخوانی تابع ( ) readdir بخشی از عبارت شرطی ساختار تکرار while را تشکیل می‌دهد. حاصل این فراخوانی به متغیر $file نسبت داده می‌شود. در داخل بدنه ساختار تکرار while از متغیر $dirname به‌همراه متغیر $file جهت ایجاد یک مسیر کامل نمونه استفاده شده است. این مسیر در خط ۱۲ مورد ارزیابی قرارگرفته است. چنانچه مسیر فوق مربوط به یک فهرست باشد در خط ۱۳ دنباله کاراکتری " " بر روی صفحه مرورگر به نمایش در می‌آید. در نهایت نام فایل در خط ۱۴ به خروجی فرستاده می‌شود.

همان‌گونه که مشاهده کردید روش محتاطانه‌ای را در بخش عبارت شرطی ساختار تکرار while مورد استفاده قراردادیم. بیشتر برنامه نویسان PHP روش زیر را ترجیح می‌دهند:

```
While ($file = readdir ($dh) ) {
   Print "$file < BR > \ n" ;
}
```

در اینجا مقدار بازگشتی از تابع ( ) readdir تحت بررسی قرار می‌گیرد. از آنجا که تمامی دنباله‌های کاراکتری غیر از " 0 " به صورت true ارزیابی می‌شوند، هیچ مشکلی پیش نخواهد آمد. با این حال وضعیتی را تصور کنید که فهرستی شامل چهار فایل با اسامی "0 " ، "1 " ، " 2 " و " 3 " باشد. آن‌چه که در این‌صورت از اجرای قطعه کد بالا بر روی یک سیستم نمونه حاصل می‌شود، چنین خواهد بود:

این از آن جهت است که نام فایل " 0 " در درون ساختار تکرار به صورت false ارزیابی شود (نام فایل " 0 " که یک دنباله کاراکتری است حاصل فراخوانی تابع ( ) readdir می باشد) و این باعث پایان یافتن عملیات ساختار تکرار می‌گردد. راه‌حل موجود در لیست ۱۴- ۱۰، یعنی بهره‌گیری از عملگر = = = اطمینان می‌دهد که مقدار بازگشتی از تابع ( ) readdir دقیقاً معادل false نمی‌باشد. به‌عبارت دیگر، تنها مقدار عددی 0 معادل با false درنظر گرفته می‌شود.

جمع‌بندی

در درس این ساعت چگونگی استفاده از تابع ( ) include جهت شامل کردن فایل‌ها در اسناد و نحوه اجرای برنامه‌های PHP موجود در آنها مورد بررسی قرار گرفت. در این درس با چگونگی استفاده از برخی از مفیدترین توابع PHP جهت بررسی و کسب اطلاعات لازم درباره فایل‌ها آشنا شدید و توابعی را که برای خواندن خط به خط یا کاراکتر به کاراکتر یا بخش دلخواهی از یک فایل در PHP تدارک دیده شده‌اند، مورد بررسی و مطالعه قراردادید. همچنین چگونگی نوشتن در فایل‌ها و نیز اضافه کردن محتوای جدید به آنها را فراگرفتید و در نهایت با نحوه ایجاد، حذف و خواندن فهرستها آشنا شدید.

اکنون‌که توانایی انجام کار با فایل‌ها را دارید، می‌توانید داده‌های مورد نیازتان را ذخیره و بازیابی کنید. با این وجود، اگر فایل‌هایی که با آنها کار می‌کنیم از حجم بالایی برخوردار باشند، سرعت اجرای برنامه در بازیابی داده‌ها از آنها بسیار پایین خواهد بود. آنچه که برای غلبه بر این مشکل بدان نیاز داریم وجود نوعی بانک اطلاعاتی است. در درس ساعت بعد توابع مربوط به DBA را مورد بررسی قرار می‌دهیم. به کمک این توابع می‌توانیم سرعت خود را در دستیابی به داده‌ها افزایش دهیم. در زبان PHP توابع بسیار مفید و متنوعی در این رابطه پیش‌بینی شده است.

پرسش و پاسخ

پرسش: آیا استفاده از تابع سیستمی ( ) include موجب کاهش سرعت اجرای برنامه می‌شود؟

پاسخ: از آنجا که فایل‌های شامل شده توسط موتور PHP باز شده و مورد پردازش قرار می‌گیرند، طبیعی است که سرباری را به اجرای برنامه تحمیل می‌کنند. با این همه در بسیاری موارد مزایای استفاده مجدد از کدهای کتابخانه‌ای آنچنان قانع‌کننده است که سربار ناشی از آنها مسأله مهمی به‌چشم نمی‌آید.

پرسش: آیا همواره خاتمه دادن به اجرای برنامه در مواردی که نتوان به‌هر علتی فایل مورد نظر را جهت خواندن یا نوشتن باز کرد، ضروری است؟

پاسخ: همیشه باید این امکان را درنظر داشته باشید. اگر اجرای صحیح برنامه کاملاً به فایلی بستگی داشته باشد که باید به‌منظور خواندن یا نوشتن باز شود، بهره‌گیری از تابع سیستمی ( ) die جهت خاتمه دادن به اجرای برنامه و ضمن آن نمایش یک پیغام مفید بر روی صفحه می‌تواند کار عاقلانه‌ای محسوب شود. در مواقعی که حساسیت به این شدت نمی‌باشد، باز هم لازم است تا امکان پایان دادن به برنامه را مثلاً ضمن ثبت وقایع و وضعیت موجود در یک فایل ثبت وضعیت درنظر داشته باشید. در درس ساعت بیست و دوم با عنوان " اشکال زدایی" مطالب بیشتری درباره چگونگی ثبت وقایع می‌آموزید.

تمرینها

هدف از این بخش دوره مطالب فراگیری شده در قالب آزمون است. پاسخ بخش آزمون بلافاصله بعد از آن آمده است. بخش فعالیتها به منظور افزایش مهارت برنامه نویسی خواننده طراحی شده و شامل تمرینهایی است که فاقد پاسخ می‌باشند.

آزمون

۱- ازچه‌توابعی می‌توان جهت اضافه کردن کدهای کتابخانه‌ای به برنامه‌های موجود استفاده کرد؟
۲- از کدام تابع می‌توان جهت بررسی وجود یک تابع بخصوص در سیستم فایل اطمینان حاصل کرد؟
۳- چگونه می‌توان از اندازه یک فایل اطلاع حاصل نمود؟
۴- از کدام تابع سیستمی می‌توان جهت بازکردن یک فایل به منظور خواندن اطلاعات از آن یا درج محتوای جدید در آن استفاده کرد؟
۵- از کدام تابع سیستمی می‌توان جهت خواندن خط به خط اطلاعات یک فایل استفاده نمود؟
۶- چگونه می‌توان اطمینان یافت که فرآیند خواندن به انتهای فایل رسیده است؟
۷- از کدام تابع می‌توان جهت نوشتن خطی از داده‌ها در یک فایل استفاده کرد؟
۸- چگونه می‌توان فهرستی را جهت خواندن باز کرد؟
۹- از کدام تابع می‌توان جهت خواندن نام یک عنصر (فایل یا فهرست) از فهرست بعد از بازکردن آن فهرست جهت خواندن استفاده کرد؟

پاسخ آزمون

۱- با استفاده از توابع include ( ) یا require ( ) می‌توان فایل‌های PHP را در درون سند جاری شامل کرد. برای این منظور از توابع include _ once ( ) و requir _ once ( ) نیز می توان استفاده کرد.

۲- تابع file _ exists ( ) را می‌توان جهت بررسی وجود یک فایل در سیستم مورد استفاده قرار داد.

۳- تابع filesize ( ) اندازه فایل را برحسب تعداد بایت‌های موجود در آن باز می‌گرداند.

۴- با استفاده از تابع Fopen ( ) می‌توان فایلی را باز کرد. آرگومان اول این تابع مسیر فایل مورد نظر و آرگومان دوم آن حالت باز شدن فایل را مشخص می‌کند. تابع مذکور مرجعی را به فایل باز شده، باز می‌گرداند.

۵- تابع fgest ( ) قادر است تا فرآیند خواندن فایل را تا احراز اندازه بافر تعیین شده در آرگومان، یا تا رسیدن به انتهای فایل یا سند ادامه دهد.

۶- تابع feof ( ) در صورتی که فرآیند خواندن فایلی که مرجع آن به‌عنوان آرگومان به این تابع ارسال شده است، به انتهای فایل برسد مقدار true را باز می‌گرداند.

۷- با استفاده از تابع fputs ( )، می‌توان داده‌ها را در فایل نوشت.

۸- با استفاده از تابع opendir ( )، می‌توان فهرستی را جهت خواندن باز کرد.

۹- تابع سیستمی readdir ( ) نام عنصری از یک فهرست باز شده توسط تابع opendir ( ) را باز می‌گرداند.

فعالیتها

۱- فرمی‌ایجادکنیدکه‌نام‌ونام خانوادگی کاربر را دریافت نماید. سپس برنامه‌ای بنویسید که این نام و نام خانوادگی را در یک فایل ذخیره کند.

۲- برنامه‌ای ایجاد کنید که قادر به خواندن اطلاعات موجود از فایل ایجاد شده در تمرین قبل باشد. این برنامه علاوه بر نمایش محتوای فایل بر روی پنجره مرورگر باید تعداد خطوط خوانده شده از فایل و همچنین اندازه آن‌را نیز نمایش دهد (استفاده از نشانه < BR > در انتهای هر خط از خروجی را فراموش نکنید).

ساعت یازدهم

بهره‌گیری از توابع DBA

چنانچه به یک بانک اطلاعاتی از نوع SQL مانند MySQL یا Oracle دسترسی ندارید، معمولاً دستیابی شما به حداقل یک سیستم بانک اطلاعاتی از نوع DBM حتمی است. واژه DBM کوتاه شده DatabaseManager و به معنی مدیر بانک اطلاعاتی است. سیستمهای بانکی شبه DBM امکاناتی نظیر ذخیره و دستکاری داده‌هایی را که به شکل زوج نام ـ مقدار بر روی سیستم شما موجودند، در اختیارتان قرار می‌دهند.

از طرف دیگر، واژه DBA کوتاه شده Database Abstraction Layer بوده و به معنی لایه انتزاعی (مفهومی) بانک اطلاعاتی می‌باشد. DBA در حقیقت متشکل از توابعی است که امکانات و تسهیلات متداولی را جهت بهره‌گیری از سیستمهای بانکی مبتنی بر فایل در اختیار برنامه‌نویس قرار می‌دهد.

با وجودی که توابع DBA توانمندی‌ها و امکانات بانکهای اطلاعاتی SQL را در اختیار نمی‌گذارند، اما قابلیت در استفاده و راحتی به‌کارگیری موجب محبوبیت آنها می‌باشد. این حقیقت که توابع DBA تسهیلات متداول سیستم‌های بانکی را در اختیار قرار می‌دهد، و به عنوان مرجع مشترکی در بالای سیستم‌های بانکی مورد استفاده قرار می‌گیرد، بدین معنی است که کد برنامه‌ای که از این توابع استفاده می‌کند از قابلیت حمل بالایی برخوردار است حتی اگر خود فایل‌های بانک اطلاعاتی از چنین قابلیتی برخوردار نباشند.

در درس این ساعت مطالب زیر را بررسی خواهیم کرد:

- **چگونگی بازکردن یک بانک اطلاعاتی**
- **چگونه اضافه کردن داده‌ها به بانک اطلاعاتی**
- **چگونگی بازیابی داده‌ها از بانک اطلاعاتی**
- **چگونگی تغییر و حذف اقلام موجود در بانک اطلاعاتی**
- **چگونگی ذخیره داده‌هایی با ساختارهای پیچیده‌تر در بانکهای اطلاعاتی نوع DBM**

در ادامه به بررسی موارد فوق می‌پردازیم.

کالبد شکافی

جهت بهره‌گیری از توابع DBA ابتدا لازم است تا یکی از سیستمهای بانکی را که این توابع را مورد پشتیبانی قرار می‌دهند، بر روی سیستم خود نصب نمایید. چنانچه سیستم‌عامل مورد استفاده‌تان از نوع Linux باشد به احتمال قوی به یکی از این‌گونه بانکهای اطلاعاتی با نام GDBM (کوتاه شده GNU Database Manager) دسترسی دارید. برای هر سیستمی یک گزینه بخصوص در PHP وجود دارد که هنگام کامپایل کردن یا نصب PHP بر روی کامپیوتر باید آن‌را مورد استفاده قرار دهید. در جدول ۱- ۱۱ نام چند نوع بانک اطلاعاتی که از توابع DBA پشتیبانی به‌عمل می‌آورند به‌همراه گزینه‌های مربوطه جهت کامپایل یا نصب PHP آمده است.

جدول ۱- ۱۱ مشخصات چند سیستم بانکی که از توابع DBA پشتیبانی به‌عمل می‌آورند.

نام سیستم بانکی	گزینه کامپایل مربوطه	توضیح
cdbm	- - with - cdbm	سیستم بانکی از نوع فقط خواندنی
db2	- - with - db2	http: // www. Sleepycat. com /
db3	- - with - db3	http: // www. Sleepycat. com /
Dbm	- - with - dbm	DBM اصلی که اکنون منسوخ شده است
gdbm	- - with - gdbm	GNU Database Manager
ndbm	- - with - ndbm	منسوخ شده است

اگر سیستم عامل و PHP موجود بر روی آن یکی از این سیستمهای بانکی را پشتیبانی کند. آن‌گاه امکان بهره‌گیری از توابع DBA بدون هیچ مشکلی در دسترس خواهد بود. توجه کنید که پشتیبانی از بانک اطلاعاتی نوع cdbm (که جهت دستیابی سریع به بانکهای اطلاعاتی ایستا طراحی شده) در PHP به‌صورت فقط خواندنی می‌باشد.

بازکردن یک بانک اطلاعاتی

با استفاده از تابع سیستمی ( ) dba _ open می‌توان اقدام به بازکردن یک بانک اطلاعاتی از نوع DBM نمود. استفاده از این تابع مستلزم ارسال سه آرگومان به آن است. اولین آرگومان مسیر بانک اطلاعاتی را مشخص می‌کند. دومین آرگومان یک دنباله کاراکتری است که نحوه بازکردن بانک اطلاعاتی را تعیین می‌کند. و بالاخره سومین آرگومان نیز که یک دنباله کاراکتری است، نمایانگر DBM

مورد استفاده است (یکی از انواع موجود در جدول ۱- ۱۱). حاصل فراخوانی تابع ( )dba _ open مرجعی از نوع DBM است که می‌توان از آن به‌عنوان آرگومان توابع DBA جهت دستیابی و کار با بانک اطلاعاتی مورد نظر استفاده نمود.

از آنجا که تابع مورد بحث مستلزم خواندن و نوشتن در فایل‌ها است، طبیعی است که برنامه PHP مربوطه باید مجوزهای لازم جهت نوشتن در فهرستی که شامل بانک اطلاعاتی مورد نظر است را داشته باشد.

نحوه باز کردن بانک اطلاعاتی توسط دنباله کاراکتری که به عنوان دومین آرگومان تابع ( ) dba_open به آن ارسال می‌شود، مشخص می‌گردد. جدول ۲- ۱۱ گزینه‌های موجود در این زمینه را نشان می‌دهد.

جدول ۲- ۱۱ گزینه‌های قابل استفاده به‌عنوان دومین آرگومان تابع ( ) dba_open جهت تعیین نحوه بازکردن بانک اطلاعاتی

گزینه قابل استفاده	توضیح
r	بانک اطلاعاتی را صرفاً جهت خواندن اطلاعات باز می‌کند
w	بانک اطلاعاتی را جهت خواندن و نوشتن اطلاعات باز می‌کند
c	یک بانک اطلاعاتی جدید ایجاد می‌کند (در صورت وجود بانک مورد نظر آن را جهت خواندن و نوشتن اطلاعات باز می‌کند)
n	بانک اطلاعاتی جدیدی ایجاد می‌کند (در صورت وجود بانک مورد نظر ابتدا نسخه قدیمی را حذف می‌کند)

قطعه کد زیر اقدام به بازکردن بانکی با نام products می‌کند و در صورت عدم وجود، آن ابتدا آن‌را ایجاد می‌کند:

```
$dbh = dba _ open (" . / data / products ", "c", "gdbm") or
   die (" couldn't open Database") ;
```

همان‌گونه که مشاهده می‌کنید، درصورتی که تلاش برای بازکردن بانک اطلاعاتی با شکست مواجه شود، تابع سیستمی ( ) die موجب پایان دادن به اجرای برنامه می‌گردد.

هنگامی که کار خود را با بانک اطلاعاتی مورد نظر انجام دادید، با استفاده از تابع( )dba_close آن را ببندید. این از آن‌جهت است که PHP همواره اقدام به قفل کردن بانک اطلاعاتی که مشغول کار با آن هستید، می‌کند بدین ترتیب سایر فرآیندها حین کارکردن شما با بانک اطلاعاتی نمی‌توانند

داده‌ها را از آن خوانده یا در آن بنویسند. اگر پس از پایان کار اقدام به بستن بانک اطلاعاتی نکنید، PHP بانک اطلاعاتی را کماکان در حال قفل‌شده نگه داشته و بنابراین سایر فرآیندها امکان کار بر روی بانک اطلاعاتی را از دست خواهند داد. تنها آرگومان تابع ( ) dba _ close یک مرجع DBM معتبر است:

```
dba _ close ($dbh) ;
```

اضافه کردن داده‌ها به یک بانک اطلاعاتی

با بهره‌گیری از تابع ( ) dba _ insert می‌توان داده‌ها را در قالب زوجهایی از نام ـ مقدار به بانک اطلاعاتی اضافه کرد. این تابع سه آرگومان دریافت می‌کند. آرگومان اول نام یک کلید و آرگومان دوم مقدار مربوط به آن است که باید در بانک اطلاعاتی در قالب زوجی از نام و مقدار ذخیره شوند. آرگومان سوم مرجع DBM معتبری است که حاصل فراخوانی تابع ( ) dba _ open می‌باشد. اگر فرآیند نوشتن اطلاعات در بانک موفقیت آمیز باشد، تابع فوق مقدار true و در غیر این‌صورت مقدار false را باز می‌گرداند (تلاش برای نوشتن در یک بانک اطلاعاتی که صرفاً جهت خواندن باز شده یا رونویسی اطلاعات موجود با همان نام از جمله موارد ناکامی تابع ( ) dba _ insert می‌باشند). اگر اطلاعات وارد شده در بانک اطلاعاتی از پیش در بانک موجود باشند، تابع ( ) dba _ insert اقدام به رونویسی آنها نخواهد کرد، به‌عبارت دیگر اطلاعات پیشین حفظ خواهند شد.

برنامه موجود در لیست ۱- ۱۱ یک بانک اطلاعاتی با نام products را باز کرده و داده‌هایی را به آن اضافه می‌کند.

```
 1: <html>
 2: <head>
 3: <title>Listing 11.1 Adding items to a database</title>
 4: </head>
 5: <body>
 6: Adding products now...
 7:
 8: <?php
 9: $dbh = dba_open( "./data/products", "c", "gdbm" )
10:             or die( "Couldn't open database" );
11:
12: dba_insert( "Sonic Screwdriver", "23.20", $dbh);
13: dba_insert( "Tricorder", "55.50", $dbh);
14: dba_insert( "ORAC AI", "2200.50", $dbh);
15: dba_insert( "HAL 2000", "4500.50", $dbh);
16:
17: dba_close( $dbh );
18: ?>
19: </body>
20: </html>
```

لیست ۱- ۱۱ افزودن داده‌ها به یک بانک اطلاعاتی

همان‌گونه که مشاهده می‌کنید در خطوط ۱۲ تا ۱۵ این برنامه جهت افزودن اطلاعات به بانک از تابع ( ) dba _ insert استفاده شده است. از آنجا که تمامی مقادیر پیش از ورود به بانک اطلاعاتی به

دنباله کاراکتری نظیر تبدیل می‌شوند، در مورد قیمت محصولات از علامت نقل قول استفاده شده است تا بدین ترتیب جامعیت قالب‌بندی داده‌ها حفظ شود. در صورت نیاز به این داده‌ها، هنگام واکنشی این دنباله‌های کاراکتری از بانک اطلاعاتی می‌توان آنها را مقادیر عددی اعشاری تصور کرد. در درس ساعت چهارم با عنوان " بلوک‌های سازنده برنامه " در مورد انواع داده‌های عددی مفصل بحث شده است. همچنین دقت کنید که کلیدهای بازیابی مقادیر از بانک اطلاعاتی می‌توانند شامل بیش از یک کلمه باشند (برای مثال مورد " Sonic Screwdirver " در خط ۱۲ برنامه).

حال اگر تلاشی برای وارد کردن مقداری در این بانک صورت بگیرد که کلید دستیابی آن مشابه یکی از کلیدهای موجود است، تابع ( ) dba _ insert با ناکامی مواجه شده و مقدار false را بدون این‌که تغییری در بانک اطلاعاتی داده باشد، باز می‌گرداند. در بعضی شرایط ممکن است این وضعیت مطلوب باشد اما گاهی اوقات نیز ترجیح می‌دهید تا علاوه بر ورود داده‌های جدید داده‌های پیشین را نیز به روز درآورید. این امر کاملاً به شرایط بستگی دارد.

به روز رسانی داده‌های موجود در یک بانک اطلاعاتی

با استفاده از تابع ( ) dba _ replace ، می‌توان داده‌های جدید را جایگزین داده‌های قدیمی مشابه در بانک اطلاعاتی نمود. (منظور از داده‌های مشابه مقادیری هستند که کلید دستیابی آنها یکسان است). این تابع سه آرگومان ورودی دارد. آرگومان اول نام کلید دستیابی، آرگومان دوم مقدار جدید موردنظر و آرگومان سوم نیز یک مرجع معتبر DBM است. در صورت موفقیت‌آمیز بودن عملیات تابع مقدار true و در غیر این‌صورت مقدار false را باز می‌گرداند. برنامه موجود در لیست ۲- ۱۱ همان برنامه لیست قبل است با این تفاوت که داده‌های جدید بدون توجه به اینکه نسخه قبلی آنها در بانک اطلاعاتی موجود است یا خیر به بانک اضافه می‌شوند.

```
 1: <html>
 2: <head>
 3: <title>Listing 11.2 Adding or changing items
 4:        belonging to database</title>
 5: </head>
 6: <body>
 7: Adding products now...
 8: <?php
 9: $dbh = dba_open( "./data/products", "c", "gdbm" )
10:             or die( "Couldn't open database" );
11: dba_replace( "Sonic Screwdriver", "25.20", $dbh );
12: dba_replace( "Tricorder", "56.50", $dbh );
13: dba_replace( "ORAC AI", "2209.50", $dbh );
14: dba_replace( "HAL 2000", "4535.50", $dbh );
```

لیست ۲- ۱۱ نوسازی مقادیر موجود در بانک اطلاعاتی

```
15: dba_close( $dbh );
16: ?>
17: </body>
18: </html>
```

دنباله لیست ۲- ۱۱

همان‌گونه که مشاهده می‌کنید، تنها تغییری که این برنامه نسبت به برنامه لیست ۱- ۱۱ دارد، تغییر تابع ( ) dba _ insert به تابع ( ) dba _ replace است.

خواندن داده‌ها از بانک اطلاعاتی

اکنون که با چگونگی اضافه کردن داده‌ها به بانک اطلاعاتی آشنا شدید، لازم است تا طریقه بازیابی آنها از بانک اطلاعاتی را نیز فرا بگیرید. تابع ( ) dba _ fetch امکان بازیابی عناصر ذخیره شده در یک بانک اطلاعاتی را در اختیار قرار می‌دهد. این تابع نام کلید دستیابی عنصر مورد نظر را در بانک و همچنین نام مرجع DBM معتبری را به‌عنوان آرگومان ورودی می‌پذیرد. تابع مورد بحث مقدار مربوط به کلید دستیابی فوق را از بانک اطلاعاتی بازیابی کرده و آن‌را در قالب یک دنباله کاراکتری به‌عنوان نتیجه عملیات باز می‌گرداند. برای مثال جهت دستیابی به بهای محصولی با نام " Tricorder "، می‌توان قطعه کد زیر را مورد استفاده قرار داد:

```
$price = dba _ fetch (" Tricorder ", $dbh) ;
```

در صورتی که محصولی با عنوان " Tricorder " در بانک اطلاعاتی موجود نباشد، تابع فوق در فرآیند بازیابی با شکست مواجه شده و مقدار false را جهت اعلام این وضعیت باز می‌گرداند.

با وجود این دقت کنید که ممکن است همیشه اسامی کلیدهای دستیابی را در اختیار نداشته باشیم. از این‌رو پرسش زیر کاملاً منطقی به نظر می‌رسد: در صورتی که لازم باشد تا اسامی تمامی محصولات و بهای مربوط به هرکدام را بر روی صفحه مرورگر نمایش دهیم، غیر از ذکر نام کلیدهای دستیابی تمام محصولات در برنامه، چه اقدام دیگری می‌توانیم صورت دهیم؟ PHP مکانیزمی دارد که با استفاده از آن می‌توان هر یک از عناصر بانک اطلاعاتی را در قالب یک حلقه تکرار پردازش نمود.

تابع ( ) dba _ firstkey جهت دستیابی به اولین کلید موجود در بانک اطلاعاتی طراحی شده است. این تابع نام یک مرجع DBM معتبر را به‌عنوان آرگومان پذیرفته و اولین کلید دستیابی موجود در بانک اطلاعاتی را باز می‌گرداند. به‌خاطر داشته باشید که منظور از اولین کلید دستیابی، کلید دستیابی اولین داده‌ای نیست که وارد بانک اطلاعاتی شده است چرا که بانکهای اطلاعاتی شبه DBM معمولاً از سیستم مرتب‌سازی خود جهت ساماندهی داده‌های موجود استفاده می‌کنند. با در دست داشتن اولین کلید دستیابی می‌توان به کمک تابع ( ) dba _ nextkey کلید بعدی را در اختیار گرفت. باردیگر، تابع ( ) dba _ nextkey نیز نام یک مرجع DBM معتبر را به‌عنوان آرگومان ورودی پذیرفته و

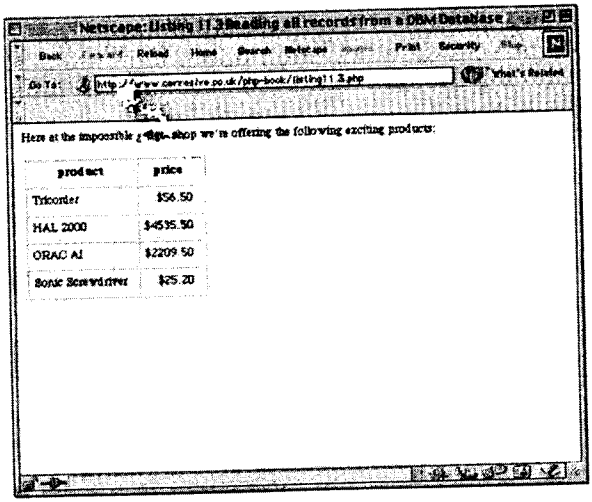
یک کلید دستیابی را به برنامه فراخواننده آن باز می‌گرداند. با ترکیب این توابع با تابع ( ) dba _ fetch می‌توان به راحتی کلیه مقادیر ذخیره شده در بانک اطلاعاتی را بازیابی کرد.

برنامه موجود در لیست ۳- ۱۱ کلیه محصولات موجود در بانک اطلاعاتی را در پنجره مرورگر نمایش می‌دهد. همان‌گونه که مشاهده می‌کنید در خط ۱۹ این برنامه از تابع ( ) dba _ firstkey استفاده شده است. در ساختار تکراری که در خط ۲۰ برنامه تشکیل می‌شود، در هربار گذر از حلقه یکی از عناصر موجود در بانک اطلاعاتی مورد پردازش قرار می‌گیرد. این فرآیند به تعداد این عناصر تکرار می‌شود. در خط ۲۱، تابع ( ) dba _ fetch اقدام به واکشی عنصری از بانک اطلاعاتی می‌کند. پس از نمایش مقدار این عنصر در صفحه مرورگر، با فراخوانی تابع ( ) dba _ nextkey کلید دستیابی در اختیار برنامه قرار می‌گیرد. این کلید دستیابی در متغیری با نام $key ذخیره می‌شود. عملیات تا جایی ادامه پیدا می‌کند که هیچ کلید دستیابی دیگری در بانک با استفاده از تابع ( ) dba _ nextkey در دسترس قرار نگیرد. در این صورت همان‌گونه که پیشتر نیز عنوان شد، تابع مذکور مقدار false را باز می‌گرداند. در این شرایط ساختار حلقه به عملیات خود پایان می‌دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 11.3 Reading all
 4:        records from a Database </title>
 5: </head>
 6: <body>
 7:    Here at the Impossible Gadget Shop
 8:    we're offering the following exciting
 9:    products:
10: <p>
11: <table border=1 cellpadding ="5">
12: <tr>
13: <td align="center"> <b>product</b></td>
14: <td align="center"> <b>price</b> </td>
15: </tr>
16: <?php
17: $dbh = dba_open( "./data/products", "c", "gdbm" )
18:             or die( "Couldn't open database" );
19: $key = dba_firstkey( $dbh );
20: while ( $key != false ) {
21:     $value = dba_fetch( $key, $dbh);
22:     print "<tr><td align = \"left\"> $key </td>";
23:     print "<td align = \"right\"> \$$value </td></tr>";
24:     $key = dba_nextkey( $dbh);
25: }
26: dba_close( $dbh );
27: ?>
28: </table>
29: </body>
30: </html>
```

لیست ۳- ۱۱ بازخوانی کلیه رکوردهای موجود در یک بانک اطلاعاتی

خروجی حاصل از اجرای این برنامه در شکل ۱- ۱۱ قابل بررسی است.



شکل ۱- ۱۱ خروجی برنامه بازخوانی رکوردهای یک بانک اطلاعاتی

تشخیص وجود یک داده مشخص در بانک اطلاعاتی

پیش از خواندن یا نوشتن داده‌ها در یک بانک اطلاعاتی اغلب بهتر است تا از وجود یا عدم وجود آن داده‌ها در بانک اطلاعاتی حاصل نماییم. تابع ( ) dba _ exists امکان انجام چنین بررسی را در اختیارمان قرار می‌دهد. این تابع نام کلید دستیابی مورد نظر و همچنین نام یک مرجع DBM معتبر را به‌عنوان آرگومان پذیرفته و در صورت وجود مقدار مربوطه در بانک اطلاعاتی مقدار true را باز می‌گرداند. در غیر این‌صورت حاصل عملیات این تابع مقدار false است. به نمونه‌ای از کاربرد این تابع توجه کنید:

```
If (dba _ exists (" Tricorder ", $dbh) )
   Print  dba _ fetch (" Tricorder ", $dbha) ;
```

حذف داده مشخصی از بانک اطلاعاتی

با استفاده از تابع ( ) dba _ delete می‌توان عنصر مشخصی از یک بانک اطلاعاتی را حذف کرد. تابع مذکور جهت اجرا به کلید دستیابی عنصر مورد نظر و همچنین نام یک مرجع DBM معتبر نیاز دارد. در صورتی که تابع فوق قادر به یافتن چنین داده‌ای در بانک اطلاعاتی باشد آن‌را حذف کرده و مقدار true را به نشانه موفقیت آمیز بودن عملیات باز می‌گرداند. در صورتی که داده مورد نظر در بانک اطلاعاتی موجود نباشد، تابع ( ) dba _ delete قادر به حذف آن نبوده و مقدار false را به‌عنوان نتیجه عملیات باز می‌گرداند. به نمونه‌ای از کاربرد این تابع توجه کنید:

```
dba _ delete (" Tricorder ", $dbh) ;
```

اضافه‌کردن‌داده‌هایی‌باساختارپیچیده‌تر به یک بانک‌اطلاعاتی

از آنجا که کلیه داده‌های موجود در بانکهای اطلاعاتی شبه DBM در قالب دنباله‌های کاراکتری بازیابی می‌شوند، محدود به ذخیره داده‌ها در یکی از صورتهای ممکن یعنی عدد صحیح، عدد اعشاری و دنباله کاراکتری هستیم. به‌عبارت دیگر ذخیره هر نوع داده دیگری در بانکهای اطلاعاتی شبه DBM موجب از دست رفتن آنها خواهد شد. برای روشن شدن مطلب اجازه دهید تا یک آرایه نمونه را در بانک اطلاعاتی ذخیره نماییم:

```
$array = array ( 1, 2, 3, 4) ;
$dbh = dba _ open (" . / data / test ", "c", "gdbm") or
   die (" couldn't open test ") ;
dba _ insert (" arraytest ", $array, $dbh) ;
print gettype (dba _ fetch (" arraytest ", $dbh) ;
// prints "string"
```

در قطعه کد بالا ابتدا آرایه‌ای ایجاد شده و در متغیری با نام $array ذخیره شده است. سپس بانک اطلاعاتی مورد نظر با نام test که بانکی از نوع gdbm است، باز شده و سعی شده تا عنصری با نام "arraytest" و با مقداری که در متغیر $array ذخیره شده (یک آرایه) به بانک اطلاعاتی اضافه شود. در گام بعدی سعی بر این است که مقدار عنصری از بانک اطلاعاتی که کلید دستیابی آن دنباله کاراکتری "arraytest" است، توسط تابع ( ) dba _ fetch بازیابی شود با این امید که این مقدار همان دنباله کاراکتری مورد انتظار باشد. در حقیقت چنانچه اقدام به نمایش مقدار به‌دست آمده بر روی پنجره مرورگر نماییم چیزی غیر از دنباله کاراکتری " Array " را مشاهده نمی‌کنیم. چنین به نظر می‌رسد که هیچ شانسی برای ذخیره آرایه‌ها و اشیا در بانکهای اطلاعاتی شبه DBM نداشته باشیم.

خوشبختانه PHP مکانیزمی را تدارک دیده است که با استفاده از آن می‌توانیم هر نوع داده‌ای را در قالب دنباله کاراکتری ذخیره کرده و سپس به‌هنگام نیاز، مجدداً آن‌را به صورت اصلی خود درآوریم. با تبدیل هر نوع داده‌ای به دنباله کاراکتری می‌توانیم آن‌را تا زمان نیاز مجدد در قالب دنباله کاراکتری در یک بانک اطلاعاتی یا یک فایل ذخیره کنیم. با استفاده از این روش می‌توانیم آرایه‌ها و حتی اشیا را در یک بانک اطلاعاتی ذخیره نماییم.

در مثال فوق برای تبدیل آرایه به یک دنباله کاراکتری می‌توانیم از تابع ( ) serialize استفاده کنیم. این تابع مقداری از هر نوع داده‌ای را دریافت کرده و یک دنباله کاراکتری باز می‌گرداند. به کاربرد نمونه زیر توجه کنید:

```
$array = array (1, 2, 3, 4) ;
print serialize ($array) ;
// print a: 4: { i: 0; i: 1; i: 1; i: 2; i: 2; i: 3; i: 3; i: 4; }
```

اکنون می‌توانیم این دنباله کاراکتری را در یک بانک اطلاعاتی ذخیره کنیم. جهت بازیابی اطلاعات در نوع اصلی خود از تابع دیگری با نام ( ) unserialize استفاده می‌کنیم. تابع نامبرده یک دنباله کاراکتری را که حاصل عملیات تابع ( ) serialize است به‌عنوان آرگومان دریافت کرده و اطلاعات را در نوع اصلی خود بازیابی می‌کند.

به کمک روشی که در بالا شرح دادیم، اکنون می‌توانیم داده‌های با ساختارهای پیچیده را نیز به‌گونه‌ای نسبتاً ساده در بانکهای اطلاعاتی شبه DBM ذخیره کرده و بازیابی نماییم. برنامه موجود در لیست با استفاده از همین روش ابتدا به‌ازای هر محصول موجود در بانک اطلاعاتی یک آرایه انجمنی را با بهره‌گیری از تابع ( ) serialize به یک دنباله کاراکتری تبدیل کرده و سپس نتیجه حاصل را در بانک اطلاعاتی ذخیره می‌کند.

```
1: <html>
2: <head>
3: <title>Listing 11.4 Adding complex data to a database</title>
4: </head>
5: <body>
6: Adding complex data to database
7: <?php
8: $products = array(
9:         "Sonic Screwdriver" => array( price=>"22.50",
10:                                      shipping=>"12.50",
11:                                      color=>"green" ),
12:
13:         "Tricorder"         => array( price=>"55.50",
14:                                      shipping=>"7.50",
15:                                      color=>"red" ),
16:
17:         "ORAC AI"           => array( price=>"2200.50",
18:                                      shipping=>"34.50",
19:                                      color=>"blue" ),
20:
21:         "HAL 2000"          => array( price=>"4500.50",
22:                                      shipping=>"18.50",
23:                                      color=>"pink" )
24:
25:         );
26: $dbh = dba_open( "./data/products2", "c", "gdbm" )
27:             or die( "Couldn't open database" );
28: while ( list ( $key, $value ) = each ( $products ) )
29:     dba_replace( $key,  serialize( $value ), $dbh );
30: dba_close( $dbh );
31: ?>
32: </body>
33: </html>
```

لیست ۴- ۱۱ اضافه کردن داده‌هایی با ساختار پیچیده به بانک اطلاعاتی

همان‌گونه که مشاهده می‌کنید ساخت یک آرایه چند بعدی در خط ۸ از این برنامه آغاز می‌شود. اسامی محصولات موجود کلیدهای دستیابی و چهار آرایه شامل اطلاعات هر یک از این محصولات، مقادیر مربوط به این کلیدهای دستیابی را تشکیل می‌دهند. در خط ۲۲ از برنامه بانک اطلاعاتی مورد نظر باز شده و در خط ۲۴، یک ساختار تکرار جهت پردازش عناصر آرایه تشکیل

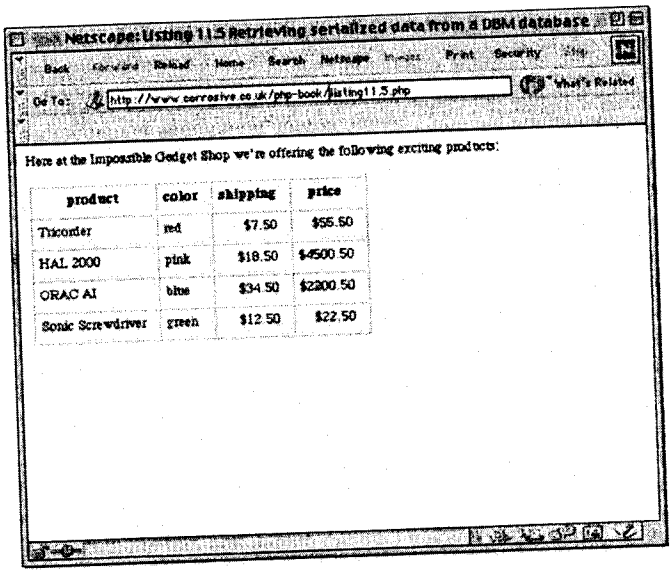
می‌گردد. به‌ازای هر عنصر از این آرایه، نام محصول و همچنین حاصل عملیات تابع ( ) serialize بر روی آن عنصر (که خود آرایه دیگری می‌باشد) در خط ۲۵ از برنامه به تابع ( ) dba _ replace ارسال می‌شود. پس از اتمام عملیات در خط ۲۶ تابع ( ) dba _ close جهت بستن بانک اطلاعاتی مورد فراخوانی قرار می‌گیرد. برنامه موجود در لیست ۵- ۱۱ سعی می‌کند تا داده های ذخیره شده در این بانک اطلاعاتی را بازیابی کند.

```
 1: <html>
 2: <head>
 3: <title>Listing 11.5 Retrieving serialized
 4:        data from a database</title>
 5: </head>
 6: <body>
 7:     Here at the Impossible Gadget Shop
 8:     we're offering the following exciting
 9:     products:
10: <p>
11: <table border=1 cellpadding ="5">
12: <tr>
13: <td align="center"> <b>product</b></td>
14: <td align="center"> <b>color</b> </td>
15: <td align="center"> <b>shipping</b> </td>
16: <td align="center"> <b>price</b> </td>
17: </tr>
18: <?php
19: $dbh = dba_open( "./data/products2", "c", "gdbm" )
20:             or die( "Couldn't open database" );
21: $key = dba_firstkey( $dbh );
22: while ( $key != false ) {
23:     $prodarray = unserialize( dba_fetch( $key, $dbh) );
24:     print "<tr><td align=\"left\"> $key </td>";
25:     print "<td align=\"left\">".$prodarray['color']." </td>\n";
26:     print "<td align=\"right\">\$".$prodarray['shipping']." </td>\n";
27:     print "<td align=\"right\">\$".$prodarray['price']." </td></tr>\n";
28:     $key = dba_nextkey( $dbh );
29: }
30: dba_close( $dbh );
31: ?>
32: </table>
33: </body>
34: </html>
```

لیست ۵- ۱۱ بازیابی داده‌هایی با ساختار پیچیده که در بانک اطلاعاتی ذخیره شده‌اند.

این برنامه شباهت زیادی به برنامه لیست ۳- ۱۱ که پیشتر مورد بررسی قرار گرفت، دارد. با این همه تفاوتی که وجود دارد نمایش تعداد بیشتری از فیلدها در برنامه اخیر است. همان‌گونه که مشاهده می‌کنید در خط ۱۹ ابتدا بانک اطلاعاتی باز شده و سپس در خط ۲۱ با استفاده از تابع dba _ firstkey ( ) و تابع dba _ nextkey ( ) در خط ۲۸ از برنامه، در هربار گذر از حلقه یکی از عناصر بانک اطلاعاتی مورد پردازش قرار گرفته‌اند. در خط ۲۳ پس از بازیابی داده‌ها توسط تابع dba_ fetch( ) با

استفاده از تابع ( ) unserialize آرایه محصولات مجدداً ساخته می‌شود. از اینجا به بعد با در دست داشتن یک آرایه انجمنی نمایش مقادیر آن بر روی صفحه کار ساده‌ای است. نتیجه اجرای این برنامه در شکل ۲- ۱۱ قابل توجه است.



شکل ۲- ۱۱ بازیابی داده‌هایی با ساختار پیچیده از بانک اطلاعاتی

بررسی یک مثال جامع

با اطلاعاتی که تا بدین‌جای درس این ساعت درباره بانکهای اطلاعاتی شبه DBM و جزئیات مربوطه فراگرفتید، می‌توانیم مثال جامعی را که دربرگیرنده تمامی نکات باشد مطرح کنیم. هدف ما در این مثال ایجاد یک صفحه مدیریتی است تا کاربر مربوطه بتواند بهای محصولات موجود در بانک اطلاعاتی products را که در برنامه لیست ۲- ۱۱ ایجاد کردیم، تغییر دهد. کاربر فوق همچنین باید بتواند اقلام جدیدی را به بانک اطلاعاتی اضافه کرده و اقلام قبلی را درصورت نیاز از بانک اطلاعاتی حذف کند. از آنجا که این برنامه از طریق یک سایت عمومی در اختیار همگان قرار نمی‌گیرد، لذا نیازی به حل مسائل امنیتی نیز نمی‌باشد.

ابتدا به فرمی نیاز داریم تا تمامی عناصر موجود در بانک اطلاعاتی را نمایش دهد. کاربری که از این فرم استفاده می‌کند با بهره‌گیری از یک فیلد متن می‌تواند بهای محصولات را به‌دلخواه تغییر داده و همچنین با استفاده از یک کادر علامت می‌تواند عناصری از بانک اطلاعاتی را جهت حذف مشخص نماید. علاوه بر این امکانات، قادر است تا با استفاده از دو فیلد متن دیگر عنصر جدیدی را به بانک اطلاعاتی اضافه کند. برنامه موجود در لیست ۶- ۱۱ کد مورد نیاز جهت تولید این فرم را نشان می‌دهد.

```
 1: <?php
 2: $dbh = dba_open( "./data/products", "c", "gdbm" )
 3:              or die( "Couldn't open database" );
 4: ?>
 5: <html>
 6: <head>
 7: <title>Listing 11.6 Building an html form based
 8:        on content from a database</title>
 9: </head>
10: <body>
11: <form action="POST">
12: <table border="1">
13: <tr>
14: <td>delete</td>
15: <td>product</td>
16: <td>price</td>
17: </tr>
18: <?php
19: $key = dba_firstkey( $dbh );
20: while ( $key != false ) {
21:     $price = dba_fetch( $key,  $dbh );
22:     print "<tr><td><input type='checkbox' name=\"delete[]\" ";
23:     print "value=\"$key\"></td>";
24:     print "<td>$key</td>";
25:     print "<td> <input type=\"text\" name=\"prices[$key]\" ";
26:     print "value=\"$price\"> </td></tr>";
27:     $key = dba_nextkey( $dbh );
28: }
29: dba_close( $dbh );
30: ?>
31: <tr>
32: <td> </td>
33: <td><input type="text" name="name_add"></td>
34: <td><input type="text" name="price_add"></td>
35: </tr>
36: <tr>
37: <td colspan=3 align="right">
38: <input type="submit" value="amend">
39: </td>
40: </tr>
41: </table>
42: </form>
43: </body>
44: </html>
```

لیست ۶- ۱۱ ایجاد یک فرم HTML بر مبنای داده‌های موجود در بانک اطلاعاتی

همان‌گونه که مشاهده می‌کنید فرآیند ساخت این فرم بلافاصله پس از بازکردن بانک اطلاعاتی موردنظر در خط ۲ از برنامه آغاز شده است. همان‌گونه که از تنظیم نشانه < form > در خط ۱۱ برنامه پیداست، فرم HTML حاصل صفحه جاری را مورد اشاره قرار می‌دهد.

پس از تعیین عناوین ستونهای جدول اصلی فرم در خطوط ۱۳ تا ۱۷، با استفاده از توابع dba_ firstkey ( ) و dba _ nextkey ( ) (به ترتیب در خطوط ۱۹ و ۲۷) ساختار تکراری جهت پردازش

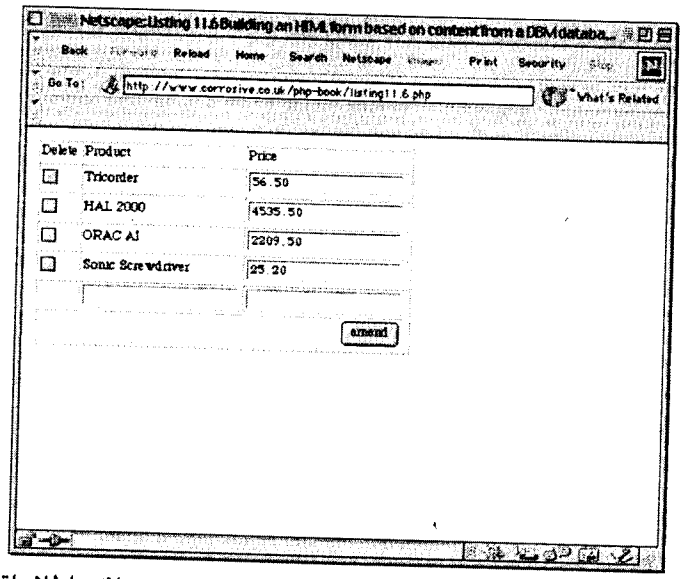
محتوای بانک اطلاعاتی تشکیل شده است. در هر بار گذر از این حلقه، تابع ( ) dba_ fetch با دریافت کلید دستیابی اقدام به بازیابی مقدار مربوط به آن از بانک اطلاعاتی می‌کند (خط ۲۱).

در اولین خانه از هریک از سطرهای جدول یک کادر علامت (checkbox) ایجاد شده است (خط ۲۲). توجه کنید که نام "[ ] delete" را برای تمامی این کادرهای علامت انتخاب کرده‌ایم. این حرکت موجب می‌شود تاPHP آرایه‌ای با نام $delete ایجاد کند. آرایه مزبور شامل تمام مقادیر ارسالی از عناصر فرم HTML است که نام آنها "[ ] delete" می‌باشد. به استفاده از نام کلیدهای دستیابی مقادیر موجود در بانک اطلاعاتی (که در متغیر $key نگهداری می‌شوند) به‌عنوان مقادیر این کادرهای علامت توجه کنید. هنگامی که فرم به وب سرور ارسال می‌شود آرایه‌ای با نام $array ایجاد می‌گردد. این آرایه شامل اسامی کلیدهای دستیابی مقادیری از بانک اطلاعاتی است که کاربر قصد حذف آنها را دارد.

عبارت موجود در خط ۲۴ برنامه نام کلید دستیابی را بر روی پنجره مرورگر نمایش داده و عبارت خط ۲۵ فیلد متن دیگری را ایجاد می‌کند. این فیلد بهای محصول را نمایش می‌دهد. کاربر در صورت نیاز می‌تواند مقدار این فیلد را تغییر دهد. نام‌گذاری این فیلد نیز مشابه تکنیک به‌کار رفته برای نام‌گذاری فیلد قبل انجام شده است. با این حال این بار نام کلیدهای دستیابی در جلوی نام price توسط جفت علامت [ ] محصور شده‌اند. به محض ارسال فرم PHP آرایه انجمنی دیگری با نام $price می‌سازد که اسامی کلیدهای دستیابی مقادیر کلیدی آن‌را تشکیل می‌دهند.

در خط ۲۹ با بستن بانک اطلاعاتی توسط تابع ( ) dba _ close وظیفه برنامه PHP نیز خاتمه می‌یابد. حال وقت آن است که دو فیلد دیگر به فرم HTML اضافه شود (خطوط ۳۳ و ۳۴). به کمک این فیلدها کاربر می‌تواند داده جدیدی را به‌همراه کلید دستیابی مربوطه (نام و بهای محصول) به بانک اطلاعاتی اضافه کند. اسامی این فیلدها name_ add و price_ add انتخاب شده‌اند.

می‌توانید خروجی حاصل از این برنامه را در شکل ۳- ۱۱ مشاهده نمایید.



شکل ۳- ۱۱ یک فرم HTML که بر مبنای داده‌های موجود در یک بانک اطلاعاتی ساخته شده است

اکنون که فرم را ایجاد کردیم لازم است برنامه اسکریپتی را جهت کنترل عملیات کاربر ایجاد کنیم. این فرآیند به راحتی قابل پیاده‌سازی است. در این مورد می‌توان سه احتمال ممکن را تشخیص داد: اول اینکه کاربر ممکن است داده‌هایی را از بانک اطلاعاتی حذف کند. دوم آنکه کاربر ممکن است بهای محصولات موجود در بانک اطلاعاتی را تغییر دهد. و نهایت اینکه وی ممکن است داده‌های جدیدی را به بانک اطلاعاتی اضافه نماید.

در ادامه روش کنترل این سه امکان را مورد بررسی قرار می‌دهیم.

اگر فرم مورد بحث ارسال شده باشد می‌توانیم اقلامی را که باید از بانک اطلاعاتی حذف شوند به‌راحتی مشخص کنیم چراکه آرایه $delete در این صورت در دسترس ما خواهد بود. برای این‌کار تنها کافی‌است تا ساختار تکراری را بر روی عناصر آرایه تشکیل داده و اقلام مربوط به آنها را با فراخوانی تابع ( ) dba _ delete از بانک اطلاعاتی حذف کنیم. قطعه کد زیر نحوه انجام این کار را نشان می‌دهد:

```
if ( ! empty ($delete) ) {
    While (list ($key, $val) = each ( $delete) ) {
        Unset ($prices [$val] ) ;
        dba _ delete ($val , $dbh) ;
    }
}
```

همان‌گونه که در این قطعه کد کوتاه مشاهده می‌کنید، ابتدا از وجود آرایه $delete و نیز خالی نبودن آن اطمینان حاصل شده است. اگر کاربر اولین باری است که صفحه شامل این فرم را مشاهده

می‌کند یا هیچ عنصری از بانک اطلاعاتی را جهت حذف انتخاب نکرده باشد این آرایه اصلاً وجود نخواهد داشت. اما درصورت وجود آن می‌توان ساختار حلقه موردنظر را تشکیل داد. در داخل حلقه تکرار به ازای هر دنباله کاراکتری موجود در آرایه $delete جهت حذف مقدار مربوطه از بانک اطلاعاتی تابع ( ) dba _ delete فراخوانی شده است. در این رابطه آرایه دیگری با نام $prices را نیز مورد استفاده قرار داده‌ایم. آرایه نامبرده یک آرایه انجمنی است که کلیدهای دستیابی و مقادیر مربوط به هریک از آنها را شامل می‌شود (البته برخی از این مقادیر ممکن است توسط کاربر دستخوش تغییر شده باشند). پیش از حذف هر عنصر از بانک اطلاعاتی کلید دستیابی و مقدار مربوط به آن نیز از این آرایه حذف می‌شود. بلوک کد بعدی باردیگر این عناصر را به بانک اطلاعاتی اضافه می‌کند.

اما جهت به‌روز رسانی بانک اطلاعاتی بر مبنای تغییرات اعمال شده توسط کاربر از طریق فرم HTML انتخابی پیش روی ماست. برای این‌کار می‌توانیم تنها مقادیری را که کاربر آنها را تغییر داده‌است، به‌روز برسانیم. از این روش بهتر است در مواقعی بهره بگیریم که چندین کاربر مختلف به‌طور هم‌زمان از بانک استفاده می‌کنند یا در مواقعی که سرعت رشد بانک اطلاعاتی زیاد باشد. اما از آنجا که این فرم مختص یک کاربر بخصوص (administrator) طراحی شده و از طرف دیگر تعداد محصولات نیز زیاد نمی‌باشد لذا تصمیم به‌روز رسانی تمامی عناصر بانک اطلاعاتی تصمیم دور از ذهنی نیست. به چگونگی انجام این کار توجه کنید:

```
if ( ! empty ($prices) ) {
   While (list ($key, $val) = each ($prices) )
      db _ replace ( $key, $val, $dbh) ;
}
```

در قطعه کد فوق ابتدا از وجود آرایه $prices اطمینان حاصل شده است. این آرایه باید شامل نسخه جدیدی از تمامی داده‌های موجود در بانک اطلاعاتی باشد. سپس در قالب یک ساختار تکرار تابع ( ) dba _ replace برای هریک از عناصر بانک اطلاعاتی فراخوانی شده است.

گام نهایی این است که بررسی لازم را جهت اطلاع از این موضوع که کاربر داده‌ای را جهت اضافه کردن آن به بانک اطلاعاتی وارد کرده، انجام دهیم. قطعه کد زیر را به‌دقت مطالعه کنید:

```
if ( ! empty ($name _ add) &&! empty ($price _ add) )
   dba _ replace (" $name _ add", "$price _ add", $dbh) ;
```

همان‌گونه که در این قطعه کد مشاهده می‌کنید، به جای بررسی اینکه آیا متغیرهای $price_add و $name _ add با مقداری تنظیم شده‌اند یا خیر این بررسی در مورد تهی بودن آنها صورت گرفته است. این تفاوتی ساده اما در عین حال مهم است. هنگامی که کاربر فرم مورد بحث را ارسال می‌کند این متغیرها همیشه با مقادیری تنظیم می‌شوند. با این وجود متغیرهای فوق ممکن است حاوی دنباله‌های کاراکتری تهی باشند. از آنجا که قصد اضافه کردن دنباله‌های کاراکتری تهی را به

بانک اطلاعاتی نداریم، ترتیبی می‌دهیم تا تنها درصورت تهی نبودن این متغیرها مقادیر آنها در بانک اطلاعاتی درج شود. قطعه کد زیر چنین عملی را انجام می‌دهد:

```
if ( ! empty ($name _ add) && ! empty ($price _ add) )
   dba _ insert (" $name _ add", " $price _ add", $dbh) ;
```

در قطعه کد بالا جهت جلوگیری از رونویسی داده‌های موجود توسط مقادیر جدیدی که کاربر وارد می‌کند به‌جای استفاده از تابع ( ) dba _ replace ترجیح داده‌ایم تا از تابع ( ) dba _ insert استفاده کنیم.

برنامه موجود در لیست ۷- ۱۱ کلیه عملیات را به‌صورت یک برنامه کامل نشان می‌دهد. خطوط ۵ تا ۹ این برنامه مربوط به حذف داده‌ها از بانک اطلاعاتی است. خطوط ۱۲ تا ۱۵ فرآیند به‌روز رسانی داده‌ها را کنترل می‌کند. و بالاخره خطوط ۱۷ و ۱۸ برنامه نظارت اضافه کردن داده‌های جدید به بانک اطلاعاتی را به‌عهده دارد.

```
 1: <?php
 2: $dbh = dba_open( "./data/products", "c", "gdbm" )
 3:              or die( "Couldn't open database" );
 4:
 5: if ( ! empty( $delete ) ) {
 6:     while ( list ( $key, $val ) = each ( $delete ) )  {
 7:         unset( $prices[$val]);
 8:         dba_delete( $val, $dbh  );
 9:     }
10: }
11:
12: if ( ! empty( $prices ) ) {
13:     while ( list ( $key, $val ) = each ( $prices ) )
14:         dba_replace( $key, $val, $dbh );
15: }
16:
17: if ( ! empty( $name_add ) && ! empty( $price_add ) )
18:     dba_insert( "$name_add", "$price_add", $dbh );
19: ?>
20:
21: <html>
22: <head>
23: <title>Listing 11.7 The complete product maintenance code</title>
24: </head>
25: <body>
26:
27: <form action="<? print $PHP_SELF; ?>" action="POST">
28:
29: <table border="1">
30: <tr>
31: <td>delete</td>
32: <td>product</td>
33: <td>price</td>
34: </tr>
35:
```

لیست ۷- ۱۱ برنامه کامل نگهداری محصولات

```
36: <?php
37: $key = dba_firstkey( $dbh );
38: while ( $key != false ) {
39:     $price = dba_fetch( $key, $dbh );
40:     print "<tr><td><input type='checkbox' name=\"delete[]\" ";
41:     print "value=\"$key\"></td>";
42:     print "<td>$key</td>";
43:     print "<td> <input type=\"text\" name=\"prices[$key]\" ";
44:     print "value=\"$price\"> </td></tr>";
45:     $key = dba_nextkey( $dbh );
46: }
47:
48: dba_close( $dbh );
49: ?>
50:
51: <tr>
52: <td> </td>
53: <td><input type="text" name="name_add"></td>
54: <td><input type="text" name="price_add"></td>
55: </tr>
56:
57: <tr>
58: <td colspan=3 align="right">
59: <input type="submit" value="amend">
60: </td>
61: </tr>
62:
63: </table>
64: </form>
65:
66: </body>
67: </html>
```

ادامه لیست ۷- ۱۱

جمع بندی

در درس این ساعت چگونگی بهره گیری از توابع کارآمد DBA در زبان PHP به‌منظور ذخیره و بازیابی داده‌ها مورد بحث و بررسی قرار گرفت. در این درس نحوه استفاده از تابع ( ) dba _ open جهت دستیابی به یک مرجع DBM تشریح شد. همان‌گونه که مشاهده کردید این مرجع به عنوان آرگومان مورد استفاده تمام توابع DBA قرار می‌گیرد. مطالب مفیدی درباره چگونگی واردکردن داده‌ها در بانک اطلاعاتی با استفاده از تابع ( ) dba _ insert، تغییر یا جایگزینی داده‌ها با استفاده از تابع dba_replace ( ) و حذف آنها از بانک اطلاعاتی با استفاده از تابع ( ) dba _ delete عنوان کردیم. همچنین مبحثی را به چگونگی بازیابی داده‌ها از بانک اطلاعاتی با استفاده از تابع ( ) dba _ fetch اختصاص دادیم. علاوه بر این در مورد چگونگی ذخیره و بازیابی داده‌هایی با ساختار پیچیده با استفاده از تابع ( ) serialize و ( ) unserialize بحث کردیم و در انتهای درس نیز با ارائه یک مثال جامع تمامی تکنیکهای درس را به‌کار گرفتیم.

توابع DBA همان‌گونه که احتمالاً تا به حال حدس زده‌اید، برای ذخیره داده‌ها در مقیاس کم مفید بوده و معمولاً مراجعات بسیار ساده‌ای برای دستیابی به این گونه داده‌ها صورت می‌گیرد. در دنیای واقعی نیاز ما معمولاً بیشتر از آنچه که در این درس مشاهده کردید، می‌باشد. در درس ساعت آینده دانش خود را در این زمینه گسترش داده و به بررسی یک بانک اطلاعاتی SQL از نوع کدباز با نام MySQL خواهیم پرداخت.

پرسش و پاسخ

پرسش: در چه مواقعی باید استفاده از یک بانک شبه DBM را به استفاده از یک بانک SQL ترجیح داد؟

پاسخ: بانکهای اطلاعاتی شبه DBM در مواردی مفید هستند که قصد ذخیره و بازیابی حجم نسبتاًکمی از داده را (که معمولاً به‌صورت زوجی شامل کلید دستیابی و مقدار در اختیارمان قرار می‌گیرند) داشته باشیم. برنامه هایی که جهت کار با بانکهای اطلاعاتی DBM توسعه یافته‌اند از قابلیت حمل بالایی برخوردارند. درصورتی که قصد شما ذخیره و بازیابی داده‌ها در مقیاس بزرگ باشد، استفاده از یک بانک اطلاعاتی SQL مانند MySQL را در دستور کارتان قرار دهید.

تمرینها

هدف از این بخش دوره مطالب فراگیری شده در قالب آزمون است. پاسخ بخش آزمون بلافاصله بعد از آن آمده است. بخش فعالیتها به منظور افزایش قابلیت برنامه‌نویسی خواننده طراحی شده و شامل تمرینهایی است که فاقد پاسخ می‌باشند.

آزمون

۱- جهت بازکردن یک بانک اطلاعاتی شبه DBM از کدام تابع DBA می‌توان استفاده کرد؟
۲- از کدام تابع DBA می‌توان برای وارد کردن داده‌ها به یک بانک اطلاعاتی شبه DBM استفاده کرد؟
۳- کدام تابع DBA را می‌توان جهت جایگزین کردن داده‌های قدیمی با داده های جدید در یک بانک اطلاعاتی شبه DBM مورد استفاده قرار داد؟
۴- روش دستیابی به داده‌های بانک اطلاعاتی از طریق کلید دستیابی کدام است؟
۵- روش دستیابی به اولین کلید دستیابی در بانک اطلاعاتی چیست؟
۶- روش دستیابی به سایر کلیدهای دستیابی کدام است؟
۷- روش حذف داده‌ها را از یک بانک اطلاعاتی شبه DBM بیان کنید؟

پاسخ آزمون

۱- با استفاده از تابع ( ) dba _ open می‌توان یک بانک اطلاعاتی را باز کرد.
۲- به کمک تابع ( ) dba _ insert می‌توان داده‌ها را به بانک اطلاعاتی اضافه کرد.
۳- به کمک تابع ( ) dba _ replace می‌توان داده‌های جدید را جایگزین داده‌های قدیمی در بانک اطلاعاتی کرد.
۴- با استفاده از تابع ( ) dba _ fetch و ارسال کلید دستیابی داده مورد نظر و نام مرجع DBM به‌عنوان آرگومان می‌توان آن داده را مورد دستیابی قرار داد.
۵- تابع ( ) dba _ firstkey کلید دستیابی اولین داده ذخیره شده در بانک اطلاعاتی را باز می‌گرداند.
۶- پس از فراخوانی تابع ( ) dba _ firstkey جهت دستیابی به اولین کلید می‌توان کلیدهای بعدی را به‌طور متوالی با فراخوانی تابع ( ) dba _ nextkey مورد دستیابی قرار داد.
۷- به کمک تابع ( ) dba _ delete می‌توان داده مورد نظر را از بانک اطلاعاتی حذف کرد.

فعالیتها

۱- جهت نگهداری کلمات عبور کاربران یک سایت اینترنتی فرضی، یک بانک اطلاعاتی ایجاد کنید. سپس برنامه‌ای بنویسید تا کاربران با استفاده از آن بتوانند خود اقدام به وارد کردن کلمات عبور در بانک اطلاعاتی نمایند. مسأله مقادیر تکراری را به‌گونه‌ای مناسب حل کنید.

۲- برنامه‌ای برای شناسایی کاربران براساس بانک اطلاعاتی کلمات عبور که در تمرین قبل آن‌را ایجاد کردید، بنویسید. چنانچه کلمه عبور وارد شده توسط کاربر در بانک اطلاعاتی موجود باشد پیغام مناسبی را جهت اطلاع وی نمایش دهید. در غیر این‌صورت باید فرم مربوط به ورود کلمه عبور را مجدداً در اختیار وی قرار دهید.

ساعت دوازدهم

بهره‌گیری از بانکهای اطلاعاتی SQL

یکی از ویژگی‌های شاخص زبان PHP توانایی آن در کار با بانکهای اطلاعاتی است. PHP تسهیلات قابل توجهی را در رابطه با اتصال به بانکهای اطلاعاتی و کار با داده‌های موجود در آنها در اختیار کاربران قرار می‌دهد. در درس این ساعت قصد ما این است که این تسهیلات را در رابطه با یکی از بانکهای اطلاعاتی کدباز و متداول با عنوان MySQL مورد بررسی قرار دهیم. این امکانات در رابطه با هر نوع دیگری از بانکهای اطلاعاتی که PHP آن را پشتیبانی کند، وجود خواهند داشت. به‌عبارت دیگر، PHP با تمامی بانکهای اطلاعاتی از طریق یک زبان مشترک سخن می‌گوید. اما دلیل انتخاب بانک اطلاعاتی MySQL چیست؟ پیش از هر چیز ویژگی مشترک MySQL با زبان PHP یعنی قیمت رایگان آن در عین توانمندی و قابلیت انکار ناپذیر برای توسعه پروژه‌های برنامه‌نویسی بزرگ به‌چشم می‌آید. (هر دو نرم‌افزار از نوع کدباز هستند)؛ ضمن اینکه نسخه‌های مختلفی از MySQL با توانمندی‌های کم و بیش یکسان برای محیطهای عامل مختلف توسعه یافته است. برای دستیابی رایگان به این بانک اطلاعاتی کافی است تا سری به آدرس http: // www. mysql. com بزنید.

در درس این ساعت مباحث زیر را مورد بررسی قرار خواهیم داد:

- **بررسی چند مثال در رابطه با زبان SQL**
- **نحوه اتصال به سرور بانک اطلاعاتی MySQL**
- **نحوه انتخاب یک بانک اطلاعاتی**
- **کنترل خطا**
- **نحوه اضافه کردن داده‌ها به یک جدول**
- **نحوه بازیابی داده‌ها از یک جدول**
- **نحوه تغییر داده‌های یک جدول**
- **بررسی ساختار بانکهای اطلاعاتی**
- **نحوه مکانیزه کردن پرس و جوها**

در ادامه به بررسی موارد فوق می‌پردازیم.

مقدمه‌ای کوتاه بر SQL

SQL کوتاه شده عبارت Structured Query Language و به معنی زبان پرس و جوی ساخت یافته است. این زبان دارای گرامر استاندارد و واحدی است که با استفاده از آن می‌توان هر نوع بانک اطلاعاتی موجود را مورد پرس و جو قرار داد (پرس و جو یا Query واژه‌ای عمومی برای درخواستی است که یک برنامه یا کاربر از بانک اطلاعاتی طلب می‌کند. این درخواست می‌تواند شامل بازیابی، حذف، ذخیره و تغییر داده‌ها و یا عملیاتی در مورد خود ساختار بانک اطلاعاتی باشد، مانند ایجاد یا حذف جداول ـ مترجم). همان‌گونه که اغلب مرورگرهای اینترنت با اضافه کردن امکاناتی به زبان نشانه‌گذاری HTML آن را توسعه داده‌اند، بیشتر بانکهای اطلاعاتی SQL نیز با افزودن امکانات مورد نظر خود به این زبان پرس و جوی استاندارد آن‌را توسعه می‌دهند. با این حال اطلاع از دستور زبان یا گرامر SQL به این معنی است که می‌توان با طیف گسترده‌ای از بانکهای اطلاعاتی تحت محیطهای عامل مختلف کار کرد.

در این کتاب حتی مجال پرداختن به اسامی دستورات این زبان هم وجود ندارد. با این حال طی این ساعت به‌طور عام SQL و به‌طور خاص نیز تا حد ممکن به MySQL خواهیم پرداخت.

همان‌گونه که پیشتر نیز عنوان شد، MySQL یک بانک اطلاعاتی کدباز است که با استفاده از زبان SQL می‌توان آن را مورد پرس‌وجو قرار داد. MySQL بر روی سیستم به‌عنوان "شبح" سرویس‌دهنده‌ای که کاربران سایر ماشینها یا کاربر همان سرور می‌توانند به آن متصل شوند، اجرا می‌شود (در سیستم عامل UNIX واژه شبح یا demon به برنامه‌ای اطلاق می‌شود که دور از چشمان کاربر اجرا شده و خدمات و امکاناتی را در اختیار وی قرار می‌دهد. این اشباح را می‌توان به سرویسهای مختلف سیستم عامل Windows تشبیه کرد ـ مترجم). پس از برقراری اتصال با سرور در صورت داشتن مجوز لازم می‌توان بانک اطلاعاتی مورد نظر را انتخاب کرد.

در داخل هر بانک اطلاعاتی جداولی موجود است که شامل داده‌ها و اطلاعات می‌باشند. هر جدولی از ترکیب سطرها و ستونها تشکیل می‌شود. محل تقاطع یک سطر با یک ستون محلی است که می‌توان یک عنصر داده‌ای را در آن ذخیره کرده و یا درصورت وجود آن‌را بازیابی کرد. هر ستون از جدول تنها قادر به پذیرش یک نوع داده از پیش تعیین شده است. برای مثال نوع داده INT جهت ذخیره اعداد صحیح و نوع داده VARCHAR برای ذخیره تعدادی کاراکتر که دارای محدودیت بوده و دنباله‌ای از کاراکترها را تشکیل می‌دهند، مورد استفاده قرار می‌گیرند.

به‌منظور ایجاد یک جدول در بانک اطلاعاتی انتخاب شده، لازم است پرس و جوی مناسبی ایجاد و برای اجرا به بانک مورد نظر ارسال گردد. به نمونه زیر در این رابطه توجه کنید:

```
CREATE TABLE mytable (first _ name VARCHAR (30) ,
    Second _ name VARCHAR (30), age INT) ;
```

جدول ایجاد شده توسط این پرس و جو شامل سه ستون خواهد بود. ستونهای first _ name و second _ name هر یک دنباله‌ای کاراکتری با محدودیت ۳۰ کاراکتر بوده و ستون age نیز شامل یک عدد صحیح خواهد بود.

جهت درج داده‌ها در جدول اخیر می‌توان از عبارت INSERT به صورت زیر استفاده کرد:

```
INSERT INTO mytable (first _ name, second _ name, age)
    VALUES ('John' , 'Smith' , 36) ;
```

اسامی فیلدهایی از جدول mytable را که مایلیم مقداردهی کنیم در داخل اولین پرانتز و مقادیر مربوط به هریک از آنها را در داخل دومین پرانتز می‌نویسیم. وجود رابطه‌ای یک به یک مابین فیلدها و مقادیر الزامی است.

برای بازیابی داده‌ها از یک جدول از عبارت SELECT استفاده می‌کنیم:

```
SELECT * FROM mytable ;
```

علامت ستاره در عبارت فوق، یک کاراکتر جانشین بوده و به معنی " همه فیلدها " می‌باشد. جهت بازیابی مقدار یک فیلد خاص از جدول کافی است تا نام آن فیلد را با علامت ستاره در عبارت فوق تعویض کنیم:

```
SELECT age FROM mytable ;
```

جهت تغییر داده‌های ذخیره شده در یک جدول از عبارت UPDATE استفاده می‌کنیم:

```
UPDATE mytable SET first _ name = ' Bert ' ;
```

عبارت فوق مقدار فیلد first _ name تمامی سطرهای جدول mytable را به ' Bert ' تغییر می‌دهد. با استفاده از بخش WHERE در عبارات SELECT و UPDATE می‌توان حوزه عملیات آنها را به‌طور دقیق‌تری تعریف نمود. برای مثال عبارت زیر:

```
UPDATE  mytable SET first _ name = "Bert" WHERE second _ name = "Baker" ;
```

مقدار فیلد first _ name سطرهایی از جدول را که مقدار فیلد second _ name آنها برابر با "Baker" است به مقدار ' Bert ' تغییر می‌دهد.

برای کسب اطلاعات بیشتر در مورد SQL پیشنهاد می‌کنیم کتاب" خودآموز SQL در ۲۱ روز" نوشته Ryan K . Stephens را مطالعه کنید.

اتصال به سرور بانک اطلاعاتی

پیش از آنکه بتوانید با بانک اطلاعاتی مورد نظرتان کار کنید، لازم است تا ابتدا به سرور بانک اطلاعاتی (همان شبحی که در قسمت قبل توضیح دادیم) متصل شوید. PHP تابعی دارد که صرفاً به همین منظور طراحی شده است. نام این تابع mysql _ connect ( ) است. تابع mysql _ connect ( ) ارسال هیچ آرگومانی را تحمیل نمی‌کند اما در صورت نیاز برنامه‌نویس می‌تواند حداکثر سه آرگومان به این تابع ارسال کند. این سه آرگومان مشخص کننده نام میزبان، نام کاربر و کلمه عبور می‌باشند.

درصورتی که مقادیر این سه آرگومان توسط برنامه‌نویس مشخص نشود PHP به‌طور خودکار مقدار پیش‌فرض localhost را برای نام میزبان مورد استفاده قرار داده و فرض می‌کند که هیچ مقداری برای نام کاربر و کلمه عبور در جدول user از بانک اطلاعاتی mysql درج نشده است مگر آنکه این مقادیر در فایل php. ini مشخص شده باشند. طبیعی است که بهره‌گیری از مقادیر پیش‌فرض جز برای تست عملکرد بانک اطلاعاتی کار عاقلانه‌ای نیست. از این‌رو در مثالهای خود از مقادیری برای این دو آرگومان بهره خواهیم گرفت. چنانچه تابع کار خود را با موفقیت انجام دهد، مرجعی را به برنامه فراخواننده باز می‌گرداند. معمولاً این مقدار بازگشتی را در یک متغیر ذخیره می‌کنیم. مادامی که این مرجع در دست باشد می‌توانیم به بهره‌گیری از بانکهای اطلاعاتی موجود بر روی سرور ادامه دهیم.

قطعه کد زیر با استفاده از تابع ( ) mysql _ connect اتصالی را با سرور بانک اطلاعاتی MySQL برقرار می‌کند، مقدار بازگشتی از تابع در متغیر $link نگهداری می‌شود:

```
$link = mysql _ connect ( "localhost", "root", "n1ckel") ;
if ( ! $link) {
   die ("couldn't connect to MySQL") ;
```

چنانچه از زبان PHP در کنار وب سرور Apache استفاده می‌کنید، برای اتصال به سرور بانک اطلاعاتی از تابع ( ) mysql _ pconnect نیز می‌توانید بهره ببرید. از دیدگاه برنامه نویسی عملکرد این تابع عملکردی کاملاً مشابه با تابع ( ) mysql _ connect دارد. اما در حقیقت تفاوت مهمی در این میان وجود دارد. اگر از تابعی که به‌واسطه بهره‌گیری از وب سرور Apache در اختیارتان قرارگرفته، یعنی ( ) mysql _ pconnect جهت اتصال با سرور بانک اطلاعاتی استفاده کنید، حتی پس از خاتمه اجرای برنامه یا فراخوانی تابع ( ) mysql _ close (که موجب قطع اتصال با سرور بانک اطلاعاتی MySQL می‌گردد)، اتصال کماکان به حیات خود ادامه خواهد داد. این مطلب البته به معنای فعال بودن اتصال نیست بلکه برای فعال شدن آن لازم است تا فرآیند دیگری تابع ( ) mysql _ pconnect را باردیگر فراخوانی کند. به بیان بهتر، به‌واسطه استفاده از تابع ( ) mysql _ pconnect سربار ناشی از بازکردن یک اتصال جدید با سرور به برنامه تحمیل نمی‌شود چراکه فراخوانی پیشین برنامه اتصال را به صورت باز به حال خود رها کرده است.

انتخاب یک بانک اطلاعاتی

اکنون‌که اتصال ما با سرور بانک اطلاعاتی (شبح MySQL) برقرار شده است، لازم است تا بانک اطلاعاتی مورد نظرمان که قصد کار با آن را داریم، مشخص کنیم. به کمک تابع ( ) mysql_select_db می‌توانیم بانک مورد نظر را تعیین کنیم. تابع ( ) mysql_select_db به نام بانک اطلاعاتی به عنوان آرگومان اجباری نیاز دارد. البته می‌توان نام یک مرجع معتبر (که از فراخوانی تابع تشریح شده در

قسمت قبل یعنی mysql _ connect ( ) یا mysql _ pconnect ( ) به دست آمده) را نیز به عنوان آرگومان اختیاری به این تابع ارسال کنیم. اگر از ارسال نام این مرجع به تابع مورد بحث خودداری کنیم مرجع بازگشتی حاصل از آخرین فراخوانی تابع mysql _ connect ( ) یا mysql _ pconnect ( ) مورد استفاده قرار خواهد گرفت. در صورتی که بانک اطلاعاتی مورد نظر موجود بوده و مانعی بر سر دستیابی به آن موجود نباشد تابع mysql_select_db ( ) مقدار true و در غیر این‌صورت مقدار false را باز می‌گرداند. در قطعه کد زیر بانک اطلاعاتی sample با استفاده از تابع مورد بحث انتخاب شده است:

```
$database = "sample" ;
mysql_select_db ($database) or die ("couldn't open $database") ;
```

اشکال یابی

کاری که در قسمت قبل انجام دادیم عبارت بود از بررسی مقدار بازگشتی حاصل از توابع MySQL و فراخوانی تابع مرگبار ( ) die جهت خاتمه برنامه در صورتی که این مقدار بازگشتی false بود. درمواقعی شاید بهتر باشد تا حین خاتمه برنامه پیغام مفیدی را نیز که بیانگر علت وقوع خطاست بر روی صفحه مرورگر نمایش دهیم. بانک اطلاعاتی MySQL به محض ناکامی یک عملیات، شماره‌ای را به‌عنوان شماره خطا و دنباله کاراکتری را نیز به‌عنوان پیغام خطا در جایی از حافظه ثبت می‌کند. به کمک دو تابع مفید mysql _ errno ( ) و mysql _ error ( ) می‌توان به راحتی شماره خطا و دنباله کاراکتری مربوطه را مورد دستیابی قرار داد. برنامه موجود در لیست ۱- ۱۲ با جمع بندی مثالهای بررسی شده تا بدین جای درس اقدام به برقراری سرور MySQL و انتخاب بانک اطلاعاتی مورد نظر می‌کند. در این برنامه جهت نمایش پیغامهای خطای احتمالی از تابع mysql _ error ( ) استفاده شده است. در خط ۱۱ برنامه فرآیند اتصال با سرور MySQL انجام می‌شود. چنانچه این فرآیند موفقیت‌آمیز باشد در خط ۱۵ برنامه بانک اطلاعاتی مورد نظر انتخاب می‌شود. همان‌گونه که ملاحظه می‌کنید تابع mysql _ close ( ) در خط ۱۸ اقدام به بستن (تخریب) اتصال موجود با سرور MySQL می‌کند.

```
 1: <html>
 2: <head>
 3: <title>Listing 12.1 Opening a connection and
 4: selecting a database</title>
 5: </head>
 6: <body>
 7: <?php
 8: $user = "harry";
 9: $pass = "elbomonkey";
10: $db = "sample";
11: $link =  mysql_connect( "localhost", $user, $pass  );
12: if ( ! $link )
```

لیست ۱-۱۲برنامه‌ای که‌پس‌از برقراری اتصال با سرور MySQL اقدام به انتخاب بانک اطلاعاتی می‌کند

```
13:     die( "Couldn't connect to MySQL" );
14: print "Successfully connected to server<P>";
15: mysql_select_db( $db )
16:     or die ( "Couldn't open $db: ".mysql_error() );
17: print "Successfully selected database \"$db\"<P>";
18: mysql_close( $link );
19: ?>
20: </body>
21: </html>
```

ادامه لیست ۱-۱۲

چنانچه در خط ۱۰ از این برنامه مقدار متغیر $db را به "notthere" تغییر دهیم تابع mysql_select_db ( ) تلاش خواهد کرد تا بانکی را انتخاب کند که بر روی سیستم موجود نمی‌باشد. در چنین حالتی خروجی حاصل از فراخوانی تابع die ( ) به صورت زیر خواهد بود:

Couldn't open sample 2 : Access denied for user : ' harry @ localhost '
to database ' notthere '

افزودن داده‌ها به جدولی از بانک اطلاعاتی

اکنون که به بانک اطلاعاتی مورد نظرمان دسترسی داریم می‌توانیم اطلاعاتی را به جدولی از آن بانک اضافه کنیم. در مورد مثال زیر تصور کنید که قصد ما طراحی یک وب سایت است که امکانات مختلفی را در رابطه با خرید فضا بر روی اینترنت و ثبت نام حوزه (Domain) در اختیار کاربران قرار می‌دهد.

پیشتر جدولی با نام domains در بانک اطلاعاتی sample ایجاد کرده‌ایم. این جدول شامل چهار ستون است. ستون اول کلید اصلی (primary key) جدول بوده و id نام دارد. مقدار این ستون از نوع عددی بوده و به محض ورود اطلاعات (یک سطر) به جدول یک واحد افزایش می‌یابد. ستون دوم با نام domain شامل تعداد متغیری از کاراکترها می‌باشد (VARCHAR). نام ستون سوم sex بوده و شامل یک کاراکتر ساده است. ستون چهارم با عنوان mail شامل آدرس پستی کاربر است. پرس و جوی SQL زیر در بانک اطلاعاتی MySQL موجب ایجاد چنین جدولی می‌شود:

```
Creat table domains (id INT NOT NULL AUTO _ INCREMENT ,
    PRIMARY KEY (id) ,
    domain VARCHAR (20) ,
    sex CHAR (1) ,
    mail VARCHAR (20) ) ;
```

جهت درج داده‌ها در این جدول لازم است تا یک پرس و جوی SQL دیگر ایجاد کرده و آن‌را اجرا نماییم. تابع mysql _ query ( ) در زبان PHP برای این منظور پیش بینی شده است. تابع mysql_ query ( ) به یک پرس و جوی SQL به عنوان آرگومان اجباری نیاز دارد. درصورت تمایل برنامه نویس می‌تواند نام یک مرجع را نیز به‌عنوان یک آرگومان اختیاری به این تابع ارسال نماید.

چنانچه از این آرگومان اختیاری استفاده نشود، نام مرجعی که آخرین بار به واسطه فراخوانی یکی از دو تابع mysql _ connect ( ) یا mysql _ pconnect ( ) به دست آمده، مورد استفاده قرار می‌گیرد. چنانچه فراخوانی تابع musql_ query ( ) موفقیت آمیز باشد این تابع یک مقدار مثبت باز می‌گرداند. در صورتی که عبارت پرس و جوی SQL شامل یک خطای دستوری بوده یا مجوزهای لازم جهت دستیابی به بانک اطلاعاتی در اختیار نباشد تابع مذکور مقدار false را باز می‌گرداند. توجه کنید که اجرای موفقیت‌آمیز یک پرس و جو لزوماً به معنی تغییر در داده‌های موجود در جدول نمی‌باشد. برنامه موجود در لیست ۲- ۱۲ برنامه لیست قبلی را اندکی تغییر داده است. پرس و جوی مورد نظر که جهت اضافه کردن داده‌ها به جدول طراحی شده در خط ۱۵ به متغیری با نام $query نسبت داده شده است. تابع mysql_ query ( ) در خط ۱۷ این پرس و جو را بر روی بانک sample اجرا می‌کند.

```
 1: <html>
 2: <head>
 3: <title>Listing 12.2 Adding a row to a table</title>
 4: </head>
 5: <body>
 6: <?php
 7: $user = "harry";
 8: $pass = "elbomonkey";
 9: $db = "sample";
10: $link =  mysql_connect( "localhost", $user, $pass  );
11: if ( ! $link )
12:     die( "Couldn't connect to MySQL" );
13: mysql_select_db( $db, $link )
14:        or die ( "Couldn't open $db: ".mysql_error() );
15: $query = "INSERT INTO domains ( domain, sex, mail )
16:        values( '123xyz.com', 'F', 'sharp@adomain.com' )";
17: mysql_query( $query, $link )
18: or die ( "Couldn't add data to \"domains\" table: "
19: .mysql_error() );
20: mysql_close( $link );
21: ?>
22: </body>
23: </html>
```

لیست ۲- ۱۲ اضافه کردن سطری از داده‌ها به جدول

توجه کنید که در پرس و جوی خط ۱۵ هیچ تدبیری جهت مقداردهی ستون id در جدول domains اندیشیده نشده است. مقدار این ستون جدول به طور خودکار افزایش می‌یابد.

واضح است که با هربار اجرای برنامه لیست ۲- ۱۲ داده‌های ثابتی به جدول domains اضافه می‌شوند. برنامه لیست ۳- ۱۲ ترتیبی می‌دهد تا داده‌های ورودی که از طریق یک فرم وارد برنامه می‌شوند، به جدول اضافه شوند.

```
<html>
<head>
<title>Listing 12.3 Adding user input to a database</title>
</head>
<body>
<?php
if ( isset( $domain ) && isset( $sex ) && isset( $domain ) ) {
    // check user input here!
    $dberror = "";
    $ret = add_to_database( $domain, $sex, $mail, $dberror );
    if ( ! $ret )
        print "Error: $dberror<BR>";
    else
        print "Thank you very much";
} else {
    write_form();
}

function add_to_database( $domain, $sex, $mail, &$dberror ) {
    $user = "harry";
    $pass = "elbomonkey";
    $db = "sample";
    $link =  mysql_pconnect( "localhost", $user, $pass  );
    if ( ! $link ) {
        $dberror = "Couldn't connect to MySQL server";
        return false;
    }
    if ( ! mysql_select_db( $db, $link ) ) {
        $dberror = mysql_error();
        return false;
    }
    $query = "INSERT INTO domains ( domain, sex, mail )
            values( '$domain', '$sex', '$mail' )";
    if ( ! mysql_query( $query, $link ) ) {
        $dberror = mysql_error();
        return false;
    }
    return true;
}

function write_form() {
    global $PHP_SELF;
    print "<form method=\"POST\">\n";
    print "<input type=\"text\" name=\"domain\"> ";
    print "The domain you would like<p>\n";
    print "<input TYPE=\"text\" name=\"mail\"> ";
    print "Your mail address<p>\n";
    print "<select name=\"sex\">\n";
    print "\t<option value=\"F\"> Female\n";
    print "\t<option value=\"M\"> Male\n";
    print "</select>\n";
    print "<input type=\"submit\" value=\"submit!\">\n</form>\n";
}
?>
</body>
</html>
```

لیست ۳- ۱۲ اضافه کردن داده‌های ورودی به جدولی از بانک اطلاعاتی

جهت خلاصه سازی یک بخش مهم، یعنی بررسی و ارزیابی داده‌های ورودی را از این برنامه حذف کرده‌ایم و فرض را بر این نهاده‌ایم که کاربر داده‌های مناسبی را وارد برنامه می‌کند. اما توجه داشته باشید که در دنیای واقعی هر نوع ورودی را باید به‌دقت مورد ارزیابی و بررسی قرار دهید. در درس ساعت هفدهم با عنوان " بهره‌گیری از دنباله‌های کاراکتری" روند ارزیابی داده‌ها را با استفاده از توابعی که در آنجا معرفی خواهیم کرد، مورد بررسی قرار می‌دهیم.

همان‌گونه که مشاهده می‌کنید در خط ۷ از این برنامه وجود مقادیر در متغیرهای $domain، $sex و $mail را مورد بررسی قرار دادیم. اگر مقادیر این متغیرها مشخص شده باشند، می‌توانیم مطمئن باشیم که کاربر داده‌های مورد نظرش را وارد برنامه کرده است. بدین ترتیب با فراخوانی تابع add _ to _ database ( ) در خط ۱۰ از برنامه می‌توانیم داده‌های مذکور را به بانک اطلاعاتی اضافه کنیم.

تابع add _ to _ database ( ) در خط ۱۹ برنامه تعریف شده و همان‌گونه که می‌بینید به چهار آرگومان ورودی نیاز دارد. سه آرگومان نخست با اسامی $domain، $sex و $mail همان مقادیر وارد شده توسط کاربر بوده و آرگومان چهارم با نام $dberror یک دنباله کاراکتری است که در مواقع بروز خطا مقدار آن شامل توصیف خطا خواهد بود. از این جهت آرگومان مذکور را از طریق مرجع به تابع ارسال کرده‌ایم تا هرگونه تغییری بر روی مقدار این آرگومان در تابع add _ to _ database ( ) در مقدار اصلی عیناً منعکس شود. در خط ۲۳ تلاشی را جهت اتصال به سرور بانک اطلاعاتی (شبح MySQL) صورت داده‌ایم. چنانچه این تلاش با شکست مواجه شود، متغیر$dberror را با توصیفی از خطا مقداردهی کرده و ضمن بازگرداندن مقدار false از تابع add _ to _ database ( ) به اجرای آن خاتمه می‌دهیم. در خط ۲۸ بانک اطلاعاتی مورد نظر را که شامل جدول domains است، انتخاب کرده و یک پرس و جوی SQL را جهت درج مقادیر ورودی کاربر تشکیل داده‌ایم. این پرس و جو را در خط ۳۴ به تابع mysql _ query ( ) ارسال کرده‌ایم. اگر چنانچه اجرای یکی از دو تابع mysql _ select _ db ( ) یا mysql _ query ( ) با شکست مواجه شود، ضمن بازگرداندن مقدار false از تابع add _ to_database ( ) مقدار بازگشتی از تابع mysql _ error ( ) را به متغیر $dberror نسبت می‌دهیم. با فرض اینکه همه عملیات کاملاً موفقیت‌آمیز و طبق انتظار باشد تابع add _ to _ database ( ) در خط ۳۸ برنامه مقدار true را به برنامه فراخواننده خود باز می‌گرداند.

در خط ۱۱ از برنامه اصلی مقدار بازگشتی از تابع add _ to _ database ( ) مورد ارزیابی قرار می‌گیرد. اگر این مقدار برابر با true باشد، می‌توانیم مطمئن شویم که عملیات درج داده‌ها در بانک اطلاعاتی موفقیت‌آمیز بوده است. در چنین شرایطی تابع print پیغام تشکرآمیزی را در خط ۱۴ به خروجی می‌فرستد. درغیر این‌صورت پیغام خطایی بر روی صفحه مرورگر نقش خواهد بست. از آنجا که

در این شرایط متغیر $dberror (که ما آن‌را در فراخوانی تابع add _ to _ database ( ) به آن ارسال کردیم) شامل اطلاعات مفیدی می‌باشد در محتوای پیغام خطا از این اطلاعات استفاده کرده‌ایم.

درصورتی که حاصل بررسی وجود مقدار در متغیرهای $domain، $sex و $mail که در قالب یک ساختار if در خط ۷ انجام شده است، نشان دهد که کاربر هیچ مقداری را جهت ورود به برنامه مشخص نکرده، با استفاده از تابع دیگری که در خط ۱۶ فراخوانی می‌کنیم، یک فرم HTML را جهت وارد کردن مقادیر پیش روی او نمایش می‌دهیم. تعریف این تابع با نام write _ form ( ) در خط ۴۱ از برنامه آمده است.

دستیابی به مقدار فیلدی از جدول که محتوای آن به‌طور خودکار افزایش می‌یابد

در مثالهای قبل برنامه‌های خود را بدون توجه و نگرانی در مورد ستون id ازجدول که مقدار آن به‌طور خودکار به‌محض ورود داده‌ها در سطری از جدول اضافه می‌شد، توسعه دادیم. چنانچه به اطلاعات ذخیره شده در این ستون به هر دلیلی نیاز داشته باشیم، می‌توانیم با استفاده از یک پرس و جوی SQL مقدار آن‌را از جدول مورد نظر بازیابی کنیم. اما اگر به مقادیر این ستون به محض ورود اطلاعات در سطرهای جدول نیاز داشته باشیم چنین پرس و جویی به کار ما نمی‌آید، چرا که این پرس و جو مقادیر تمامی اطلاعات این ستون را به یکباره در اختیارمان قرار می‌دهد. خوشبختانه برای غلبه براین مشکل در PHP تابع مفیدی با عنوان mysql _ insert _ id ( ) پیش‌بینی شده که مقدار یک ستون کلیدی از جدول را که محتوای آن به‌طور خودکار با ورود اطلاعات یک سطر افزایش می‌یابد، در اختیارمان می‌گذارد. این تابع یک آرگومان اختیاری می‌پذیرد که همان مرجع بازگشتی از تابع mysql_connect ( ) یا mysql _ pconnect ( ) است. باردیگر، درصورتی که از این آرگومان استفاده نشود آخرین مرجع بازگشتی از این توابع مورد استفاده قرار خواهد گرفت.

بنابراین اگر بخواهیم شماره‌ای را که به سفارش کاربر اختصاص دادیم نمایش دهیم، می‌توانیم پس از درج داده‌ها در جدول موردنظر تابع mysql _ insert _ id ( ) را مستقیماً فراخوانی کنیم:

```
$query = "INSERT INTO domains (domain, sex, mail)
      values (' $domain ', ' $sex ', ' $mail ')" ;
mysql _ query ($query, $link) ;
$id = mysql _ insert _ id ( ) ;
print "Thank you. Your transaction number is $id. please.
   Quote it in any queries." ;
```

دستیابی به اطلاعات

اکنون که با چگونگی درج داده‌ها در جداول بانک اطلاعاتی آشنا شدیم، باید روشی برای بازیابی آنها بیابیم. همان‌گونه که احتمالاً تا به حال حدس زده‌اید، می‌توانیم برای انجام این کار با استفاده از تابع ( ) mysql _ query یک پرس و جو از نوع SELECT ترتیب دهیم. با این حال چگونه می‌توانیم یک همچون پرس و جویی از نوع SELECT را با موفقیت اجرا کنیم؟ تابع ( ) mysql _ query مرجعی را به نتیجه حاصل از اجرای پرس و جوی مذکور باز می‌گرداند. با ارسال این مرجع به عنوان آرگومان به سایر توابعی که به این منظور طراحی شده‌اند، می‌توانیم به مجموعه جواب دسترسی پیدا کنیم.

تعیین تعداد سطرهای بازیابی شده توسط پرس و جوی SELECT

با استفاده از تابع ( ) mysql _ num _ rows می‌توانیم از تعداد سطرهای بازیابی شده توسط پرس و جوی SELECT اطلاع حاصل کنیم. این تابع مرجع بازگشتی از تابع ( ) mysql _ query را به‌عنوان آرگومان پذیرفته و تعداد سطرهای بازیابی شده توسط پرس و جوی SELECT را باز می‌گرداند. برنامه موجود در لیست ۴- ۱۲ با استفاده از یک پرس و جوی SELECT کلیه سطرهای جدول domains را بازیابی کرده و به کمک تابع ( ) mysql _ num _ rows تعداد سطرهای بازیابی شده را در اختیار برنامه قرار می‌دهد. اگر تمام آن چیزی که بدان نیاز داریم تعداد سطرهای بازیابی شده توسط پرس و جو باشد، بهتر است بدانیم که وقت خود را با این کار تلف کرده‌ایم چراکه اطلاع از این مقدار به خودی خود منفعتی را در پی ندارد. از طرف دیگر تابعCOUNT از MySQL روش بسیار ساده‌تری جهت اطلاع از تعداد سطرها در اختیارمان قرار می‌دهد. با این حال اگر قصد ما پردازش و انجام عملیات مورد نظر بر روی سطرهای بازگشتی باشد، وضعیت اندکی فرق می‌کند. شاید مایل باشیم تا محتوای سطرهای بازیابی شده را در قالب یک جدول بر روی صفحه مرورگر نمایش دهیم. با فرض در دست داشتن اطلاعات بازیابی شده توسط پرس و جوی SELECT می‌توانیم با فراخوانی تابع ( ) mysql_ num _ rows به اطلاعات جالبی در مورد داده‌های بازیابی شده دسترسی، پیدا کنیم.

```
 1: <html>
 2: <head>
 3: <title>Listing 12.4 Using mysql_num_rows()</title>
 4: </head>
 5: <body>
 6: <?php
 7: $user = "harry";
 8: $pass = "elbomonkey";
 9: $db = "sample";
10: $link =  mysql_connect( "localhost", $user, $pass  );
11: if ( ! $link )
12:     die( "Couldn't connect to MySQL" );
```

لیست ۴- ۱۲ اطلاع از تعداد سطرهای بازیابی شده توسط پرس و جوی SELECT

```
13: mysql_select_db( $db, $link )
14:     or die ( "Couldn't open $db: ".mysql_error() );
15: $result = mysql_query( "SELECT * FROM domains" );
16: $num_rows = mysql_num_rows( $result );
17: print "There are currently $num_rows rows in the table<P>";
18: //
19: // Further work with the $result resource here
20: //
21: mysql_close( $link );
22: ?>
23: </body>
24: </html>
```

ادامه لیست ۴- ۱۲

تابع ( ) mysql _ query مرجعی را جهت دستیابی به مجموعه جواب حاصل از اجرای پرس و جو باز می‌گرداند. همان‌گونه که مشاهده می‌کنید این مرجع هنگام فراخوانی( ) mysql_ num _ rows به‌عنوان آرگومان مورد استفاده قرار می‌گیرد. تابع نامبرده نیز تعداد سطرهای بازیابی شده را به برنامه فراخواننده باز می‌گرداند.

در خط ۱۰ از این برنامه، فرآیند اتصال به سرور بانک اطلاعاتی و در خط ۱۳ نیز فرآیند انتخاب بانک اطلاعاتی مورد نظر انجام شده است. در خط ۱۵ پرس و جوی SELECT به عنوان آرگومان ( ) mysql _ query ارسال شده است. این تابع مرجعی را به مجموعه جواب حاصل از پرس و جو باز می‌گرداند که می‌توان آن‌را در خط ۱۶ به‌عنوان آرگومان تابع ( ) mysql_ num _ rows مورد استفاده قرار داد. با در دست داشتن تعداد سطرهای موجود در مجموعه جواب حاصل از پرس و جوی SELECT می‌توانیم عملیات مورد نیاز خود را بر روی این مجموعه انجام دهیم. قسمت بعدی درس به همین امر اختصاص یافته است.

دستیابی به مجموعه جواب

پس از اجرای موفقیت آمیز پرس و جوی SELECT و در دست‌گرفتن مرجع مربوط به آن، می‌توان با استفاده از یک ساختار تکرار به هر یک از سطرهای جدول دسترسی پیدا کرد. PHP جهت ثبت موقعیت فعلی در مجموعه جواب از یک اشاره‌گر درونی (سیستمی) بهره می‌گیرد. به موازات دستیابی برنامه به هریک از سطرهای مجموعه جواب، اشاره‌گر مذکور سطر بعدی را مشخص می‌کند.

با بهره‌گیری از تابع ( ) mysql_ fetch _ rows به سادگی می‌توان به آرایه‌ای که عناصر آن‌را مقادیر ستونهای هر سطر از مجموعه جواب تشکیل می‌دهد، دست یافت. این تابع با دریافت مرجعی به مجموعه جواب، آرایه‌ای را که شامل فیلدهای هر سطر از جدول است، باز می‌گرداند. با رسیدن به انتهای مجموعه جواب این تابع مقدار false را باز می‌گرداند. برنامه موجود در لیست ۵- ۱۲ به این روش کلیه اطلاعات جدول domains را بر روی صفحه مرورگر نمایش می‌دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 12.5 Listing all rows and fields in a table</title>
 4: </head>
 5: <body>
 6: <?php
 7: $user = "harry";
 8: $pass = "elbomonkey";
 9: $db = "sample";
10: $link =  mysql_connect( "localhost", $user, $pass  );
11: if ( ! $link )
12:     die( "Couldn't connect to MySQL" );
13: mysql_select_db( $db, $link )
14:     or die ( "Couldn't open $db: ".mysql_error() );
15: $result = mysql_query( "SELECT * FROM domains" );
16: $num_rows = mysql_num_rows( $result );
17: print "There are currently $num_rows rows in the table<P>";
18: print "<table border=1>\n";
19: while ( $a_row = mysql_fetch_row( $result ) ) {
20:     print "<tr>\n";
21:     foreach ( $a_row as $field )
22:         print "\t<td>$field</td>\n";
23:     print "</tr>\n";
24: }
25: print "</table>\n";
26: mysql_close( $link );
27: ?>
28: </body>
29: </html>
```

لیست ۵- ۱۲ نمایش تمام اطلاعات یک جدول

همان‌گونه که در این برنامه مشاهده می‌کنید پس از اتصال به سرور بانک اطلاعاتی و انتخاب بانک مورد نظر، با بهره‌گیری از تابع( ) mysql _ query در خط ۱۵ اقدام به اجرای یک پرس و جوی SELECT کرده‌ایم. مرجع بازگشتی به مجموعه جواب پرس و جو در متغیری با نام $result ذخیره شده است. ما از این مرجع جهت اطلاع از تعداد سطرهای موجود در مجموعه جواب استفاده می‌کنیم.

در عبارت شرطی ساختار تکرار while در خط ۱۹ برنامه نتیجه فراخوانی تابع ( ) mysql_fetch_row را به متغیری با نام $a _ row نسبت داده‌ایم. به‌خاطر بیاورید که عملگر نسبت‌دهی (=) مقدار سمت راست را به متغیر موجود در سمت چپ نسبت می‌دهد. از این‌رو چنانچه تابع ( ) mysql_ fetch _ row مقدار عددی مثبتی را بازگرداند، عبارت شرطی مورد بحث معادل true ارزیابی خواهد شد. در داخل بدنه ساختار while آرایه موجود در $a _ row در هربار گذر از حلقه مورد پردازش قرار می‌گیرد (خط ۲۱). نتیجه حاصل عبارت، نمایش هریک از عناصر آرایه مذکور در یکی از خانه‌های جدولی است که بر روی صفحه مرورگر اینترنت به نمایش در می‌آید.

علاوه براین به یکی ازدوروش موجود می‌توان فیلدها یا مقادیر ستونهای هر سطر را با استفاده از اسامی آنها مورد دستیابی قرار داد. تابع ( ) mysql_fetch_array نیز مانند تابع ( ) mysql_fetch_ row

یک آرایه عددی باز می‌گرداند. این تابع همچنین یک آرایه انجمنی را نیز باز می‌گرداند. در چنین حالتی اسامی فیلدهای هر سطر مقادیرکلیدهای دستیابی این آرایه را تشکیل می‌دهند. قطعه کد زیر بازنویسی ساختار تکرار while برنامه لیست ۵-۱۲ است با این تفاوت که در آن به جای تابع mysql_ fetch _ row ( ) از تابع mysql_ fetch _ array( ) استفاده شده است:

```
Print " < TABLE BORDER = 1 > \ n " ;
While ($a _ row = mysql _ fetch _ array ($result) ) {
   Print " < TR > \ n " ;
   Print " < TD > ". $a _ row [ ' mail ' ] . " < / TD > < TD> " .
      $a _ row [ ' domain ' ] . "< / TD > \ n " ;
   print "< / TR > \ n " ;
}
print "< / TABLE > \ n " ;
```

بنا به پیش‌فرض، تابع mysql_ fetch _ array ( ) آرایه‌ای را باز می‌گرداند که مقادیر آنها هم توسط دنباله‌های کاراکتری و هم توسط اعداد صحیح شاخص‌گذاری شده‌اند. این شرایط درصورتی که قصد استفاده مجزا از فیلدها را داشته باشیم، مناسب است. اما درصورت دستیابی به کلیه مقادیر آرایه نیازی به چنین روش شاخص‌گذاری نمی‌باشد. تابع mysql_ fetch _ array ( ) یک آرگومان اختیاری را علاوه بر آرگومان اصلی خود به عنوان آرگومان دوم می‌پذیرد.

این آرگومان که از نوع عدد صحیح است معادل یکی از سه مقدار ثابت MYSQL _ ASSOC، MYSQL _ NUM و یا MYSQL _ BOTH می‌باشد. استفاده از مقدار ثابت MYSQL _ BOTH منجر به همان نتیجه پیش‌فرض می‌شود یعنی نتیجه‌ای که در اثر عدم استفاده از آرگومان دوم این تابع حاصل می‌آید، اما بهره‌گیری از مقدار ثابت MYSQL _ ASSOC تنها منجر به بازگشت یک آرایه شاخص‌گذاری شده با دنباله‌های کاراکتری و بهره‌گیری از مقدار ثابت MYSQL _ NUM نیز تنها منجر به بازگشت یک آرایه شاخص‌گذاری شده با اعداد صحیح خواهد شد.

در صورتی که هدف شما بازگشت یک آرایه شاخص‌گذاری شده با دنباله‌های کاراکتری باشد، جالب است بدانید که راه میانبری برای این کار در PHP4.03 پیش‌بینی شده است. فراخوانی زیر را درنظر بگیرید:

```
masql _ fetch _ array ($result, MYSQL _ ASSOC ) ;
```

نتیجه حاصل از فراخوانی فوق دقیقاً مشابه نتیجه فراخوانی تابع mysql _ fetch _ assoc ( ) با همان آرگومان $result است: Mysql_fetch_assoc ($result); علاوه بر این توابع، با استفاده از تابع mysql_ fetch_object ( ) می‌توان فیلدهای هر سطر از مجموعه جواب پرس و جو را به عنوان خصوصیات یک شئ مورد دستیابی قرار داد. در این صورت اسامی فیلدها به‌عنوان اسامی خصوصیتهای شئ موردنظر بازیابی خواهند شد. در قطعه کد زیر باردیگر ساختار تکرار while از برنامه لیست ۵- ۱۲

بازنویسی شده است با این تفاوت که این بار عملیات پردازش بدنه حلقه با استفاده از تابع مذکور انجام گرفته است:

```
Print " < TABLE BORDER = 1 > \ n " ;
While ($a _ row = mysql _ fetch _ object ($result) ) {
   Print " < tr > \ n " ;
   Print " < td > $a _ row → mail < / td > < td > $a _ row → domain < / td > \ n " ;
   print "< / tr > \ n " ;
}
print "< / table > \ n " ;
```

همان‌گونه که شاهد بودید، هر دو تابع mysql _ fetch _ array ( ) و mysql_ fetch _object ( ) روشهای‌ساده‌ای‌راجهت بازیابی اطلاعات موجود در یک سطر از مجموعه جواب پرس و جوی SELECT در اختیار برنامه‌نویس قرار می‌دهند. سرعت اجرایی هیچ یک از این دو تابع کمتر از تابع mysql _ fetch _ row ( ) نمی‌باشد. اینکه کدام‌یک از این دو را جهت استفاده انتخاب می‌کنید مطلبی است که به خود شما بستگی دارد هر چند که استفاده از تابع mysql _ fetch _ array ( ) در بین برنامه‌نویسان متداول‌تر است.

تغییر داده‌ها

به کمک یک پرس و جوی UPDATE و فراخوانی تابع mysql _ query ( ) جهت تحویل آن به سروربانک‌اطلاعاتی(به‌منظوراجرای پرس و جو) می‌توان داده‌های موجود در بانک اطلاعاتی را تغییر داد.

به خاطر داشته باشید که اجرای موفقیت‌آمیز یک پرس و جوی UPDATE لزوماً به معنی تغییر داده‌های موجود نمی‌باشد. جهت اطمینان از تغییر داده‌های یک جدول همواره می‌توان از تابع mysql_affected _ rows ( ) استفاده کرد. تابع mysql_affected _ rows ( ) به طور اختیاری آرگومانی را دریافت می‌کند که مرجع پیوند حاصل از فراخوانی یکی از دو تابع mysql _ connect ( ) یا mysql_ pconnect ( ) می‌باشد. چنانچه این آرگومان به تابع مورد بحث ارسال نشود، از مرجعی که حاصل آخرین فراخوانی یکی از دو تابع فوق است به‌عنوان آرگومان پیش‌فرض استفاده خواهد شد. تابع mysql_affected _ rows ( ) را می‌توان به‌همراه هر پرس و جویی که قادر به تغییر داده‌های بانک اطلاعاتی باشد، مورد استفاده قرار داد.

برنامه موجود در لیست ۶- ۱۲ شامل اسکریپتی است که امکان تغییر هریک از داده‌های موجود در ستون domain بانک اطلاعاتی را در اختیار کاربر قرار می‌دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 12.6 Using mysql_query()
 4: to alter rows in a database</title>
 5: </head>
 6: <body>
 7: <?php
 8: $user = "harry";
 9: $pass = "elbomonkey";
10: $db = "sample";
11: $link =  mysql_connect( "localhost", $user, $pass  );
12: if ( ! $link )
13:     die( "Couldn't connect to MySQL" );
14: mysql_select_db( $db, $link )
15:     or die ( "Couldn't open $db: ".mysql_error() );
16:
17: if ( isset( $domain ) && isset( $id ) ) {
18:     $query = "UPDATE domains SET domain = '$domain' where id=$id";
19:     $result = mysql_query( $query );
20:     if ( ! $result )
21:         die ("Couldn't update: ".mysql_error());
22:     print "<h1>Table updated ". mysql_affected_rows() .
23:     " row(s) changed</h1><p>";
24: }
25: ?>
26: <form action="<? print $PHP_SELF ?>" method="POST">
27: <select name="id">
28: <?
29: $result = mysql_query( "SELECT domain, id  FROM domains" );
30: while( $a_row = mysql_fetch_object( $result ) ) {
31:     print "<OPTION VALUE=\"$a_row->id\"";
32:     if ( isset($id) && $id == $a_row->id )
33:         print " SELECTED";
34:     print "> $a_row->domain\n";
35: }
36: mysql_close( $link );
37: ?>
38: </select>
39: <input type="text" name="domain">
40: </form>
41: </body>
42: </html>
```

لیست ۶- ۱۲ بهره‌گیری از تابع ( ) mysql _ query جهت تغییر داده‌های موجود در جدول

همان‌گونه که مشاهده می‌کنید، در این برنامه نیز مشابه برنامه‌های قبل اتصالی را با سرور بانک اطلاعاتی MySQL برقرار کرده و بانک مورد نظرمان را انتخاب نموده‌ایم. سپس در خط ۱۷ وجود مقدار در دو متغیر $domain و $id را مورد بررسی قرار دادیم. چنانچه این متغیرها حاوی مقدار باشند در خط ۱۸ پرس و جویی را از نوع UPDATE تشکیل می‌دهیم. این پرس و جو مقدار فیلد domain را هرجا که فیلد id شامل همان مقداری باشد که در متغیر $id ذخیره شده است، تغییر می‌دهد. این برنامه درصورتی که فیلد id حاوی مقدار نبوده یا مقدار متغیر $domain با مقدار فعلی فیلد domain

در سطر مورد نظر از بانک یکی باشد، پیغام خطایی را نشان نمی‌دهد، در عوض در چنین مواقعی تابع mysql_affected _ rows ( ) به‌سادگی مقدار صفر را باز می‌گرداند. مقدار بازگشتی حاصل از عملیات این تابع (که در مورد مثال ما برابر با 1 است) در خطوط ۲۲ و ۲۳ بر روی پنجره مرورگر به نمایش در می‌آید.

با شروع از خط ۲۶ یک فرم HTML ایجاد می‌شود که امکان اعمال تغییر بر روی بانک اطلاعاتی را در اختیار کاربر قرار می‌دهد. بار دیگر توجه کنید که در خط ۲۹ از تابع mysql _ query ( ) جهت بازیابی مقادیر ستونهای id و domain از جدول و بهره‌گیری از آنها در نشانه < select > از فرم HTML (خطوط ۲۷ و ۳۸) استفاده شده است. کاربر می‌تواند حوزه (domain) موردنظر خود را جهت تغییر از این لیست انتخاب کند. اگر کاربر درحال حاضر فرم را ارسال کرده باشد و شرایط به‌گونه‌ای باشد که مقدار id انتخاب شده و توسط او با مقدار فیلد id که به عنوان بخشی از خروجی نمایش داده شده است، برابر باشد؛ دنباله کاراکتری "SELECTED" به نشانهٔ OPTION در خط ۳۳ از برنامه اضافه می‌شود. بدین ترتیب مقدار تغییر داده شده بلافاصله در مقابل چشمان وی قرار می‌گیرد.

ایجاد یک کلاس مفید در رابطه با بانکهای اطلاعاتی

بدون هیچ تعارفی باید اعتراف کرد که ایجاد برنامه‌های قابل حمل و باثبات در رابطه با بانکهای اطلاعاتی امر دشواری است. برای مثال، درصورتی که کد بانک اطلاعاتی ارتباط تنگاتنگی با یک برنامه داشته باشد، بعید است بتوان آن‌را به‌سادگی از روی یک سرور به سرور دیگر منتقل کرد.

در این قسمت قصد داریم تا به منظور جداسازی کد مربوط به بانک اطلاعاتی و کد مربوط به برنامه یک کلاس مفید ایجاد کنیم (این‌گونه جداسازی‌ها در دنیای برنامه‌نویسی مرسوم بوده و به‌واسطه افزایش قابلیت حمل و مزایای دیگر با استقبال فراوانی مواجه است ـ مترجم). از دیدگاه کلی این کلاس دو ویژگی قابل توجه دارد. اول آنکه ارتباطی مابین برنامه و بانک اطلاعاتی ایجاد می‌کند که از طریق آن پرس و جوهای SQL ارسال می‌شوند و دوم اینکه روند ایجاد پرس و جوهایی را که اغلب جهت انجام‌عملیات مختلف مورد استفاده قرار می‌گیرند، مکانیزه می‌کند (مانند پرس و جوهای نوع SELECT و UPDATE). در مورد پرس و جوهای نوع SELECT مجموعه جواب به صورت یک آرایه چند بعدی تولید می‌شود، بدین ترتیب که بعد دوم آن خود یک آرایه انجمنی خواهد بود. این کلاس باید بتواند دو نوع تسهیلات ویژه را در اختیار برنامه‌نویسی که قصد استفاده از آن را دارد، قرار دهد. اولین تسهیلات به‌طور خلاصه این است که در راستای مکانیزه کردن روند ایجاد پرس و جوهای ساده باید نیاز به ساخت عبارت SQL ساده را مرتفع نماید و نکته دوم این مطلب که باید با در اختیار گذاشتن یک رابط مناسب از قابلیت حمل خوبی برخوردار باشد. چنانچه پروژه برنامه‌نویسی به‌دلایلی به یک سرور بانک

اطلاعاتی‌دیگرمنتقل شد می‌توان کلاس دیگری را که همین قابلیتها و عملکردها را در اختیار می‌گذارد، طراحی کرد اما کد مربوط به پیاده سازی آن قطعاً متفاوت خواهد بود.

بااین‌حال توجه به این نکته ضروری است که تمامی کلاسهایی که چنین امکاناتی را در اختیار برنامه‌نویسان قرار می‌دهند (مانند رابط مستقل بانک اطلاعاتی یا Database Independent Interface در زبان برنامه‌نویسی Perl که به اختصار کتابخانه DBI نامیده می‌شود)، از یک نقطه ضعف بزرگ رنج می‌برند. نکته در اینجاست که سرورهای بانک اطلاعاتی مختلف (همان شبحهای MySQL، Oracle، Sybase، Informix و غیره) معمولاً از ویژگی‌ها و دستورات زبان مختلفی برای نوشتن عبارات SQL طراحی شده برای یک بانک اطلاعاتی MySQL نمی‌تواند توسط سرور بانک اطلاعاتی دیگری مثلاً Oracle اجرا شود. در مورد پروژه‌های برنامه‌نویسی کوچک و ساده این مشکل با استفاده از دستور زبان استانداردی که توسط مؤسسه ANSI برای SQL تدوین شده است، و اجتناب از ویژگی‌هایی که هر بانک اطلاعاتی به‌طور انحصاری پشتیبانی می‌کند، قابل حل و فصل است (وجود یک نوع استاندارد مانند همیشه مسائل عدم سازگاری را حل می‌کند. امروزه بانکهای اطلاعاتی مختلف علاوه بر پشتیبانی از استانداردهای تدوین شده برای SQL ، ویژگی‌ها و امکاناتی را نیز به‌طور متنوع پشتیبانی می‌کنند. عدم اطمینان به این ویژگی‌های انحصاری و صرف استفاده از ویژگی‌های استاندارد تا اندازه زیادی به قابلیت حمل برنامه‌ها کمک می‌کند، اما عدم استفاده از ویژگی‌های انحصاری بانکهای اطلاعاتی که اغلب بسیار مفید و کارآمد هستند، مسأله دردآوری برای یک توسعه‌دهنده محسوب می‌شود ـ مترجم).

اتصال به بانک اطلاعاتی

برای شروع اجازه دهید تا متدهای لازم جهت اتصال به سرور بانک اطلاعاتی و انتخاب بانک موردنظر را ایجاد کنیم. به موازات این کار می‌توانیم متدهایی را نیز جهت گزارش خطا توسعه دهیم. لیست ۷- ۱۲ بخشی از کلاس DataLayer را که چنین کدی را شامل می‌شود، نشان می‌دهد.

```
1: <?
2: class DataLayer {
3:     var $link;
4:     var $errors = array();
5:     var $debug = false;
6:
7:     function DataLayer( ) {
8:     }
9:
10:    function connect( $host, $name, $pass, $db ) {
11:        $link =  mysql_connect( $host, $name, $pass );
12:        if ( ! $link )  {
13:            $this->setError("Couldn't connect to database server");
14:            return false;
15:        }
```

لیست ۷- ۱۲ متدهای مربوط به اتصال به سرور، انتخاب بانک اطلاعاتی و گزارش خطا

```
16:
17:             if ( ! mysql_select_db( $db, $link ) ) {
18:             $this->setError("Couldn't select database: $db");
19:             return false;
20:         }
21:         $this->link = $link;
22:         return true;
23:     }
24:
25:     function getError( ) {
26:         return $this->errors[count($this->errors)-1];
27:     }
28:
29:     function setError( $str ) {
30:         array_push( $this->errors, $str );
31:     }
32:
33:     function _query( $query ) {
34:         if ( ! $this->link ) {
35:             $this->setError("No active db connection");
36:             return false;
37:         }
38:         $result = mysql_query( $query, $this->link );
39:         if ( ! $result )
40:             $this->setError("error: ".mysql_error());
41:         return $result;
42:     }
43:
44:     function setQuery( $query ) {
45:             if (! $result = $this->_query( $query ) )
46:             return false;
47:         return mysql_affected_rows( $this->link );
48:     }
49:
50:     function getQuery( $query ) {
51:             if (! $result = $this->_query( $query ) )
52:             return false;
53:         $ret = array();
54:         while ( $row = mysql_fetch_assoc( $result ) )
55:             $ret[] = $row;
56:         return $ret;
57:     }
58:
59:     function getResource( ) {
60:         return $this->link;
61:     }
62:
63:     function select( $table,  $condition="", $sort="" ) {
64:                 $query = "SELECT * FROM $table";
65:         $query .= $this->_makeWhereList( $condition );
66:         if ( $sort != "" )
67:             $query .= " order by $sort";
68:         $this->debug( $query );
69:                 return $this->getQuery( $query, $error );
70:     }
```

دنباله لیست ۷-۱۲

```
71:
72:     function insert( $table, $add_array ) {
73:                 $add_array = $this->_quote_vals( $add_array );
74:                 $keys = "(".implode( array_keys( $add_array ), ", ").")";
75:                 $values = "values (".implode( array_values( $add_array ),
➥", ").")";
76:                 $query = "INSERT INTO $table $keys $values";
77:         $this->debug( $query );
78:                 return $this->setQuery( $query );
79:     }
80:
81:     function update( $table, $update_array, $condition="" ) {
82:         $update_pairs=array();
83:         foreach( $update_array  as $field=>$val )
84:             array_push( $update_pairs, "$field=".$this->_quote_val( $val )
➥);
85:
86:                 $query = "UPDATE $table set ";
87:         $query .= implode( ", ", $update_pairs );
88:         $query .= $this->_makeWhereList( $condition );
89:         $this->debug( $query );
90:                 return $this->setQuery( $query );
91:     }
92:
93:     function delete( $table, $condition="" ) {
94:                 $query = "DELETE FROM  $table";
95:         $query .= $this->_makeWhereList( $condition );
96:         $this->debug( $query );
97:                 return $this->setQuery( $query, $error );
98:     }
99:
100:    function debug( $msg ) {
101:        if ( $this->debug )
102:            print "$msg<br>";
103:    }
104:
105:    function _makeWhereList( $condition ) {
106:        if ( empty( $condition ) )
107:            return "";
108:        $retstr = " WHERE ";
109:        if ( is_array( $condition ) ) {
110:            $cond_pairs=array();
111:            foreach( $condition  as $field=>$val )
112:                array_push( $cond_pairs, "$field=".$this->_quote_val( $val
) );
113:            $retstr .= implode( " and ", $cond_pairs );
114:        } elseif ( is_string( $condition ) && ! empty( $condition ) )
115:            $retstr .= $condition;
116:        return $retstr;
117:    }
118:
119:    function _quote_val( $val ) {
120:        if ( is_numeric( $val ) )
121:            return $val;
```

دنباله لیست ۷-۱۲

```
122:          return "'".addslashes($val)."'";
123:      }
124:
125:      function _quote_vals( $array ) {
126:          foreach( $array as $key=>$val )
127:              $ret[$key]=$this->_quote_val( $val );
128:          return $ret;
129:      }
130: }
131: ?>
```

دنباله لیست ۷- ۱۲

همان‌گونه که مشاهده می‌کنید ما نام DataLayer را برای کلاس خود انتخاب کرده‌ایم. این کلاس شامل سه خصوصیت می‌باشد. خصوصیت $link نماینده مرجع مربوط به سرور بانک اطلاعاتی است. خصوصیت $errors شامل آرایه‌ای از پیغامهای خطا می‌باشد. و بالاخره خصوصیت $debug نشانه‌ای است که جهت مشاهده و کنترل رفتار کلاس به ما کمک خواهد کرد.

متد ( ) connect به‌سادگی با استفاده از دو تابع آشنای ( ) mysql _ connect و mysql _ ( ) select_db تسهیلات لازم جهت اتصال با سرور بانک اطلاعاتی و انتخاب بانک مورد نظر را در اختیار قرار می‌دهد. چنانچه خودتان قصد پیاده‌سازی این کلاس را دارید، شاید بخواهید آرگومانهای این توابع یعنی $host، $name، $pass و $db را به‌عنوان خصوصیات کلاس تعریف نمایید. ما در طراحی خود از این اقدام صرف‌نظر کرده‌ایم. درصورتی که با هر مشکلی در رابطه با اتصال روبرو شویم، متد( ) setError را فراخوانی می‌کنیم. این متد پشته‌ای از پیغامهای خطا را نگه می‌دارد. اگر فرآیند بدون وجود خطایی انجام شود مرجع اشاره کننده به سروربانک اطلاعاتی که توسط تابع ( ) mysql_connect به برنامه فراخواننده باز می‌گردد در خصوصیت $link از کلاس DataLayer ذخیره خواهدشد.

ساخت عبارت پرس و جو

اکنون آماده‌ایم تا متدهای مربوط به ساخت پرس و جوها را توسعه دهیم. ما این فرآیند را در قالب سه متد مختلف انجام می‌دهیم. لیست ۸- ۱۲ کد مربوط به این متدها را نشان می‌دهد.

```
1: <html>
2: <head>
3: <title>Listing 12.8 Working with the DataLayer Class</title>
4: </head>
5: <body>
6: <?php
7: include("listing12.9.php");
8: $people = array(
9:      array( 'name'=>"bob", 'age'=>20, 'description'=>"student" ),
10:     array( 'name'=>"mary", 'age'=>66, 'description'=>"librarian" ),
11:     array( 'name'=>"su", 'age'=>31, 'description'=>"producer" ),
12:     array( 'name'=>"percy", 'age'=>45, 'description'=>"civil servant" )
```

لیست ۸- ۱۲ متدهای لازم جهت ساخت عبارات پرس و جو

```
13:     );
14:
15: $dl = new DataLayer( );
16: $dl->debug=true;
17: $dl->connect( "localhost", "", "", "test" )  or die ( $dl->getError() );
18:
19: $dl->delete( "test_table" );
20:
21: foreach ( $people  as $person ) {
22:     $dl->insert( "test_table", $person ) or die( $dl->getError() );
23: }
24:
25: foreach ( $people as $person ) {
26:     $person['age']++;
27:     $dl->update( "test_table", $person, array( 'name'=>$person['name'] ) );
28: }
29:
30: $dl->delete( "test_table", "age < 25" );
31: $table = $dl->select( "test_table" );
32:
33: print "<table border=\"1\">";
34: foreach( $table as $d_row ) {
35:     print "<tr>";
36:     foreach ( $d_row as $field=>$val )
37:         print "<td>$val</td>";
38:     print "</tr>";
39: }
40: print "</table>";
41: ?>
42: </body>
43: </html>
```

دنباله لیست ۸- ۱۲

همان‌گونه که مشاهده می‌کنید متد ( ) query _ بررسی‌هایی را از برخی جهات انجام می‌دهد اما در حقیقت وظیفه اصلی آن ارسال یک رشته کاراکتری به تابع ( ) mysql _ query است که شامل یک پرس و جوی SQL می‌باشد. این متد مرجعی را به مجموعه جواب حاصل از اجرای این پرس و جو باز می‌گرداند. هر دو متد دیگر در این رابطه، یعنی ( ) getQuery و ( ) setQuery متد ( ) query _ را فراخوانی کرده و یک دنباله کاراکتری شامل پرس و جوی SQL را به‌عنوان آرگومان به آن ارسال می‌کنند. با این وجود مقدار بازگشتی از این دو تابع با یکدیگر تفاوت دارد. متد ( ) getQuery جهت کار با پرس و جوهایی طراحی شده که تغییراتی را در داده‌های بانک اطلاعاتی ایجاد می‌کنند، مانند پرس و جوی UPDATE. این متد درصورت موفقیت عدد صحیحی را باز می‌گرداند که نشان‌دهنده تعداد سطرهای دستکاری شده در جدول است. این درحالی است که متد ( ) getQuery در اصل جهت کار با پرس و جوی SELECT طراحی شده است. این متد آرایه‌ای از مجموعه جوابها را باز می‌گرداند. تا بدین‌جا کار تعریف خصوصیات و طراحی متدهای کلاس DataLayer را انجام دادیم. اکنون وقت آن است که عملکرد آن‌را مورد ارزیابی قرار دهیم.

بررسی عملکرد کلاسDataLayer را انجام دادیم. اکنون وقت آن است که عملکرد آن‌را مورد ارزیابی قرار دهیم.

بررسی عملکرد کلاس DataLayer

جهت ارزیابی این کلاس فرض می‌کنیم که جدولی با عنوان test _ table در اختیار ماست. به‌منظور تکمیل این فرض عبارت SQL مربوط به ساخت این جدول را در اینجا آورده‌ایم:

```
CREA TE TABLE test _ table (
   Id INT NOT NULL AUTO _ INCREMENT ,
   PRIMARY KEY (id) ,
   Name VARCHAR (255) ,
   Age INT ,
   Description BLOB
   ) ;
```

برنامه‌ای که این کلاس را مورد ارزیابی قرار می‌دهد ابتدا به سرور بانک اطلاعاتی متصل شده و پس از وارد کردن داده‌های نمونه و بازیابی مجدد آنها با استفاده از یک ساختار تکرار نتایج حاصل از بازیابی را در قالب یک جدول HTML بر روی صفحه مرورگر نشان می‌دهد. لیست ۹- ۱۲ این عملیات را به تصویر کشیده است.

```
$dl = new DataLayer( );
$dl->connect( "localhost", "", "", "test" )  or die ( $dl->getError() );
$dl->setQuery("DELETE FROM test_table");
$dl->setQuery("INSERT INTO test_table (name, age, description)
               VALUES('bob', 20, 'student')");
$dl->setQuery("INSERT INTO test_table (name, age, description)
               VALUES('mary', 66, 'librarian')");
$dl->setQuery("INSERT INTO test_table (name, age, description)
               VALUES('su', 31, 'producer')");
$dl->setQuery("INSERT INTO test_table (name, age, description)
               VALUES('percy', 45, 'civil servant')");
$table = $dl->getQuery("SELECT * from test_table");

print "<table border=\"1\">";
foreach( $table as $d_row ) {
        print "<tr>";
        foreach ( $d_row as $field=>$val )
                print "<td>$val</td>";
```

لیست ۹- ۱۲ ارزیابی عملکرد کلاس DataLayer

مکانیزه کردن عبارت پرس و جو

استفاده از SQL می‌تواند فرآیند بسیار پیچیده‌ای باشد و قصد ما این نیست که خود را در پیچ و خم آن سردرگم کنیم. با این حال به این نکته واقفیم که برخی از عملیات اصلی بارها و بارها تکرار می‌شوند ولی از طرف دیگر انجام تکراری آنها نیز در برنامه بسیار کسل‌کننده و مزاحم است. متدهایی که در این قسمت آنها را معرفی می‌کنیم کار ما را در انجام چنین عملیاتی ساده خواهند کرد.

اجازه دهید تا تکنیکی را بررسی کنیم که به واسطه آن می‌توانیم اجرای پرس و جوهای SELECT از بانک اطلاعاتی را مکانیزه کنیم. لیست ۱۰- ۱۲ کد لازم جهت انجام این کار را نشان می‌دهد.

```
function select( $table, $condition = "", $sort = "" ) {
      $query = "SELECT * FROM $table";
      $query .= $this -> _makeWhereList( $condition );

      if( $sort != "" )
            $query .= " ORDER BY $sort";
      $this -> debug( $query );
      return $this -> getQuery( $query, $error );
}

function _makeWhereList( $condition ) {
      if( empty( $condition ) )
            return "";

      $retstr = " WHERE ";
      if( is_array( $condition ) ) {
            $cond_pairs = array();

            foreach( $condition as $field => $val )
                  array_push( $cond_pairs, "$field = ".$this
-> _quote_val( $val ) );
            $retstr .= implode( " AND ", $cond_pairs );
      }
      elseif( is_string( $condition )&& !empty( $condition ))
            $retStr .= $condition;

      return $retstr;
}
```

لیست ۱۰- ۱۲ تسهیل اجرای پرس و جوهای SELECT

همان‌گونه که مشاهده می‌کنید متد ( ) select به‌گونه‌ای طراحی شده که جهت اجرا نام جدولی از بانک اطلاعاتی را به‌عنوان آرگومان دریافت می‌کند. آرگومانهای اختیاری این متد عبارتند از دو دنباله

کاراکتری که به ترتیب شرایط بازیابی داده‌ها از جدول و نحوه مرتب سازی داده‌های بازیابی شده را مشخص می‌کنند. آرگومان اختیاری sort برای مثال می‌تواند به صورت "age DESC, name" باشد. آرگومان اختیاری condition هم می‌تواند یک آرایه انجمنی یا یک دنباله کاراکتری باشد. درصورتی که این آرگومان یک آرایه انجمنی باشد، بخش WHERE از پرس و جوی SELECT شکل می‌گیرد. در این حالت کلیدهای دستیابی چنین آرایه‌ای معادل اسامی فیلدها یا همان ستونهای جدول فرض می‌شوند. بنابراین عبارت زیر:

Array (name ⇒ "bob" , age ⇒ 20)

به صورت زیر تعبیر می‌شود:

WHERE name = ' bob ' AND age = 20

در صورتی‌که به عبارت شرطی پیچیده‌تری مانند عبارت زیر در بخش WHERE از پرس و جوی SELECT نیاز داشته باشیم، لازم است تا آرگومان اختیاری condition را در قالب یک دنباله کاراکتری به متد ( ) select ارسال نماییم. این دنباله کاراکتری باید شامل یک عبارت SQL معتبر باشد:

WHERE name = ' bob ' AND age < 25

در چنین وضعیتی متد ( ) makeWhereList _ مسئول ساخت عبارت شرطی‌بخش WHERE از پرس و جوی SELECT خواهد بود. چنانچه پرس و جوی SELECT شامل هیچ شرطی نباشد، تابع مذکور یک دنباله کاراکتری تهی را باز می‌گرداند. اگر آرگومان مورد بحث از نوع دنباله کاراکتری باشد به‌سادگی در بخش WHERE از پرس و جو مورد استفاده قرار می‌گیرد. اما درصورتی که این آرگومان یک آرایه باشد ابتدا یک آرایه انجمنی از اسامی فیلدها و مقادیر مربوطه با نام $cond _ pairs ایجاد می‌شود. سپس با فراخوانی تابع ( ) implode آرایه جدید به دنباله کاراکتری که شامل پرس و جوی SELECT است، اضافه می‌شود. در چنین شرایطی اسامی فیلدها و مقادیر با واژه AND از یکدیگر جدا می‌شوند.

توجه کنید که هنگام ساخت دنباله کاراکتری نهایی از متد کمکی ( ) quote _ val _ استفاده کرده‌ایم. این متد به گونه‌ای طراحی شده که پیش از کاراکترهای خاص (مانند علامت کوتیشن ساده) علامت \ را درج می‌کند. همچنین دنباله کاراکتری حاصل را در داخل جفت علامت " " قرار می‌دهد (البته منهای مقادیر عددی). تعریف این متد کمکی به صورت زیر است:

```
Function _ quote _ val ($val) {
   if (is _ numeric ($val) )
      return $val ;
   return " ' ". addslashes ($val). " ' " ;
}
```

تابع ( ) addslashes یک تابع سیستمی در زبان PHP است. این تابع یک دنباله کاراکتری را به‌عنوان آرگومان پذیرفته و دنباله کاراکتری دیگری را به‌عنوان نتیجه باز می‌گرداند در حالی‌که پیش از کاراکترهای خاص در دنباله اصلی از علامت \ استفاده شده است. این تابع مفیدی برای ما محسوب

می‌شود چراکه پیش از ارسال دنباله‌های کاراکتری حاوی پرس و جو به سروربانک اطلاعاتی MySQL باید آنها را درون جفت علامت " " قرار دهیم.

اکنون‌جهت‌دستیابی‌به‌آرایه‌ای‌که شامل لیست کاملی از مقادیر موجود در جدول است، می‌توانیم از متد ( ) select به صورت زیر استفاده کنیم:

$table = $dl → select (" test _ table ") ;

همچنین برای دستیابی به سطرهایی که مقدار فیلد age در آنها کمتر از 40 است می‌توان متد ( ) select را به صورت زیر مورد استفاده قرار داد:

$table = $dl → select (" test _ table ", "age < 40") ;

و یا برای مشاهده اطلاعات همه افرادی که نام آنها bob است، می‌توان از متد ( ) select به‌صورت زیر بهره‌گرفت:

$table = $dl → select (" test _ table ", array (' name ' ⇒ "bob")) ;

ارائه کد کامل کلاس DataLayer

در لیست ۱۱- ۱۲ برنامه کامل کلاس DataLayer شامل متدهای بررسی شده در قسمتهای قبل ارائه شده است. این لیست علاوه بر متد ( ) select شامل متدهای مفید دیگری با عنوان ( )update ( ) delete و ( ) insert نیز می‌باشد. پیاده‌سازی این متدها مشابه متد ( ) select بوده و جهت اجرای پرس و جوهای متداول SQL طراحی شده‌اند.

```
class DataLayer {
      var $link;
      var $errors = array();
      var $debug = false;

      function DataLayer( ) {
      }

      function connect( $host, $name, $pass, $db ) {
            $link =  mysql_connect( $host, $name, $pass );
            if ( ! $link )  {
                  $this->setError("Couldn't connect to
database server");
                  return false;
            }

            if ( ! mysql_select_db( $db, $link ) ) {
                  $this->setError("Couldn't select database:
$db");
```

لیست ۱۱- ۱۲ برنامه کامل کلاس DataLayer

```
            return false;
        }
        $this->link = $link;
        return true;
}

function getError( ) {
        return $this->errors[count($this->errors)-1];
}

function setError( $str ) {
        array_push( $this->errors, $str );
}

function _query( $query ) {
        if ( ! $this->link ) {
            $this->setError("No active db connection");
            return false;
        }
        $result = mysql_query( $query, $this->link );
        if ( ! $result )
            $this->setError("error: ".mysql_error());
        return $result;
}

function setQuery( $query ) {
        if (! $result = $this->_query( $query ) )
            return false;
        return mysql_affected_rows( $this->link );
}

function getQuery( $query ) {
        if (! $result = $this->_query( $query ) )
            return false;
        $ret = array();
        while ( $row = mysql_fetch_assoc( $result ) )
            $ret[] = $row;
        return $ret;
}

function getResource( ) {
        return $this->link;
}

function select( $table,  $condition="", $sort="" ) {
          $query = "SELECT * FROM $table";
```

```
  $query .= $this->_makeWhereList( $condition );
            if ( $sort != "" )
                  $query .= " order by $sort";
            $this->debug( $query );
                return $this->getQuery( $query, $error );
      }

      function insert( $table, $add_array ) {
                $add_array = $this->_quote_vals( $add_array
);
                $keys = "(".implode( array_keys( $add_array
), ", ").")";
                $values = "values (".implode(.array_values(
$add_array ), ", ").")";
                $query = "INSERT INTO $table $keys $values";
            $this->debug( $query );
                return $this->setQuery( $query );
      }

      function update( $table, $update_array, $condition="" )
{
            $update_pairs=array();
            foreach( $update_array  as $field=>$val )
                  array_push( $update_pairs, "$field=".$this-
>_quote_val( $val ) );

                $query = "UPDATE $table set ";
            $query .= implode( ", ", $update_pairs );
            $query .= $this->_makeWhereList( $condition );
            $this->debug( $query );
                return $this->setQuery( $query );
      }

      function delete( $table, $condition="" ) {
                $query = "DELETE FROM  $table";
            $query .= $this->_makeWhereList( $condition );
            $this->debug( $query );
                return $this->setQuery( $query, $error );
      }

      function debug( $msg ) {
            if ( $this->debug )
                  print "$msg<br>";
      }

      function _makeWhereList( $condition ) {
```

```
            if ( empty( $condition ) )
                  return "";
            $retstr = " WHERE ";
            if ( is_array( $condition ) ) {
                  $cond_pairs=array();
                  foreach( $condition  as $field=>$val )
                        array_push( $cond_pairs,
"$field=".$this->_quote_val( $val ) );
                  $retstr .= implode( " and ", $cond_pairs );
            } elseif ( is_string( $condition ) && ! empty(
$condition ) )
                  $retstr .= $condition;
            return $retstr;
      }

      function _quote_val( $val ) {
            if ( is_numeric( $val ) )
                  return $val;
            return "'".addslashes($val)."'";
      }

      function _quote_vals( $array ) {
            foreach( $array as $key=>$val )
                  $ret[$key]=$this->_quote_val( $val );
            return $ret;
      }
}
```

دنباله لیست ۱۱-۱۲

همان‌گونه که مشاهده می‌کنید متدهای update( )، delete ( ) و insert ( ) که به ترتیب در خطوط ۸۱، ۹۳ و ۷۲ از برنامه تعریف شده‌اند، دارای عملکرد و رابط مشابهی با متد select ( ) هستند که در خط ۶۳ برنامه تعریف شده است.

تعریف متد update ( ) به‌گونه‌ای است که برای اجرا نیاز به دنباله کاراکتری دارد که نماینده یک جدول از بانک اطلاعاتی است. آرگومان ضروری دیگر این متد نام یک آرایه انجمنی متشکل از اسامی کلیدها و مقادیر مربوطه است. کلیدهای دستیابی این آرایه را اسامی فیلدهای جدول و مقادیر آن کلیدها همان مقادیر فیلدهای جدول مورد نظر می‌باشد.

آرگومان سوم متد update ( ) یک آرگومان اختیاری است. تمام شرایط و عملکرد این آرگومان مشابه آرگومان اختیاری متد select ( ) است که پیشتر آن را مورد بحث قرار دادیم. این آرگومان نیز می‌تواند یک دنباله کاراکتری یا یک آرایه باشد.

متد ( ) delete نیز که در خط ۹۳ تعریف شده است، نیازمند یک آرگومان ضروری است که از نوع دنباله کاراکتری بوده و نام جدول را مشخص می‌کند. آرگومان دوم این متد اختیاری بوده و تعیین کننده شرایط حذف داده‌ها از جدول است.

و بالاخره متد ( ) insert در خط ۷۲ برنامه نیازمند یک آرگومان از نوع دنباله کاراکتری است که نام جدول مورد نظر را مشخص می‌کند. آرگومان ضروری دیگر این متد یک آرایه انجمنی از فیلدهایی است که به جدول مورد نظر اضافه می‌شوند. کلیدهای دستیابی این آرایه را اسامی فیلدها و مقادیر مربوط به آنها را مقادیر فیلدها تعیین می‌کنند.

بهتر آن است که عملکرد این کلاس را در قالب یک برنامه مورد ارزیابی قرار دهیم. برنامه موجود در لیست ۱۲- ۱۲ شامل اسکریپتی است که داده‌هایی را در یک جدول از بانک اطلاعاتی وارد کرده و سپس عملکرد کلاس DataLayer را با فراخوانی متدهای آن مورد ارزیابی قرار می‌دهد.

```
1: <html>
2: <head>
3: <title>Listing 12.8 Working with the DataLayer Class</title>
4: </head>
5: <body>
6: <?php
7: include("listing12.9.php");
8: $people = array(
9:     array( 'name'=>"bob", 'age'=>20, 'description'=>"student" ),
10:    array( 'name'=>"mary", 'age'=>66, 'description'=>"librarian" ),
11:    array( 'name'=>"su", 'age'=>31, 'description'=>"producer" ),
12:    array( 'name'=>"percy", 'age'=>45, 'description'=>"civil servant" )
13:    );
14:
15: $dl = new DataLayer( );
16: $dl->debug=true;
17: $dl->connect( "localhost", "", "", "test" )  or die ( $dl->getError() );
18:
19: $dl->delete( "test_table" );
20:
21: foreach ( $people  as $person ) {
22:     $dl->insert( "test_table", $person ) or die( $dl->getError() );
23: }
24:
25: foreach ( $people as $person ) {
26:     $person['age']++;
27:     $dl->update( "test_table", $person, array( 'name'=>$person['name'] ) );
28: }
29:
30: $dl->delete( "test_table", "age < 25" );
31: $table = $dl->select( "test_table" );
32:
33: print "<table border=\"1\">";
34: foreach( $table as $d_row ) {
35:     print "<tr>";
36:     foreach ( $d_row as $field=>$val )
```

لیست ۱۲- ۱۲ بهره‌گیری از کلاس DataLayer در یک برنامه

```
37:         print "<td>$val</td>";
38:     print "</tr>";
39: }
40: print "</table>";
41: ?>
42: </body>
43: </html>
```

دنباله لیست ۱۲-۱۲

همان‌گونه که مشاهده می‌کنید برای شروع از خط ۸ برنامه، آرایه‌ای دوبعدی را جهت درج مقادیر آن در بانک اطلاعاتی تعریف و مقداردهی کرده‌ایم. در خط ۱۵ یک شئ از کلاس DataLayer نمونه‌گیری کرده و در خط ۱۶ خصوصیت $debug آن‌را برابر با true قرار داده‌ایم. در خط ۱۷ برنامه اتصالی را با سرور بانک اطلاعاتی MySQL ایجاد نموده‌ایم. از آنجا که شئ نمونه‌گیری درحالت debug است (خط ۱۶)، تولید و ارسال هرگونه پرس و جوی SQL به سرور بانک اطلاعاتی جهت اجرا منجر به نمایش خروجی حاصل از آن بر روی صفحه مرورگر اینترنت خواهدشد.

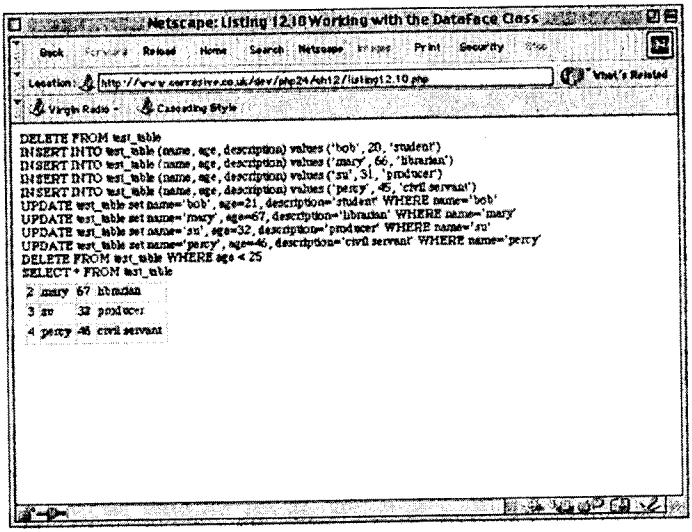
متد ( ) delete در خط ۱۹ این برنامه به‌منظور حذف هرگونه داده احتمالی در جدول test_table مورد فراخوانی قرارگرفته است. در خط ۲۱، یک حلقه تکرار برای پردازش عناصر آرایه $people تشکیل شده و در خط ۲۲ متد ( ) insert به ازای هر عنصر آرایه مذکور فراخوانی شده است. از آنجا که کلاس DataLayer جهت کار با آرایه‌های انجمنی طراحی شده است، تنها ارسال عناصر آرایه $people به متد ( ) insert به‌منظور درج داده‌ها در جدول کفایت می‌کند.

در گام بعدی سعی شده تا مقادیر موجود در فیلد age از هر یک از سطرهای جدول تغییر داده شود. باردیگر جهت پردازش عناصر آرایه $people در خط ۲۵ از برنامه حلقه تکراری تشکیل شده است. در بدنه این حلقه به ازای هریک از عناصر نامبرده متد ( ) update فراخوانی شده است اما پیش از این کار در خط ۲۴ به مقدار ذخیره شده در فیلد age یک واحد اضافه شده است. توجه کنید که می‌توانیم کل آرایه $person را جهت پردازش به متد( ) update ارسال کنیم. هرچند که این بدان معنی است که بیشتر فیلدهای موجود در سطرهای جدول با همان مقادیر اصلی خودشان به‌روز رسانی می‌شوند اما این روش از دیدگاه برنامه‌نویس سریع‌تر و آسان‌تر است. سومین آرگومان متد ( ) update آرایه‌ای انجمنی از کلیدهای دستیابی و مقادیر مربوطه است. این آرایه شرایط به‌روز رسانی را مشخص می‌کند. درحقیقت از این آرگومان در برنامه‌های واقعی هنگامی استفاده می‌شود که برنامه‌نویس قصد به‌روز رسانی تنها یک سطر از جدول را داشته باشد.

در انتها یک بار دیگر متد ( ) delete در خط ۳۰ فراخوانی شده است. همان‌گونه که مشاهده می‌کنید این‌بار در فراخوانی متد مذکور از آرگومان اختیاری نیز استفاده شده‌است. این آرگومان همان‌گونه که پیشتر توضیح دادیم بیانگر عبارت شرطی می‌باشد. از آنجا که تنها مایل به حذف سطرهایی از جدول هستیم که فیلد age در آنها از مقداری کمتر از 25 برخوردار است، لذا از یک دنباله

کاراکتری به‌جای یک آرایه به عنوان آرگومان دوم این متد بهره‌گرفته‌ایم. به‌خاطر بیاورید که استفاده از آرایه به‌عنوان آرگومان دوم متد ( ) delete تنها در شرایط بررسی تساوی کاربرد دارد. برای استفاده از علامت کوچک‌تری (<) لازم است آن‌را در قالب یک دنباله کاراکتری بیان کنیم.

خروجی حاصل از اجرای برنامه ۱۲- ۱۲ در شکل ۱- ۱۲ قابل بررسی است. از آنجا که شئ نمونه‌گیری شده از کلاس DataLayer در حالت debug است، لذا خروجی حاصل از اجرای پرس و جوهای SQL به صفحه مرورگر اینترنت ارسال می‌شود.



شکل ۱- ۱۲ نتیجه حاصل از به‌کارگیری کلاس DataLayer

جمع‌بندی

در درس این ساعت سعی ما بر این بود تا نکات اصلی مربوط به ذخیره و بازیابی اطلاعات را دررابطه با بانکهای اطلاعاتی MySQL مورد بررسی قرار دهیم.

اکنون‌بااین‌بررسی باید بتوانید با استفاده از تابع ( ) mysql _ connect یا ( )mysql _ pconnect اتصالی را با سرور بانک اطلاعاتی MySQL برقرار نمایید.

همچنین بانک اطلاعاتی مورد نظرتان را با استفاده از تابع ( ) mysql _ select _ db انتخاب کرده و در صورت عدم موفقیت آمیز این فرآیند باید بتوانید علت شکست فرآیند را با بهره‌گیری از تابع ( ) mysql _ error کشف نمایید.

علاوه بر این هم‌اکنون باید قادر به ایجاد پرس و جوهای SQL و ارسال آنها به سروربانک اطلاعاتی از طریق به کارگیری تابع ( ) mysql _ query باشید. به کمک مرجعی که این تابع به مجموعه جواب حاصل از پرس و جوی SQL باز می‌گرداند، باید قادر به دستیابی به داده‌ها و تشخیص

تعداد سطرهای موجود در مجموعه جواب باشید.

با دانشی که اکنون از توابع مخصوص بانک اطلاعاتی MySQL در زبان PHP به دست آورده‌اید، باید قادر به تشخیص تعداد بانکهای اطلاعاتی، جداول و فیلدهای قابل دستیابی در هر جدول از بانک اطلاعاتی بوده و بتوانید اطلاعات بیشتری درمورد ویژگی‌ها و خصوصیات هر فیلد به طور مجزا به دست آورید.

و بالاخره با مطالبی که در مورد کلاسها و بانکهای اطلاعاتی فراگرفتید، باید بتوانید برخی از تکنیکهای بررسی شده در این درس در مورد کلاس DataLayer را جهت مکانیزه کردن پرس و جوهای SQL و جداکردن کد مربوط به کار با بانک اطلاعاتی از کد برنامه اصلی مورد استفاده و بهره‌گیری قرار دهید.

در درس ساعت آینده ویژگی‌ها و قابلیتهای زبان PHP را از جنبه‌ای دیگر مورد بررسی قرار می‌دهیم. به‌طور دقیق‌تر، به‌زودی با تکنیک‌هایی آشنا می‌شوید که امکان برقراری ارتباط با سایر ماشینها یا کامپیوترها و کسب اطلاعات مورد نیاز در مورد آنها را در اختیارتان قرار می‌دهند.

پرسش و پاسخ

پرسش: درس این ساعت به مسائل مربوط به بانک اطلاعاتی MySQL اختصاص داشت. آیا تکنیکهای بحث شده در این درس را می‌توان در مورد سایر بانکهای اطلاعاتی SQL نیز مورد استفاده قرار داد؟

پاسخ: به طور عام در رابطه با SQL توابعی وجود دارد که از لحاظ عملکرد و بهره‌گیری تقریباً مشابه توابعی است که در مورد بانک اطلاعاتی MySQL به طور خاص طراحی شده است. علاوه بر این متناسب‌با دیگر سرورهای بانک اطلاعاتی توابعی در زبان PHP پیش بینی شده که ویژگی‌ها و قابلیتهای هریک را مورد پشتیبانی قرار می‌دهند. به طور خلاصه، ویژگی مشترک بسیاری از این مجموعه توابع توانایی ارسال پرس و جوهای SQL توسط آنهاست. با این حال اگر بتوانید از دستور زبان استانداردی که توسط مؤسسه ANSI در مورد زبان SQL تدوین شده است، استفاده کنید؛ با دردسرهای کمتری در مورد قابلیت حمل برنامه‌ها بر روی سرورهای مختلف بانک اطلاعاتی مواجه خواهید شد.

پرسش: بهترین روش جهت نوشتن برنامه‌هایی که بتوان به آسانی آنها را با سرورهای مختلف بانک اطلاعاتی مورد استفاده قرار داد، کدام است؟

پاسخ: ایده خوب در این رابطه این است که کدهای مربوط به پرس و جو از بانک اطلاعاتی را در یک کلاس یا کتابخانه واحد جمع کنیم. اگر در این حالت لازم باشد تا برنامه‌های خود را جهت کار با بانکهای اطلاعاتی مختلف دوباره نویسی کنید، کافی است تا بدون اعمال تغییر کلی و کلان در کل

پروژه، تغییرات مورد نظر را بر روی بخشهای مجزایی از آن اعمال کنید. با این وجود این نکته را همواره به خاطر داشته باشید که تفاوتهایی در SQL مربوط به بانکهای اطلاعاتی مختلف وجود دارد.

تمرینها

هدف از این بخش دوره مطالب فراگیری شده در قالب آزمون است. پاسخ بخش آزمون بلافاصله بعد از آن آمده است. بخش فعالیتها شامل تمرینهایی است که به منظور افزایش مهارت برنامه نویسی خواننده طراحی شده و البته فاقد پاسخ است.

آزمون

۱- چگونه می‌توان اتصالی را با یک سرور بانک اطلاعاتی MySQL برقرار کرد؟
۲- از کدام تابع می‌توان جهت انتخاب بانک اطلاعاتی استفاده کرد؟
۳- ازکدام‌تابع‌می‌توان‌جهت ارسال یک پرس و جوی SQL به سرور بانک اطلاعاتی استفاده نمود؟
۴- عملکرد تابع ( ) mysql _ insert _ id چیست؟
۵- با فرض ارسال موفقیت آمیز یک پرس و جوی SELECT به سروربانک اطلاعاتی MySQL جهت اجرا، نام سه تابع را که می‌توان از آنها برای دستیابی به سطرهای مجموعه جواب استفاده کرد، ذکر نمایید.
۶- با فرض ارسال موفقیت آمیز یک پرس و جوی UPDATE به سروربانک اطلاعاتی MySQL جهت اجرا، از چه تابعی می‌توان برای تشخیص تعداد سطرهای متأثر از پرس و جوی فوق استفاده کرد؟
۷- به‌منظورتعیین‌لیست‌تمامی بانکهای اطلاعاتی قابل انتخاب پس از برقراری اتصال موفقیت‌آمیز با سرور بانک اطلاعاتی MySQL از چه تابعی می‌توان استفاده کرد؟
۸- ازچه‌تابعی‌می‌توان جهت اطلاع از لیست تمامی جداول موجود در یک بانک اطلاعاتی استفاده نمود؟

پاسخ آزمون

۱- با استفاده از تابع ( ) mysql _ connect می‌توان اتصالی مابین برنامه و سرور بانک اطلاعاتی MySQL برقرار کرد.
۲- با استفاده از تابع ( ) mysql _ select _ db می‌توان بانک اطلاعاتی مورد نظر را البته پس از برقراری یک اتصال موفقیت‌آمیز با سرور بانک اطلاعاتی MySQL انتخاب نمود.

۳- با بهره‌گیری از تابع ( ) mysql _ query می‌توان پرس و جوی SQL مورد نظر را جهت اجرا به سرور بانک اطلاعاتی MySQL ارسال کرد.

۴- تابع ( ) mysql _ insert _ id مقدار فیلدی از جدول را که به محض ورود سطر جدیدی از اطلاعات به میزان یک واحد افزایش می‌یابد، باز می‌گرداند.

۵- برای دستیابی به هریک از سطرهای مجموعه جواب یک پرس و جوی SELECT ، می‌توان هر یک از سه تابع ( ) mysql _ fetch _ rows، ( ) mysql _ fetch _ array یا mysql _ fetch _ object ( ) استفاده کرد.

۶- با استفاده از تابع ( ) mysql _ affected _ row، می‌توان تعداد سطرهای متاثر از اجرای یک پرس و جوی UPDATE را مشخص نمود.

۷- تابع ( ) mysql _ list _ dbs مرجعی را باز می‌گرداند که با استفاده از آن می‌توان به لیست بانکهای اطلاعاتی موجود و قابل انتخاب دسترسی داشت.

۸- تابع ( ) mysql _ list _ tables مرجعی را باز می‌گرداند که با استفاده از آن می‌توان به لیست جداول موجود در یک بانک اطلاعاتی دسترسی داشت.

فعالیتها

۱- یک بانک اطلاعاتی از نوع MySQL ایجاد کنید که شامل سه فیلد email (حداکثر ۷۰ کاراکتری)، message (حداکثر ۲۵۰ کاراکتری) و date (یک عدد صحیح که بیانگر زمان در قالب UNIX است) باشد. برنامه‌ای بنویسید که امکان وارد کردن داده‌ها را از طریق یک فرم ورودی در اختیار کاربر قرار دهد.

۲- برنامه‌ای بنویسید که کلیه اطلاعات موجود در بانک اطلاعاتی ایجاد شده در تمرین قبل را در اختیار کاربر قرار دهد.

ساعت سیزدهم

بررسی عملیات سمت سرور

در درس این ساعت به بررسی برخی از توابعی می‌پردازیم که امکانات مفیدی را در برقراری ارتباط با دنیای خارج از کامپیوتر کلاینت در اختیارمان قرار می‌دهند.

مطالب زیر را در درس این ساعت مورد بررسی قرار خواهیم داد:

- **بررسی بیشتر در مورد متغیرهای سیستمی**
- **کالبد شناسی یک اتصال HTTP**
- **چگونگی دستیابی به اسناد موجود بر روی یک کامپیوتر سرور راه دور**
- **چگونگی برقراری اتصال HTTP**
- **چگونگی اتصال به سایر سرویسهای شبکه**
- **چگونگی ارسال email از طریق برنامه‌های اسکریپت**

در ادامه به بررسی موارد فوق می‌پردازیم.

متغیرهای سرور

تا بدین جای بحث خود در مورد PHP با برخی از متغیرهای سیستمی آشنا شدید که PHP و همچنین وب سرور آنها را در اختیارتان قرار می‌دهند، در این قسمت قصد داریم تا متغیرهایی را که وب سرور در اختیار ما به‌عنوان برنامه‌نویس قرار می‌دهد، با شرح مفصل‌تری نسبت به قبل بررسی کنیم. این متغیرها در اصل توسط وب سرور در اختیار PHP قرار می‌گیرند و از این طریق برنامه‌نویس می‌تواند آنها را مورد بهره‌برداری قرار دهد. اگر از وب سرور Apache استفاده می‌کنید، به احتمال قوی تمامی متغیرهای سیستمی مورد بحث در درس این ساعت را در اختیار خواهید داشت. در صورت استفاده از یک وب سرور دیگر، برای مثال وب سرور مایکروسافت با نام Internet Information server یا به اختصار IIS، هیچ تضمینی در مورد دستیابی به این متغیرها وجود ندارد. از این‌رو همواره بهتر است پیش از به‌کارگیری آنها از وجودشان اطمینان حاصل کنید. جدول ۱- ۱۳، لیست برخی از متغیرهای سیستمی را که در صورت دستیابی و استفاده از آنها می‌توانید اطلاعات مفیدی در مورد بازدیدکنندگان اینترنتی از وب سایت به‌دست آورید، نشان داده است.

جدول ۱- ۱۳ برخی از متغیرهای سیستمی مفید

نام متغیر	توضیح
$HTTP _ REFERER	یک آدرس URL که برنامه جاری از آن طریق مورد دستیابی و فراخوانی قرارگرفته است (املای نادرست REFERER در این میان قابل توجه است).
$HTTP _ USER _ AGENT	اطلاعاتی در مورد مرورگر اینترنت و محیط عامل مورد استفاده بازدیدکننده وب سایت
$REMOTE _ ADDR	آدرس IP بازدیدکننده وب سایت
$REMOTE _ HOST	نام کامپیوتر میزبان بازدیدکننده وب سایت
$QUERY _ STRING	یک دنباله کاراکتری رمزگذاری شده که ممکن است به انتهای آدرس URL ضمیمه شود (به‌صورت کلی ?akey = avalue&anotherkey =anothervalue) کلیدها و مقادیرآنها از طریق متغیرهای سراسری در اختیار برنامه قرار می‌گیرند.
$PATH _ INFO	سایر اطلاعاتی که ممکن است به آدرس URL ضمیمه شود.

برنامه موجود در لیست ۱- ۱۳ قادر است تا محتوای این متغیرها را به‌عنوان خروجی به مرورگر اینترنت ارسال کند.

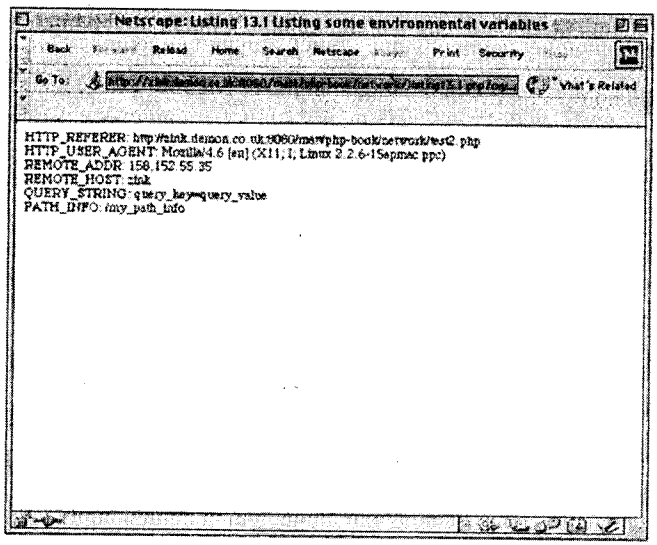
```
 1: <html>
 2: <head>
 3: <title>Listing 13.1 Listing some server variables</title>
 4: </head>
 5: <body>
 6: <?php
 7: $envs = array(  "HTTP_REFERER", "HTTP_USER_AGENT", "REMOTE_ADDR",
 8:         "REMOTE_HOST", "QUERY_STRING", "PATH_INFO" );
 9: foreach ( $envs as $env )
10:     print "$env: $GLOBALS[$env]<br>";
11: ?>
12: </body>
13: </html>
```

لیست ۱- ۱۳ نمایش محتوای برخی از متغیرهای سرور

خروجی حاصل از این برنامه در شکل ۱- ۱۳ قابل بررسی است. خروجی را که در این شکل ملاحظه می‌کنید، به‌واسطه فراخوانی برنامه از طریق کلیک بر روی پیوندی از صفحه حاصل آمده است (پیوند یا link، عنصری از یک صفحه وب است که امکان برقراری پیوند با سایر صفحات را در اختیار کاربران قرار می‌دهد ـ مترجم). پیوندی که این برنامه را فرا می‌خواند، می‌تواند به شکل زیر باشد (با فرض اینکه نام برنامه Listing 13.1. php باشد) :

< A HREF = ' Listing 13.1. php / my _ path _ info? query _ key =
query _ value ' > go < / A >

همان‌گونه که مشاهده می‌کنید در این پیوند از آدرس‌دهی نسبی جهت فراخوانی برنامه Listing13.1.php استفاده شده است.



شکل ۱- ۱۳ نمایش مقادیر برخی از متغیرهای سرور

اطلاعات اضافی در مورد مسیر (my _ path _ info) همان‌گونه که مشاهده می‌کنید بعد از نام

سند واقع شده و از طریق متغیر سیستمی $PATH _ INFO در اختیار برنامه قرار گرفته است.

با اندکی دقت در می‌یابید که در این پیوند از دنباله پرس و جوی query _ key = query _ value استفاده شده که آن‌هم از طریق متغیر سیستمی $QUERY _ STRING در اختیار برنامه قرارگرفته است. معمولاً در مورد فرمهایی که از آرگومان GET جهت ارسال اطلاعات به سرور استفاده می‌کنند، با دنباله‌های پرس و جو مواجه می‌شویم. با این حال درصورت تمایل می‌توانیم هنگام ارسال اطلاعات از یک سند وب به دیگری نیز از این دنباله‌ها استفاده نماییم.

دنباله پرس و جو ترکیبی از اسامی کلیدی و مقادیر مربوط به آنهاست که توسط علامت & از یکدیگر جدا شده‌اند. معمولاً چنین عنوان می‌شود که دنباله‌های پرس و جو رمزگذاری شده‌اند. این گفته بدان معنی است که هر کاراکتر غیر مجاز در آدرس‌دهی URL یا هر کاراکتری که معنای خاصی در این نوع آدرس‌دهی داشته باشد به معادل خود که عددی در مبنای شانزده است، تبدیل می‌شود. با وجودی که دستیابی به کل دنباله پرس و جو از طریق متغییرسیستمی $QUERY _ STRING امکان‌پذیر است اما در بیشتر موارد کل دنباله پرس و جو مدنظر نمی‌باشد. نام هر کلید از دنباله پرس و جو از طریق یک متغیر سراسری (در مورد مثال ما $query _ value) در دسترس برنامه‌نویس قرار می‌گیرد که به نوبه خود مقدار رمزگشایی شده مربوطه (در این موردquery _ value) را در خود ذخیره می‌کند. علاوه بر این با استفاده از یک آرایه انجمنی با نام $HTTP _ GET _ VARS ، می‌توانیم به کلیدهای دستیابی و مقادیر مربوط به آنها از دنباله پرس و جو دسترسی پیدا کنیم.

متغیر سیستمی $HTTP _ REFERER در مواقعی مفید است که بخواهیم موقعیت سندی را که موجب فراخوانی برنامه شده است، تشخیص دهیم. با این همه باید دقت کنید که این متغیر و سایر متغیرهای محیطی (سیستمی) به‌راحتی قابل تحریف هستند. نحوه انجام این کار را به‌زودی در همین درس فرا می‌گیرید. به املای نادرست REFERER در این متغیر سیستمی توجه کنید. املای صحیح آن‌به‌صورت REFERRER است، اما در صورت تصحیح آن به شکل فوق با مشکلات ناشی از عدم سازگاری مواجه خواهید شد. از آنجا که همه مرورگرهای اینترنت از متغیر سیستمی $HTTP_REFERER پشتیبانی به عمل نمی‌آورند، لذا بهتر است از اعتماد کردن به آن خودداری کنید.

با تحلیل مقدار متغیر سیستمی $HTTP _ USER _ AGENT می‌توانید به اطلاعات جالب توجهی در مورد نوع مرورگر اینترنت و سیستم عامل مورد استفاده کاربر دست پیدا کنید اما بار دیگر یادآوری می‌کنیم که این متغیر نیز قابل تحریف و بنابراین غیر قابل اعتماد است. برنامه‌نویسان معمولاً از این متغیر جهت ارائه کدهای HTML و JavaScript متفاوتی بسته به نوع مرورگر اینترنت و شماره نسخه آن استفاده می‌کنند. در درس ساعت هفدهم با عنوان "بهره‌گیری از دنباله‌های کاراکتری" و

درس ساعت هجدهم با عنوان "بهره‌گیری از عبارات منظم" با ابزارهای لازم برای استخراج اطلاعات مورد نیاز از درون دنباله‌های کاراکتری آشنا می‌شوید.

متغیر سیستمی $REMOTE _ ADDR ، شامل آدرس IP کاربر سایت بوده و با استفاده از آن می‌توان کاربران مختلف را شناسایی کرد. (آدرسهای IP منحصر به فرد هستند) دقت کنید که تمامی کاربران اینترنت از آدرس IP ثابتی استفاده نمی‌کنند؛ بدین معنی که ISP مورد استفاده آنها در هر تماس جهت برقراری ارتباط با اینترنت یک آدرس IP را به آنها اختصاص می‌دهند که ممکن است با آدرسهای قبلی متفاوت باشد. به بیان دیگر، یک آدرس IP خاص ممکن است توسط کاربران مختلفی از وب سایت شما مورد استفاده قرار بگیرد. همچنین یک کاربر خاص ممکن است در دفعات مختلف از آدرسهای IP متفاوتی استفاده نماید.

بسته به نوع پیکربندی و تنظیمات وب سرور متغیر $REMOTE _ HOST ، ممکن است در دسترس قرار داشته باشد. درصورتی که این متغیر در دسترس باشد، حاوی نام میزبان کاربر خواهد بود. از آنجا که وجود این متغیر مستلزم سرباری است که به ازای هر درخواست دریافتی توسط سرور به آن تحمیل می‌شود (جهت یافتن نام میزبان)، معمولاً ترتیبی داده می‌شود تا این متغیر غیرفعال و غیر قابل دسترس باشد. اگر به این متغیر دسترسی ندارید، با استفاده از متغیر سیستمی دیگری با نام $REMOTE _ ADDR که در پاراگراف قبل شرح دادیم، می‌توانید به اطلاعات لازم دست پیداکنید. چگونگی انجام این کار را به‌زودی در درس همین ساعت فراخواهید گرفت.

مقدمه‌ای بر HTTP و گفتگوی اینترنتی بر پایه شیوه client / server

بررسی تمام اطلاعاتی که هنگام درخواست یک سند مابین کامپیوتر کاربر (به‌طور دقیق‌تر مرورگر اینترنت مورد استفاده کاربر) و کامپیوتر سرور (باز هم به‌طور دقیق‌تر برنامه وب سرور موجود بر روی کامپیوتر سرویس دهنده) رد و بدل می‌شوند، خارج از حوصله این کتاب بوده و شاید ارزش فوق‌العاده‌ای هم نداشته باشد، چراکه PHP خود بیشتر جزئیات این فرآیند را هدایت و کنترل می‌کند. اما اطلاع از بخشهای اساسی این فرآیند می‌تواند مفید باشد به‌ویژه اگر برنامه شما قصد دستیابی به اسناد وب موجود بر روی سرور یا بررسی وضعیت آدرسهای وب را داشته باشد.

واژه HTTP کوتاه شده عبارت hypertext transfer protocol و به معنای قرارداد انتقال اطلاعات فرامتنی است. HTTP در اصل مجموعه‌ای از قوانینی است که فرآیند ارسال درخواست از Client به Server و ارسال پاسخ از Server به Client را تشریح می‌کند. هم Client و هم Sever در این فرآیند اطلاعاتی را در مورد خود و داده‌هایی که ارسال می‌کنند، در اختیار یکدیگر قرار می‌دهند. بیشتر این اطلاعات از طریق متغیرهای سیستمی در اختیار ما قرار دارد.

فرآیند درخواست

فرآیند درخواست اطلاعات (در قالب اسناد اینترنتی) از Server توسط Client و بر مبنای یک مجموعه قوانین سفت و سخت انجام می‌شود. هر درخواستی حداکثر از سه بخش تشکیل می‌شود:

- خط درخواست
- بخش هدر
- بدنه درخواست

خط درخواست بخش ضروری هر درخواستی را تشکیل می‌دهد. این بخش از درخواست نحوه درخواست را که یکی از انواع HEAD, GET یا POST است، مشخص می‌کند. سایر عناصر (اطلاعات) این بخش عبارتند از آدرس سند درخواستی و نسخه‌ای از قرارداد HTTP که جهت انجام فرآیند مورد استفاده Client قرار گرفته است (در این رابطه تنها دو نسخه HTTP / 1.0 و HTTP / 1.1 موجود است). یک خط درخواست نمونه جهت دستیابی به سندی با نام mydoc. Html می‌تواند به صورت زیر باشد:

```
GET / mydoc. html HTTP / 1.0
```

همان‌گونه که مشاهده می‌کنید، در نمونه فوق برنامه Client درخواستی از نوع GET را مشخص کرده است.

به‌عبارت دیگر، کل یک سند مورد درخواست قرارگرفته است اما در عین حال هیچ داده‌ای به سمت سرور گسیل نشده است (البته شاید ذکر این نکته جالب باشد که به‌عنوان بخشی از یک درخواست GET می‌توان داده‌هایی را نیز در قالب دنباله پرس و جویی که به انتهای آدرس URL ضمیمه می‌شود، برای سرور ارسال کرد). از روش درخواست HEAD تنها هنگامی استفاده می‌شود که صرفاً اطلاعاتی در مورد یک سند مورد نیاز باشد. روش POST نیز برای ارسال داده‌ها از Client به Server مورد استفاده قرار می‌گیرد. از این روش معمولاً هنگامی استفاده می‌شود که ارسال اطلاعات یک فرم HTML به Server مدنظر باشد.

همان‌گونه که مشاهده نمودید، خط درخواست خود به تنهایی جهت تنظیم یک درخواست از نوع GET کفایت می‌کند. اما توجه کنید که جهت اطلاع Server از وجود چنین درخواستی وجود یک خط خالی بعد از خط درخواست الزامی است. بیشتر برنامه‌های Client بعد از خط درخواست بخشی را به‌عنوان هدر که شامل مجموعه‌ای از اسامی و مقادیر مربوط به آنهاست به آن ضمیمه می‌کنند. برخی از این اسامی از طریق متغیرهای سیستمی در دسترس ما می‌باشند. بخش هدر ترکیبی از یک کلید (یک نام منحصر بفرد) و مقدار آن است که توسط علامت کولون (:) از یکدیگر جدا شده‌اند. جدول ۲-۱۳ لیست برخی از این اسامی را نمایش می‌دهد.

جدول ۲- ۱۳ اسامی برخی از هدرهای Client

نام هدر	توضیح
Accept	نوع رسانه‌هایی که Client قادر به کار با آنهاست
Accept _ Encoding	انواع روشهای موجود برای فشرده سازی اطلاعات که Client قادر به پشتیبانی از آنهاست.
Accept _ charset	مجموعه‌های کاراکتری که Client پشتیبانی می‌کند.
Accept _ Language	زبان ترجیحی Client (زبان پیش‌فرض انگلیسی است)
Host	نام میزبانی که Client اطلاعات مورد نیازش را از آنجا درخواست کرده است (برخی از سرورهایی که نقش چندین میزبان مجازی را بازی می‌کنند از این هدر ویژه بهره زیادی می‌برند)
Referer	سندی که درخواست از آنجا صادر شده است
User _ Agent	اطلاعاتی درمورد نوع و شماره نسخه برنامهClient

در مورد روشهای HEAD و GET بخش هدر به عنوان نقطه پایان درخواست تلقی شده و بعد از آن یک خط خالی توسط برنامه Client جهت اطلاع Server از تکمیل درخواست درج می‌شود. برای درخواستهای نوع POST پس از این خط خالی بدنه درخواست موردنظر ذکر می‌گردد. بدنه درخواست شامل هرگونه داده‌ای است که به Server ارسال می‌شود. معمولاً این بخش شامل اسامی کلیدی و مقادیر مربوطه است، که مشابه دنباله‌های پرس و جو رمزگذاری شده‌اند.

برنامه موجود در لیست ۲- ۱۳ نمونه یک درخواست ارسالی توسط برنامه Netscape 4.6 را که عهده‌دار نقش Client است، نشان می‌دهد.

```
1: GET / HTTP/1.0
2: Connection: Keep-Alive
3: User-Agent: Mozilla/4.51 (Macintosh; I; PPC)
4: Host: www.corrosive.co.uk:8080
5: Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
6: Accept-Encoding: gzip
7: Accept-Language: en
8: Accept-Charset: iso-8859-1,*,utf-8
```

لیست ۲- ۱۳ نمونه‌ای از یک درخواست که تنها شامل خط درخواست و بخش هدر بوده و توسط برنامه Netscape 4.6 تنظیم شده است.

فرآیند پاسخ

پس از دریافت درخواست Client توسط برنامه Server این برنامه پاسخی را به برنامهClient

ارسال می‌کند. هرپاسخی که Server ارسال می‌کند از سه بخش زیر تشکیل می‌گردد:

- خط وضعیت
- بخش هِدر
- بدنه پاسخ

همان‌گونه که احتمالاً متوجه شدید، تقارن زیادی بین درخواست Client و Server وجود دارد. درحقیقت برخی از هِدرها هم از سمت Client و هم از سمت Server به سوی دیگر ارسال می‌شوند. این وضعیت به‌ویژه در مورد هدرهایی صدق می‌کند که شامل اطلاعاتی درباره بدنه درخواست یا پاسخ هستند.

خط وضعیت شامل سه نوع اطلاعات است: اول نسخه قرار داد HTTP مورد استفاده Server (که یکی از دو نسخه موجود HTTP / 1.0 یا HTTP / 1.1 است)، دوم کدی که نشان دهنده پاسخ است و سوم متنی است که کد مذکور را توصیف می‌کند.

برنامه Server کدهای پاسخ بسیار متنوعی را می‌تواند در پاسخ یک درخواست Client ارسال کند. هریک از این کدها شامل اطلاعات مفیدی درباره موفقیت یا عدم موفقیت‌آمیز بودن درخواست Client می‌باشند. جدول ۳- ۱۳ لیست برخی از کدهای متداول در این زمینه را نشان می‌دهد.

جدول ۳- ۱۳ برخی از کدهای متداول در رابطه با پاسخ Server

شماره کد	متن معادل	توضیح
200	OK	درخواست موفقیت‌آمیز بوده و سند مورد درخواست Client برای وی ارسال می‌گردد.
301	Moved Permanently	اطلاعات درخواستی دیگر بر روی Server موجود نمی‌باشد. بخش‌هدر پاسخ آدرس جدید اطلاعات مورد نظر را مشخص می‌کند.
302	Moved Temporarily	اطلاعات درخواستی به موقعیت دیگری انتقال یافته است. بخش هدر پاسخ آدرس جدید اطلاعات موردنظر را مشخص می‌کند.
404	Not Found	اطلاعات‌درخواستی‌در موقعیت مشخص شده توسط Client موجود نمی‌باشد.
500	Internal Server Error	برنامه Server یا برنامه CGI به هنگام دستیابی به داده‌ها با مشکل مواجه شده است.

با این توصیف خط پاسخ مربوط به یک درخواست نمونه ممکن است به شکل زیر باشد:
HTTp/1.0 zoook

بخش هدر هر پاسخی شامل هدرهای مختلفی است که قالبی مشابه قالب هدرهای ارسالی از Client دارند. جدول ۴- ۱۳ برخی از هدرهای متداول ارسالی توسط Server را نشان می‌دهد.

جدول ۴- ۱۳ برخی از هدرهای متداولی که برنامه Server ارسال می‌کند

نام هدر	توضیح
Date	تاریخ جاری
Server	نام و شماره نسخه برنامه Server
Content - Type	نوع داده ارسالی توسط Server (یکی از انواع داده‌های تعریف شده در MIME)
Content - Size	اندازه محتوای ارسالی به Client برحسب بایت
Location	آدرس کامل سند

برنامه موجود در لیست ۳-۱۳، یک پاسخ ارسالی نمونه توسط Server را نشان می‌دهد. همان‌گونه که مشاهده می‌کنید، پس از ارسال هدرها (خطوط ۲ تا ۹) برنامه Server اقدام به ارسال یک خط خالی (خط ۱۰) و به‌دنبال آن ارسال بدنه پاسخ می‌کند (یعنی همان سند درخواستی توسط برنامه Client).

```
 1: HTTP/1.1 200 OK
 2: Date: Tue, 25 Sep 2001 16:10:06 GMT
 3: Server: Apache/1.3.12 Cobalt (Unix) PHP/4.0.6 mod_perl/1.24
 4: Last-Modified: Tue, 25 Sep 2001 16:08:29 GMT
 5: ETag: "147829-62-3bb0abfd"
 6: Accept-Ranges: bytes
 7: Content-Length: 98
 8: Connection: close
 9: Content-Type: text/html
10:
11: <html>
12: <head>
13: <title>Listing 13.3 A server response</title>
14: </head>
15: <body>
16: Hello
17: </body>
18: </html>
```

لیست ۳- ۱۳ یک پاسخ نمونه ارسالی توسط برنامه Server

دستیابی به سندی از طریق یک آدرس راه دور

با وجودی که PHP یک زبان برنامه‌نویسی برای سمت سرور است، اما خود به مانند یک برنامه Client می‌تواند اطلاعات مورد نیاز را از یک سرور راه دور مورد درخواست قرار داده و پس از دریافت، آنها را در اختیار سایر برنامه‌ها قرار دهد. PHP به‌همان راحتی که امکان خواندن فایل‌های موجود بر روی سرور را دراختیار می‌گذارد، امکانات مشابهی را نیز جهت خواندن فایل‌های موجود در هر موقعیتی از وب در اختیار قرار می‌دهد. درواقع نحوه انجام این دو کار شباهت بسیاری به هم دارد. با بهره‌گیری از تابع ( ) Fopen ، می‌توانید همانند ارتباط با فایل‌های موجود بر روی سرور، با هر آدرسی بر روی وب ارتباط برقرار کنید. برنامه موجود در لیست ۴-۱۳، روند اتصال به یک سرور راه دور و درخواست سندی از آن را با استفاده از این تابع نشان می‌دهد. برنامه فوق پس از دریافت سند محتوای آن را بر روی صفحه مرورگر نمایش می‌دهد.

```
1: <html>
2: <head>
3: <title>Listing 13.4 Getting and printing a web page with fopen()</title>
4: </head>
5: <body>
6: <?php
7: $webpage = "http://www.corrosive.co.uk/php/hello.html";
8: $fp = fopen( $webpage, "r" ) or die("couldn't open $webpage");
9: while ( ! feof( $fp ))
10:     print fgets( $fp, 1024 );
11: ?>
12: </body>
13: </html>
```

لیست ۴- ۱۳ دستیابی به یک سند وب از طریق تابع ( ) fopen

جهت اجرای موفقیت آمیز این برنامه مطمئن شوید که گزینه allow _ url _ fopen از فایل php. ini با مقدار on تنظیم شده است. البته این تنظیم به‌طور پیش‌فرض توسط PHP انجام شده است.

معمولاً بعید است که قصد نمایش کل یک سند را بر روی صفحه مرورگر داشته باشیم. به‌عبارت دیگر، معمولاً تنها به بخشهایی از یک سند علاقمندیم. از این‌رو اغلب صفحه یا سند دریافتی از وب را به منظور تعیین بخشهای مورد نیاز پردازش می‌کنیم.

تابع ( ) fopen در نسخه‌های پیش از PHP4.0.5 مستقیماً قرارداد HTTP را مورد پشتیبانی قرار نمی‌داد. در نسخه‌های پیشین هنگامی که مرورگر کد پاسخ 301 یا 302 را دریافت می‌کرد، اقدام به تنظیم و ارسال درخواست دیگری برمبنای محتوای هدر Location می‌نمود. اکنون در نسخه‌های جدید تابع ( ) fopen خود این ویژگی را پشتیبانی می‌کند و این بدان معنی است که آن دسته از آدرسهای URL که فهرستها را مورد اشاره قرار می‌دهند نیازی به استفاده از علامت / در انتهای آدرس نخواهند داشت.

چنانچه اتصال به سرور راه دور با موفقیت انجام شود، تابع ( ) fopen مرجعی را به فایل مورد نظر باز می‌گرداند. درصورتی که فرآیند اتصال با شکست مواجه شده یا سند مورد درخواست موجود نباشد، این تابع مقدار false را به برنامه فراخواننده خود باز می‌گرداند. با در دست داشتن مرجع فایل بازگشتی از این تابع می‌توان از آن جهت خواندن فایل مربوطه استفاده کرد. در این فرآیند PHP خود را به‌عنوان Client به سرور راه دور معرفی می‌کند. نمونه یک درخواست در این فرآیند به صورت زیر است:

```
GET / php /  hello. Html  HTTP / 1.0
Host : www. corrosiv. co. uk
User _ Agent : PHP / 4.0.6
```

علاوه بر تابع ( ) fopen به کمک تابع ( ) include نیز می‌توانید به فایل‌های موجود برروی سرور راه دور دسترسی پیدا کنید. درصورتی که گزینه allow _ url_fopen با مقدار on تنظیم شده باشد، با ارسال یک آدرس URL معتبر به تابع ( ) include فایل درخواستی به آدرس URL فوق به‌عنوان بخشی از برنامه اسکریپت در آن شامل خواهد شد (البته درصورت موفقیت آمیز بودن فرآیند).

از این روش تنها هنگامی استفاده کنید که از روند عملیات اطلاع کافی و جامع داشته باشید. توجه داشته باشید که شامل‌کردن کد تولید شده توسط سایر توسعه‌دهندگان در برنامه خود مسأله‌ای است که همواره با خطرات جانبی همراه است.

این فرآیند کاملاً ساده بوده و همواره می‌توان از آن‌جهت دستیابی به صفحات وب استفاده نمود. با این حال ممکن است اتصال به سرویس دیگری از شبکه مدنظرتان بوده و یا تمایل داشته باشید تا مطالب بیشتری درباره سند وب مورد نظرتان از طریق تحلیل و بررسی اطلاعاتی که Server در هدر پاسخ برای شما ارسال کرده، به‌دست آورید. روش انجام این کار را به‌زودی در درس همین ساعت فرا می‌گیرید.

تبدیل آدرسهای IP و اسامی میزبانها

حتی درصورتی که وب سرور مورد استفاده شما متغیر $REMOTE _ HOST را در اختیارتان قرار ندهد، به‌احتمال قوی خواهید توانست آدرس IP بازدیدکنندگان سایت را از طریق متغیر سیستمی دیگری با نام $REMOTE _ ADDR تشخیص دهید. از این متغیر سیستمی می‌توانید به‌همراه تابع ( ) gethostbyaddr جهت تعیین نام میزبان استفاده نمایید. این تابع به دنباله کاراکتری به‌عنوان

آرگومان نیاز دارد که نماینده یک آدرس IP باشد. مقدار بازگشتی تابع فوق نام میزبان مربوطه است. چنانچه در این میان خطایی رخ دهد، تابع ( ) gethostbyaddr آرگومان دریافتی را به‌عنوان نتیجه عملیات بازمی‌گرداند. برنامه موجود در لیست ۵- ۱۳ سعی دارد تا با بهره‌گیری از این تابع (درصورت در دسترس نبودن متغیر سیستمی $REMOTE _ HOST) نام میزبان کاربر را مشخص کند.

```
 1: <html>
 2: <head>
 3: <title>Listing 13.5 Using gethostbyaddr() to get a host name</title>
 4: </head>
 5: <body>
 6: <?php
 7: if ( isset( $REMOTE_HOST ) )
 8:     print "Hello visitor at $REMOTE_HOST<br>";
 9: elseif ( isset ( $REMOTE_ADDR ) )
10:       print "Hello visitor at ".gethostbyaddr( $REMOTE_ADDR )."<br>";
11: else
12:     print "Hello you, wherever you are<br>";
13: ?>
14: </body>
15: </html>
```

لیست ۵- ۱۳ بهره‌گیری از تابع ( ) gethostbyaddr جهت تعیین نام میزبان

همان‌گونه که مشاهده می‌کنید چنانچه برنامه به متغیر $REMOTE _ HOST دسترسی داشته باشد، درخط ۸ اقدام به نمایش محتوای آن می‌کند. درغیر این‌صورت برنامه سعی می‌کند تا با دسترسی به متغیر سیستمی $REMOTE _ ADDR و بهره‌گیری از آن به‌عنوان آرگومان ( ) gethostbyaddr نام میزبان را بر روی صفحه نمایش دهد (خط ۱۰). چنانچه عملیات فوق با شکست مواجه شود برنامه در خط ۱۲ اقدام به نمایش یک پیغام عمومی بر روی صفحه می‌کند.

فرآیند معکوس، یعنی تبدیل نام میزبان به آدرس IP مربوطه از طریق به‌کارگیری تابع ( ) gethostbyaddr قابل انجام است. این تابع نام میزبان را به‌عنوان آرگومان ورودی دریافت می‌کند. درصورت موفقیت‌آمیز بودن عملیات، تابع فوق آدرس IP متناظر با نام میزبان ورودی را باز می‌گرداند. چنانچه عملیات با شکست مواجه شود، تابع مورد بحث آرگومان ورودی را به‌عنوان مقدار بازگشتی به تابع فراخواننده ارسال می‌کند.

برقراری اتصال با شبکه

تا بدین‌جا کار سختی را درمورد بازیابی اسناد مورد نیاز از وب درپیش نداشتیم. این بدان دلیل بود که PHP فرآیند دستیابی به اسناد وب موجود بر روی سرورهای راه دور را به اندازه بازکردن یک فایل موجود بر روی سیستم ساده‌کرده است. با این حال گاهی اوقات لازم است تا کنترل بیشتری بر روی اتصال شبکه داشته باشیم یا اطلاعات بیشتری در مورد اتصال شبکه در دسترس باشد.

با استفاده از تابع ( ) fsockopen، می‌توانیم اتصالی را با یک سرور شبکه برقرار کنیم. این تابع چهار آرگومان ضروری و یک آرگومان اختیاری دریافت می‌کند. آرگومانهای ضروری این تابع نام میزبان (یا آدرس IP متناظر با آن)، یک شماره پورت ارتباطی و دو متغیر مرجع می‌باشند. دو متغیر مرجع جهت ارسال اطلاعاتی در مورد اتصال فوق مورد استفاده قرار می‌گیرند. از این اطلاعات تابع مورد بحث هنگام وقوع خطا در برقراری اتصال استفاده می‌کند. آرگومان اختیاری تابع ( ) fsockopen یک عدد صحیح است که به‌عنوان آخرین آرگومان مورد استفاده قرار می‌گیرد. این آرگومان مدت زمانی را برحسب ثانیه مشخص می‌کند که تابع ( ) fsockopen منتظر برقراری اتصال با سرور می‌ماند. با سپری‌شدن زمان مذکور و عدم برقراری اتصال با سرور، تابع فوق دست از تلاش برداشته و مقدار false را به برنامه فراخواننده باز می‌گرداند. درصورت موفقیت آمیز بودن عملیات، یعنی برقراری اتصال صحیح با سرور تابع فوق مرجعی را به یک فایل باز می‌گرداند.

فراخوانی زیر چگونگی استفاده از این تابع را جهت برقراری اتصال به یک سرور نشان می‌دهد:

```
$fp = fsockopen ("www. corrosive. co.uk", 80, $errno, $errdesc, 30) ;
```

پورت شماره ۸۰ پورتی است که وب سرور از طریق آن درخواستها را تحت نظر می‌گیرد.

اولین متغیر مرجع در فراخوانی فوق، یعنی $errno، حاوی شماره خطایی است که به هنگام عدم موفقیت‌آمیز بودن فرآیند اتصال مشخص می‌شود. متغیر دوم یعنی $errdesc نیز شامل توصیفی درباره علت خطاست.

با در دست داشتن مرجع فایل بازگشتی از تابع ( ) fsockopen، می‌توان به کمک توابع( ) fputs و ( ) fgets فایل موردنظر را مورد بازنویسی یا بازخوانی قرار داد (درست مشابه وضعیتی که در مورد فایل‌های معمولی با آنها مواجه بودیم). پس از اتمام کار موردنظر لازم است تا با استفاده از تابع)fclose (اتصال را ببندید (به تشابه زیاد این فرآیند با فرآیند کار با فایل‌ها توجه کنید).

اکنون با دانستن این مطالب می‌توانیم اتصال مورد نظرمان را با یک وب سرور برقرار کنیم. برنامه موجود در لیست ۶- ۱۳ پس از برقراری اتصالی از نوع HTTP با وب سرور موردنظر اقدام به بازیابی یک صفحه از آن و ذخیره آن در یک متغیر می‌کند.

```
 1: <html>
 2: <head>
 3: <title>Listing 13.6 Retrieving a Web page using fsockopen()</title>
 4: </head>
 5: <body>
 6: <?php
 7: $host = "www.corrosive.co.uk";
 8: $page = "/index.html";
 9: $fp = fsockopen( "$host", 80, $errno, $errdesc);
10: if ( ! $fp )
11:     die ( "Couldn't connect to $host:\nError: $errno\nDesc: $errdesc\n" );
```

لیست ۶- ۱۳ بازیابی یک صفحه وب با استفاده از تابع ( ) fsockopen

```
12:
13: $request = "GET $page HTTP/1.0\r\n";
14: $request .= "Host: $host\r\n";
15: $request .= "Referer: http://www.corrosive.co.uk/refpage.html\r\n";
16: $request .= "User-Agent: PHP test client\r\n\r\n";
17:
18: $page = array();
19: fputs ( $fp, $request );
20: while ( ! feof( $fp ) )
21:     $page[] = fgets( $fp, 1024 );
22: fclose( $fp );
23: print "the server returned ".(count($page))." lines!";
24: ?>
25: </body>
26: </html>
```

دنباله لیست ۶- ۱۳

توجه‌کنیدکه هدرهای درخواست (خطوط ۱۳ تا ۱۶) در خط ۱۹ برنامه با استفاده از تابع fputs ( ) به سرور ارسال شده است. راهبر سیستم راه دوری که به آن متصل شده‌ایم، قادر است تا مقدار ارسال شده به‌عنوان هدر USER _ AGENT را در فایل ثبت چنین اطلاعاتی مشاهده نماید. همچنین ممکن است، چنین فرض کند که کاربر بازدیدکننده از این صفحه از طریق پیوند زیر:

http: // www. corrosive. co. uk / refpage. html اقدام‌به برقراری اتصال با سرور نموده است. به‌همین‌دلیل،همواره‌لازم است تا توجه خاصی به برخی از متغیرهای سیستمی مورد استفاده در برنامه‌ها داشته‌باشید.به‌جای اعتماد صددرصد به این متغیرها، همواره از وجودشان به‌عنوان راهنما استفاده کنید.

دلایل مختلفی ممکن است برای تحریف یک هدر وجود داشته باشد. برای مثال، ممکن است لازم باشد تا برخی از داده‌هایی را که تنها به مرورگر اینترنت Netscape و مرورگرهای سازگار با آن ارسال می‌کنید، موردبررسی و تحلیل و پردازش قرار دهید. یکی از راههای انجام این کار استفاده از واژه "Mozilla" در هدر USER _ AGENT است. با این‌حال بهتر است در مورد راهبر سیستم (وب سرور) اندکی منصف باشید چراکه بر مبنای آمارهایی که وی از سرور جمع‌آوری می‌کند، معمولاً تصمیمات مهمی اتخاذ می‌شود. از این‌رو بهتر است از ایجاد اطلاعات نادرست که در نتیجه آمارگیری مؤثر است، خودداری کنید.

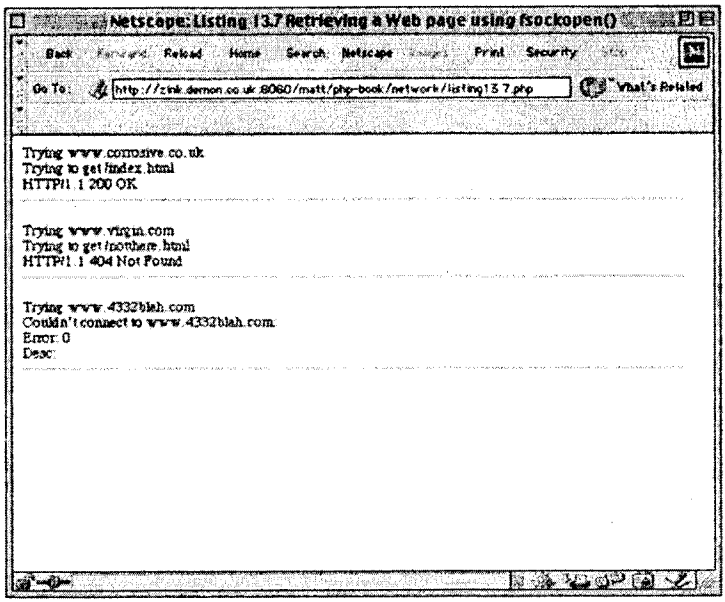
برنامه موجود در لیست ۶- ۱۳ توانایی و قابلیت اندکی را به تابع سیستمی PHP مورد استفاده جهت بازیابی صفحات وب از وب سرور اضافه کرد. برنامه لیست ۷- ۱۳ با استفاده از تابع fsockopen( ) مقادیر کدهای بازگشتی از سرور را دررابطه با بازیابی صفحات از وب سرور مورد ارزیابی قرار می‌دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 13.7 Outputting the status lines returned by web encountered a
➥servers</title>
 4: </head>
 5: <body>
 6: <?php
 7: $to_check = array (
 8:                     "www.corrosive.co.uk"  =>  "/index.html",
 9:                     "www.virgin.com"       =>  "/notthere.html",
10:                     "www.4332blah.com"     =>  "/nohost.html"
11:             );
12:
13: foreach ( $to_check as $host => $page ) {
14:     $fp = fsockopen( "$host", 80, $errno, $errdesc, 10);
15:     print "Trying $host<BR>\n";
16:     if ( ! $fp ) {
17:         print "Couldn't connect to $host:\n<br>Error: $errno\n<br>Desc:
➥$errdesc\n";
18:         print "<br><hr><br>\n";
19:         continue;
20:     }
21:     print "Trying to get $page<br>\n";
22:     fputs( $fp, "HEAD $page HTTP/1.0\r\n\r\n" );
23:     print fgets( $fp, 1024 );
24:     print "<br><br><br>\n";
25:     fclose( $fp );
26: }
27:
28: ?>
29: </body>
30: </html>
```

لیست ۷- ۱۳ نمایش کدهای بازگشتی توسط یک وب سرور در پاسخ به تلاش یک برنامه Client جهت دستیابی به چند صفحه مختلف از وب سرور

همان‌گونه که مشاهده می‌کنید در این برنامه از یک آرایه انجمنی جهت ذخیره اسامی سرورها و آدرس صفحات متناظر استفاده شده است (خط ۷). در خط ۱۳ با بهره‌گیری از یک حلقه تکرار هریک از عناصر، این آرایه مورد پردازش قرار گرفته است. در هربار گذر از حلقه فوق ابتدا توسط تابع fsockopen( ) در خط ۱۴ اتصالی با سرور مورد نظر ایجاد شده‌است. مدت زمان انتظار جهت دریافت پاسخ از سرور برابر با ۱۰ ثانیه فرض شده است. چنانچه فرآیند اتصال با شکست مواجه شود، عبارت خط ۱۷ پیغامی را جهت مشاهده کار بر روی صفحه مرورگر نمایش داده و دستورالعمل continue در خط موجب تکرار فرآیند سرور بعدی می‌شود. درصورتی که فرآیند اتصال موفقیت‌آمیز باشد، تابع fputs ( ) در خط ۲۲ درخواستی را به سرور ارسال می‌کند. در این تابع از آن جهت از روش درخواست HEAD استفاده شده است که درخواست شامل بدنه نمی‌باشد. تابع fgets ( ) در خط ۲۳ خط وضعیت درخواست را از سرور دریافت می‌کند. از آنجا که این برنامه از هدرهای سرور بهره‌ای نمی‌برد، اتصال در خط ۲۵ با فراخوانی تابع fclose ( ) بسته شده و فرآیند برای سرور بعدی تکرار می‌شود.

خروجی حاصل از این برنامه در شکل ۲- ۱۳ قابل بررسی است.



شکل ۲-۱۳ نتایج حاصل از تلاش برنامه ۷- ۱۳ جهت برقراری اتصال با چند سرور مختلف

درصورتی‌که قصددارید تا برنامه‌نویسی سمت Client را با جدیت بیشتری دنبال‌کنید، توصیه می‌کنیم نگاهی به بسته نرم‌افزاری CURL بیندازید. در این‌رابطه اطلاعات موردنیاز را می‌توانید از طریق آدرس http: //curl. haxx. Se دریافت کنید. با پشتیبانی PHP4.02 از این بسته‌نرم‌افزاری‌مفیدمی‌توانیدبرخی از ویژگی‌های قابل توجه و مفید HTTP مانند اعتبارسنجی ازطریق نام کاربر و کلمه عبور، کوکی‌ها و ارسال فرمها به روش POST را به‌راحتی در برنامه‌های خود مورد استفاده قراردهید. علاوه بر اینها بسته نرم‌افزاری CURL امکان بسیار جالب‌توجهی را برای انجام تراکنشهای مالی امن از طریق قرارداد HTTPS و طیف دیگری از قراردادها در اختیارتان قرارمی‌دهد. برای کسب اطلاعات بیشتر در این زمینه به راهنمای PHP در آدرس زیر مراجعه کنید:

<http: // www. php. net / manual/ en/ ref. curl. php >

برقراری اتصال با یک سرور NNTP با استفاده از تابع ( ) fsockopen

تابع ( ) fsockopen توانایی برقراری اتصال با هرنوع سروراینترنتی را دارد (منظور سرورهایی است که خدمات اینترنتی را دراختیار می‌گذارند). برنامه موجود در لیست ۸- ۱۳ سعی دارد تا با

استفاده از این تابع به سروری از نوع NNTP متصل شده و پس از انتخاب گروه خبری مورد نظر عنوان اولین پیغامها را برروی صفحه نمایش دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 13.8 A basic NNTP connection using fsockopen()</title>
 4: </head>
 5: <body>
 6: <?php
 7: $server = "news"; // change this to your news server
 8: $group  = "alt.test";
 9: $line = "";
10: print "<pre>\n";
11: print "— Trying to connect to $server\n\n";
12:
13: $fp = fsockopen( "$server", 119, $error, $description, 10 );
14: if ( ! $fp )
15:     die("Couldn't connect to $server\n$errno\n$errdesc\n\n");
16: print "— Connected to $server\n\n";
17:
18: $line = fgets( $fp, 1024 );
19: $status = explode( " ", $line );
20:
21: if ( $status[0] != 200 ) {
22:     fputs( $fp, "close" );
23:     die("Error: $line\n\n");
24: }
25:
26: print "$line\n";
27: print "— Selecting $group\n\n";
28: fputs( $fp, "group alt.test\n" );
29: $line = fgets( $fp, 1024 );
30: $status = explode( " ", $line );
31:
32: if ( $status[0] != 211 ) {
33:     fputs( $fp, "close" );
34:     die("Error: $line\n\n");
35: }
36:
37: print "$line\n";
38: print "— Getting headers for first message\n\n";
39: fputs( $fp, "head\n" );
40: $line = fgets( $fp, 1024 );
41: $status = explode( " ", $line );
42: print "$line\n";
43:
44: if ( $status[0] != 221 ) {
45:     fputs( $fp, "close" );
46:     die("Error: $line\n\n");
47: }
48:
49: while ( ! ( strpos($line, ".")  === 0 ) ) {
50:     $line = fgets( $fp, 1024 );
51:     print $line;
```

لیست ۸- ۱۳ بهره‌گیری از تابع( ) fsockopen جهت اتصال به سرور NNTP

```
52: }
53:
54: fputs( $fp, "close\n" );
55: print "</pre>";
56: ?>
57: </body>
58: </html>
```

دنباله لیست ۸- ۱۳

برنامه موجود در این لیست درواقع نمایشی بیش از توانایی تابع ( ) fsockopen جهت اتصال به یک سرور NNTP است. در برنامه‌های واقعی معمولاً لازم است تا خط به خط محتوای ورودی مورد پردازش قرار بگیرد. این امر جهت جلوگیری از دوباره‌کاری و همچنین جهت کسب اطلاعات بیشتر در مورد محتوای مورد نظر از سرور بسیار مفید است. در این راستا به‌جای ساختن ابزارهای مورد نیاز برای این کار می‌توانیم از توابع IMAP موجود درPHP استفاده کنیم. این توابع علاوه بر امکاناتی که برای برقراری اتصال با سرورهای NNTP و pop3 ، بسیاری از کارهای مورد نیاز را به‌طور خودکار انجام می‌دهند.

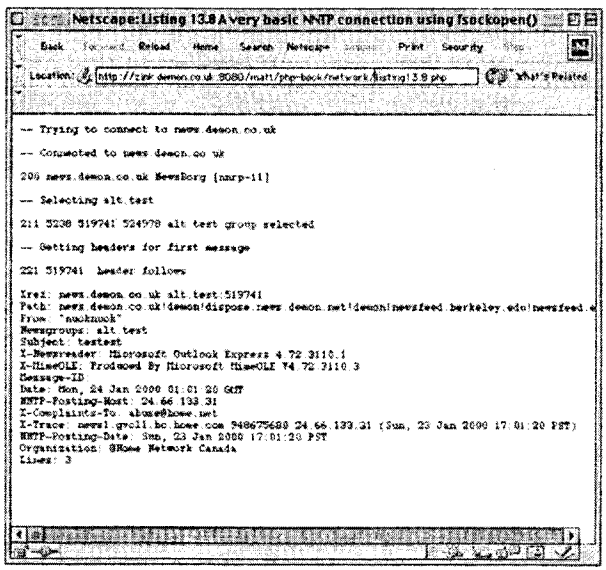
همان‌گونه که در این برنامه مشاهده می‌کنید در خطوط ۷ و ۸ نام میزبان سرور مورد نظر در متغیری با نام $server و همچنین نام گروه خبری مورد نظر نیز در متغیر $group ذخیره شده‌اند. جهت اجرای این برنامه لازم است تا نام میزبان سرور NNTP مربوط به ISP مورد استفاده‌تان را در متغیر $server ذخیره کنید. در خط ۱۳ این برنامه از تابع ( ) fsockopen، جهت اتصال به میزبان از طریق پورت شماره ۱۱۹ اقدام شده است (پورت شماره ۱۱۹ پورت معمولی جهت اتصال به سرورهای NNTP است). اگر تابع ( ) fsockopen مرجع معتبری را به یک فایل بازنگرداند تابع ( ) die در خط ۱۵ جهت نمایش شماره خطا و پیغام خطای مربوطه بر روی صفحه نمایش فراخوانی می‌شود. به‌هنگام برقراری اتصال، سرور پیغامی را جهت تایید ارسال می‌کند. جهت دریافت این پیغام از تابع ( ) fgets در خط ۱۸ استفاده شده است. اگر کل فرآیند بدون هیچ مشکلی انجام شود پیغام فوق با کد وضعیت 200 آغاز می‌شود. جهت بررسی و اطمینان از این وضعیت در خط ۱۹ تابع ( ) explode فراخوانی شده است. این تابع دنباله کاراکتری ذخیره شده در متغیر ( ) $line را به اجزای تشکیل دهنده خود تقسیم می‌کند (این تقسیم‌بندی بر مبنای فضاهای خالی موجود در دنباله کاراکتری انجام می‌شود). نتیجه این تقسیم‌بندی در یک آرایه ذخیره می‌شود (در درس ساعت هفدهم مطالب بیشتری درباره تابع ( )explode فرا می‌گیرد). چنانچه اولین عنصر این آرایه برابر با دنباله کاراکتری "200" باشد، فرآیند ادامه پیدا می‌کند؛ در غیر این‌صورت برنامه به کار خود خاتمه می‌دهد.

درصورتی که فرآیند مطابق انتظار پیش برود، در خط ۲۸ فرمان " group " جهت انتخاب گروه خبری موردنظر به سرور NNTP ارسال می‌شود. چنانچه این عمل با موفقیت انجام شود سرور در پاسخ به آن یک دنباله کاراکتری را که با کد 211 آغاز می‌شود، برای Client ارسال می‌کند. باردیگر برنامه در

خط ۳۲ با تحلیل دنباله کاراکتری بازگشتی به‌صورتی که در پاراگراف قبل توضیح داده شد در مورد ادامه عملیات یا پایان دادن به آن تصمیم‌می گیرد.

پس از انتخاب گروه خبری مورد نظر، برنامه می‌تواند با استفاده از فرمان " head " عناوین اولین پیغام موجود در گروه خبری مزبور را درخواست نماید. این فرآیند در خط ۳۹ انجام شده است. باردیگر در خط ۴۴ پاسخ سرور مورد بررسی و ارزیابی واقع شده است. این‌بار درصورتی که دنباله کاراکتری بازگشتی از سرور در پاسخ به درخواست فوق با کد 221 آغاز شود، بدین معنی است که سرور درکار خود موفق بوده است. بدین ترتیب عناوین خبری مورد نیاز در اختیار برنامه قرار می‌گیرد. از آنجا که هریک از عناوین ارسالی توسط سرور با علامت نقطه به پایان می‌رسد، با استفاده از یک ساختار تکرار می‌توان وضعیت را مورد بررسی قرار داد. این فرآیند در خط ۴۹ انجام شده است. بدین ترتیب تا زمانی که عنوان ارسالی از سرور با علامت نقطه آغاز نشده باشد، برنامه عنوان بعدی را مورد درخواست قرار خواهد داد.

درنهایت پس از بازیابی اطلاعات درخواستی از سرور اتصال بسته می‌شود. شکل ۳- ۱۳ خروجی حاصل از این برنامه را نشان می‌دهد.



شکل ۳- ۱۳ خروجی حاصل از برقراری اتصالی از نوع NNTP و دریافت عناوین خبری

ارسال e-mail با استفاده از تابع ( ) mail

PHP می‌تواند فرآیند ارسال نامه‌های الکترونیکی شما را مکانیزه کند. این فرآیند با استفاده از

تابع ( ) mail انجام می‌شود. این تابع سه دنباله کاراکتری را به‌عنوان آرگومان دریافت می‌کند. این آرگومانها به ترتیب آدرس مقصد، موضوع نامه و بالاخره متن آن‌را مشخص می‌کنند. چنانچه تابع mail ( ) در روند عملیات خود با مشکلی مواجه شود، مقدار false را باز می‌گرداند. قطعه کد زیر چگونگی استفاده از این تابع را نشان می‌دهد:

```
$to = "someone @ adomain. com" ;.
$subject = "hi" ;
$message = "just a test message !" ;
mail( $to, $subject, $message) or print "couldn't send mail" ;
```

چنان چه تحت سیستم عامل UNIX از PHP استفاده می‌کنید، تابع ( ) mail جهت انجام عملیات خود از یک برنامه کاربردی مانند Sendmail استفاده خواهد کرد. در مورد سایر سیستمها این تابع از امکاناتی که سرور محلی یا راه دور SMTP در اختیار قرار می‌دهد، استفاده می‌کند. تعیین جزییات این فرآیند با تنظیم گزینه SMTP از فایل php. ini امکان‌پذیر است.

امکانات تابع ( ) mail به عناوینی که آرگومآنهای ضروری این تابع تعیین می‌کنند، محدود نمی‌شود. درواقع از طریق چهارمین آرگومان این تابع که یک آرگومان اختیاری است، می‌توان عناوین بیشتری را نیز مشخص نمود. هریک از این عناوین باید از طریق کاراکترهای CRLF (ترکیب ' \ r \ n ') از یکدیگر جدا شوند. در قطعه کد نمونه زیر علاوه بر استفاده از فیلد From (مبدا یا فرستنده email) از هدری با عنوان X -Priority نیز استفاده شده که البته برخی از برنامه‌های Client قادر به تشخیص آن هستند:

```
$to = "someone @ adomain. com" ;
$subject= "hi" ;
$from = "book @ corrosive. co. uk" ;
$message = "just a test message !" ;
mail ($to, $subject, $message", $from \ r \ nX - Priority : 1
   (Highest)" ) or print "couldn't send mail" ;
```

در نسخه PHP4.0.5 آرگومان دیگری نیز به‌عنوان آرگومان اختیاری تابع ( ) mail معرفی شده است که از طریق آن می‌توان پارامترهایی مشابه پارامترهای مورد استفاده در سطر فرمان را مستقیماً به برنامه SMTP ارسال کرد.

جمع‌بندی

دردرس این‌ساعت چگونگی استفاده از متغیرهای سیستمی را که از طریق آنها می‌توان اطلاعات بیشتری‌درباره‌بازدیدکنندگان‌سایت کسب کرد، مورد بحث و بررسی قرار دادیم. اگر به‌نام میزبان کاربرتان دسترسی ندارید اکنون چگونگی دستیابی به آن‌را از طریق تابع ( ) gethostbyaddr می‌دانید.

درس این ساعت بحث مفصلی درمورد چگونگی اتصال و بهره‌برداری از آن دررابطه بین

برنامه‌های Client و Server بر پایه قرارداد HTTP ارائه داد.

در این درس با چگونگی بهره‌گیری از تابع ( ) fopen جهت دستیابی به سندی از وب و همچنین نحوه استفاده از تابع ( ) fsockopen را به‌منظور برقراری یک اتصال HTTP فراگرفتید. اکنون باید بتوانید با به کارگیری این تابع اتصال مورد نیاز خود با سایر سرویس‌های شبکه را برقرار کنید. علاوه براین باید بتوانید از طریق تابع ( ) mail اقدام به ارسال email از درون برنامه‌ها نمایید.

اگر دقت کرده باشید تا بدین جای کتاب بحث ما صرفاً در مورد متن و چگونگی نمایش آن بود. دردرس ساعت آینده با تکنیک‌هایی آشنا می‌شوید که از طریق آنها می‌توان اقدام به ساخت و دستکاری تصاویر نمود.

پرسش و پاسخ

پرسش: قرارداد HTTP و جزئیات مربوطه اندکی گیج کننده به‌نظر می‌آیند. آیا واقعاً اطلاع از این مسایل جهت تولید برنامه‌های کارآمد توسط PHP ضروری است؟

پاسخ: خیر.حتی‌بدون‌اطلاع از جزئیات مربوط به رابطه Client / Server نیز می‌توان برنامه‌های بسیار کارآمد و قابل توجهی نوشت. با این‌حال درک مفاهیم اصلی این رابطه کمک بسیار زیادی جهت نوشتن برنامه‌هایی است که کاری بیشتر از دست‌یابی به صفحات موجود بر روی سرورهای راه دور و بارگذاری آنها انجام می‌دهند.

پرسش: اگربتوان هدرهای جعلی به سرورهای راه دور فرستاد، پس چگونه می‌توان به متغیرهای سیستمی در این رابطه اعتماد کرد؟

پاسخ: درصورتی‌که‌صحت‌متغیرهای‌سیستمی‌همچون$HTTP _ REFERER و $HTTP_USER _ AGENT کاملاً وابسته به برنامه شما باشد، نباید به آنها اعتماد کنید. با این وجود به‌خاطر داشته باشید که بخش بزرگی از بازدیدکنندگان از سایت حقایقی را که انتظار دارید دراختیارتان قرار می‌دهند. درصورتی که در این تجربه به تشخیص نوع مرورگر مورد استفاده کاربر تکیه کرده یا از اطلاعات آماری جمع‌آوری شده استفاده می‌کنید، نیازی نیست تا مساله اعتماد یا عدم اعتماد به این‌گونه متغیرهای سیستمی آزارتان دهد.

تمرینها

هدف از این بخش ارائه تمرینهایی در قالب آزمون است. پاسخ بخش آزمون بلافاصله بعد از آن آمده است. بخش فعالیتها شامل تمرینهایی است که جهت افزایش قابلیت برنامه‌نویسی خواننده طراحی شده و البته فاقد پاسخ است.

آزمون

۱- از کدام متغیر سرور می‌توان جهت تشخیص آدرس URL صفحه درخواست کننده استفاده کرد؟

۲- چرا نمی‌توان جهت تشخیص و ردیابی کاربری که چندین مرتبه اقدام به استفاده از برنامه اسکریپت کرده است از متغیر $REMOTE _ ADDR استفاده کرد؟

۳- اصطلاح HTTP کوتاه شده کدام عبارت است؟

۴- کدام هدر ارسالی از Client (جهت درخواست) نوع مرورگر مورد استفاده کاربر را برای برنامه Server فاش می‌کند؟

۵- کد پاسخ 404 به چه معنا می‌باشد؟

۶- بدون ایجاد اتصال شبکه از کدام تابع می‌توان به‌منظور دستیابی به صفحات وب موجود بر روی یک سرور راه دور دسترسی پیدا کرد؟

۷- چنانچه آدرس IP در دسترس باشد، توسط کدام تابع می‌توان نام میزبان متناظر با آن‌را مشخص کرد؟

۸- از کدام تابع می‌توان جهت ایجاد یک اتصال شبکه استفاده کرد؟

۹- از کدام تابع PHP می‌توان جهت ارسال email استفاده کرد؟

پاسخ آزمون

۱- آدرس URL صفحه درخواست‌کننده از برنامه server همواره در متغیر سیستمی $HTTP _ REFERER ذخیره می‌شود.

۲- بسیاری از مراکز ISP از روش Dynamic IP Addressing استفاده می‌کنند، بدین ترتیب که در هربار تمامی مشترکین خود آدرس IP متفاوتی را به آنها اختصاص می‌دهند. از این‌رو هرگز نمی‌توان با بهره‌گیری از متغیر سیستمی $REMOTE _ ADDR کاربران را مورد ردیابی قرار داد.

۳- اصطلاح HTTP کوتاه شده عبارت Hypertext Transfer Protocol است.

۴- برنامهClient با ارسال هدر USER _ AGENT اطلاعاتمورد نیاز در مورد نوع مرورگر اینترنت و سیستم عامل مورد استفاده کاربر را در اختیار Server قرار می‌دهد.
۵- کد پاسخ 404 که برنامه Server برای اطلاع Client ارسال می‌کند، بدین معنی است که صفحه مورد درخواست برروی Server پیدا نشده است.
۶- از تابع ( ) fopen می‌توان جهت دستیابی به اسناد وب موجود بر روی یک سرور راه دور و همچنین دستیابی به فایل‌های محلی استفاده کرد.
۷- تابع ( ) gethostbyaddr یک آدرس IP را به‌عنوان آرگومان ورودی پذیرفته و آن‌را به نام میزبان متناظر تبدیل می‌کند.
۸- تابع ( ) fsockopen اتصالی را با یک سرور راه دور برقرار می‌کند.
۹- به کمک تابع ( ) mail می‌توان اقدام به ارسال email از درون برنامه نمود.

فعالیتها

۱- یک برنامه اسکریپت ایجاد کنید که با دریافت یک نام میزبان (مانند http://www.microsoft. Com) از طریق فرم ورودی، درخواستی از نوع HEAD را پس از برقراری اتصال با میزبان فوق از طریق تابع( ) fsockopen برای آن ارسال نماید. پاسخ میزبان باید بر روی صفحه مرورگر اینترنت به نمایش درآید. به‌خاطر داشته باشید که برنامه باید بتواند مواردی را که طی آنها هیچ‌گونه اتصالی با سرور برقرار نمی‌شود، کنترل نماید.

۲- برنامه‌ای بنویسید که پیغامی را از ورودی دریافت کرده و از طریق email آن‌را برای شما ارسال کند. برنامه باید ضمیمه کردن متغیرهای سیستمی مورد نیاز به متن پیغام اطلاعاتی را نیز درباره مرورگر اینترنت مورد استفاده کاربر و آدرس IP وی در اختیارتان قرار دهد.

ساعت چهاردهم

بهره‌گیری از گرافیک

توابعی را که در درس این ساعت مورد بررسی قرار می‌دهیم بر مبنای یک کتابخانه کدباز با عنوان GD عمل می‌کنند.

کتابخانه GD مجموعه‌ای از ابزارهایی است که امکانات بسیار مفیدی را درارتباط با ایجاد تصاویر گرافیکی و پردازش آنها در اختیار برنامه‌نویس قرار می‌دهد. چنانچه مایل باشید، می‌توانید اطلاعات بیشتری را درباره این کتابخانه از طریق آدرس http: // www. boutell. com / gd / به دست آورید. درصورتی که درحال حاضر فایل‌های این کتابخانه را بر روی سیستم خود نصب کرده و PHP را نیز به‌گونه‌ای کامپایل کرده‌اید که قابلیت بهره‌گیری از آن را داشته باشد، هم‌اکنون قادرید تا توابع گرافیکی PHP را جهت ایجاد تصاویر گرافیکی پویا مورد استفاده قرار دهید. برخی از سیستمهاهنوز از نسخه‌های قدیمی‌تر این کتابخانه کدباز استفاده می‌کنند. نسخه‌های پیشین این کتابخانه امکاناتی را جهت ایجاد تصاویر گرافیکی در قالب GIF در اختیار استفاده‌کنندگان قرار می‌دهد. نسخه‌های جدیدتر به‌دلایل عدم برخورداری از حق امتیاز قالب گرافیکی مذکور را مورد پشتیبانی قرار نمی‌دهند. چنانچه از آن دسته کاربرانی هستید که از نسخه‌های جدید استفاده می‌کنید نگران نباشید، چراکه با کامپایل برنامه PHP به‌گونه‌ای که توابع گرافیکی این زبان بتواند از قالب تصویری PNG بهره‌مند شوند، می‌توانید آلام حاصل از این عدم پشتیبانی را کاهش دهید. جالب است بدانید که قالب تصویری PNG در حال حاضر توسط بیشتر مرورگرهای متداول مورد پشتیبانی قرارگرفته است.

با دراختیار داشتن کتابخانه GD می‌توانید توابع گرافیکی زبان برنامه‌نویسی PHP را جهت ایجاد تصاویر گرافیکی مورد نظرتان به‌کارگیرید. در درس این ساعت با مباحث زیر آشنا خواهید شد:

- **چگونگی ایجاد تصاویر و ارسال آنها به خروجی**
- **کار با رنگهای مختلف**
- **چگونگی رسم تصاویری چون کمان، چهارضلعی و چندضلعی**
- **چگونگی رنگ آمیزی تصاویر**
- **بهره‌گیری از فونت‌های True Type**

در ادامه به بررسی موارد فوق می‌پردازیم.

چگونگی ایجاد تصاویر گرافیکی و ارسال آنها به خروجی

پیش از هرگونه کاری با یک تصویر گرافیکی، دراختیار داشتن مرجعی به آن تصویر که امکان دسترسی به آن‌را فراهم کند، الزامی است. این مرجع را می‌توان با استفاده از تابع ( ) imagecreate به‌دست آورد. تابع ( ) imagecreate جهت انجام عملیات خود به دو آرگومان نیاز دارد. اولین آرگومان این تابع ارتفاع تصویر و دومین آرگومان آن نیز پهنای تصویر را مشخص می‌کند. حاصل عملیات این تابع مرجعی است به یک تصویر گرافیکی که آن‌را به برنامه فراخواننده باز می‌گرداند. با دراختیار داشتن این مرجع می‌توان بیشتر توابعی را که در درس این ساعت بررسی می‌کنیم، مورد استفاده و بهره‌برداری قرار دهیم. اغلب این توابع مرجع مورد بحث را به‌عنوان آرگومان ورودی مورد استفاده قرار می‌دهند. این‌گونه مراجع هنگام کار با فایل‌ها و بانکهای اطلاعاتی مورد استفاده قرار می‌گیرند و ما آنها را در درسهای مربوطه، پیشتر مورد بررسی قرار دادیم. لازم به ذکر است که مرجع بازگشتی از تابع ( ) imagecreate، آرگومان ضروری اکثر توابعی است که در درس این ساعت مورد بررسی قرار می‌دهیم.

پس از آنکه به واسطه تابع ( ) imagecreate مرجع تصویر موردنظر را به‌دست گرفتید، می‌توانید تصویر مربوطه را به‌عنوان خروجی بر روی صفحه مرورگر اینترنت نمایش دهید. برای انجام این‌کار از تابعی با عنوان ( ) imagegif استفاده می‌کنیم. این تابع همان‌گونه که احتمالاً حدس می‌زنید مرجع بازگشتی از تابع ( ) imagecreate را به‌عنوان آرگومان ضروری مورد استفاده قرار می‌دهد. برنامه موجود در لیست ۱- ۱۴ نمونه‌ای از به‌کارگیری دو تابع فوق را جهت ایجاد یک تصویر گرافیکی و ارسال آن به خروجی نشان می‌دهد.

```
1: <?php
2: header("Content-type: image/gif");
3: $image = imagecreate( 200, 200 );
4: imagegif($image);
5: ?>
```

لیست ۱- ۱۴ ایجاد و نمایش یک تصویر گرافیکی در مرورگر اینترنت

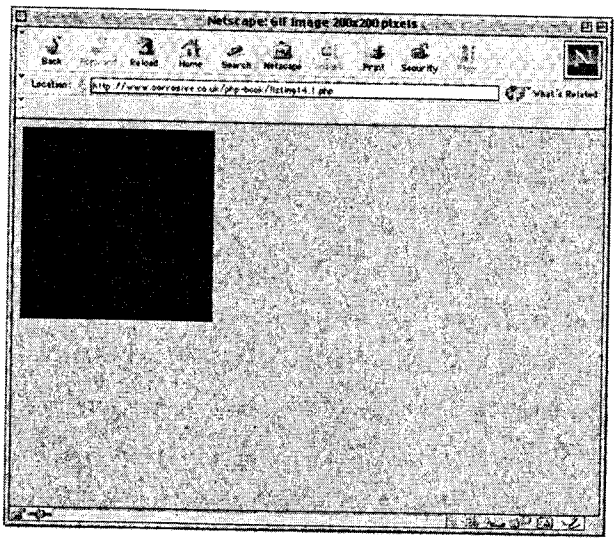
در این برنامه توجه کنید که هدر Content _ type که تعیین‌کننده نوع محتوایی است که مرورگر اینترنت باید نمایش دهد، در خط ۲ پیش از انجام هر عمل دیگری توسط تابع ( ) header به برنامه مرورگر ارسال شده است. این فرآیند، یعنی آگاه کردن مرورگر اینترنت از نوع اطلاعاتی که باید نمایش دهد از جمله ضروریات است. چنانچه برنامه ما در خط ۲ مرورگر را از وجود اطلاعات تصویری آگاه نمی‌کرد، مرورگر کاربر اطلاعات دریافتی را به‌سادگی یک صفحه HTML درنظر می‌گرفت. هم‌اکنون با وجود اطلاع مرورگر از محتوای تصویری می‌توانیم این برنامه را مستقیماً در مرورگر اینترنت

نمایش داده یا از آن در قالب نشانه < img > استفاده کنیم:

< img src = "listing 14.1.php" alt = "a PHP generatedimage" >

خروجی حاصل از این برنامه در شکل ۱- ۱۴ قابل توجه است.

آنچه که کاملاً واضح است، این است که ما توانستیم مربعی را بر روی پنجره مرورگر به نمایش درآوریم اما هیچ روشی را جهت کنترل رنگ آن به کار نبردیم.



شکل ۱- ۱۴ تصویر گرافیکی ایجاد شده توسط برنامه

استفاده از رنگ

برای کار با رنگها لازم است تا ابتدا مرجعی را که امکان استفاده از رنگ را دراختیارمان قرار می‌دهد، به‌دست آوریم. تابعی که چنین کاری را برای ما انجام می‌دهد، ( ) imagecolorallocate نام دارد. این تابع یک مرجع تصویری (حاصل فراخوانی تابع ( ) imagecreate) و سه عدد صحیح مابین صفر تا 255 را که به ترتیب نماینده رنگهای قرمز، آبی و سبز هستند به‌عنوان آرگومان ورودی دریافت می‌کند. تابع مورد بحث مرجعی را باز می‌گرداند که با استفاده از آن می‌توانیم رنگ یک شکل یا متن را مشخص‌کرده‌ویاشکلها را رنگ آمیزی کنیم. فراخوانی نمونه زیر نحوه عملکرد این تابع را نمایان می‌سازد:

```
$red = imagecolorallocate ($image, 255, 0, 0) ;
```

نکته قابل توجه درمورد فراخوانی تابع فوق این است که به محض اولین فراخوانی آن رنگ پیش‌فرض تصویر مورد نظر مشخص خواهد شد. بنابراین با افزودن فراخوانی فوق به برنامه موجود در

لیست ۱- ۱۴ آنچه که در خروجی به نمایش در می‌آید، یک مربع به رنگ قرمز خواهد بود.

رسم خطوط

پیش از رسم خطی بر روی یک تصویر لازم است تا نقاط ابتدایی و انتهایی خط مورد نظر خود را برروی تصویر مشخص نمایید. با تعیین این دو نقطه خط کاملاً قابل تعریف خواهد بود.

تصاویر را می‌توانیم به‌عنوان بلوک‌هایی از پیکسل فرض کنیم که از شماره صفر شاخص‌گذاری شده و در دو راستای افقی و عمودی توزیع شده‌اند. مرجع هر شکل، یعنی موقعیتی که سایر نقاط شکل نسبت به آن تعیین موقعیت می‌شوند، گوشه بالای سمت چپ آن است.

به عبارت بهتر، پیکسلی با موقعیت 5 و 8 محل برخورد ششمین پیکسل در راستای افق با نهمین پیکسل در راستای عمود می‌باشد (حرکت در راستای افق از چپ به راست و حرکت در راستای عمودی از بالا به پایین انجام می‌شود. محل شروع حرکت مرجع شکل است).

با استفاده از تابع ( ) imageline، می‌توان خط راستی را مابین دو نقطه مشخص از یک تصویر (مابین دو پیکسل) ترسیم نمود. این تابع جهت عملیات خود به شش آرگومان ضروری نیاز دارد. اولین آرگومان این تابع مرجع یک تصویر است. چهار آرگومان بعدی چهار عدد صحیح هستند که هر جفت از آنها نماینده موقعیتی از تصویر است:

دو عدد اول نقطه شروع و دو عدد آخر نقطه پایان خط را مشخص می‌کنند. آرگومان آخر مرجعی است که تعیین‌کننده رنگ مورد استفاده جهت ترسیم خط می‌باشد.

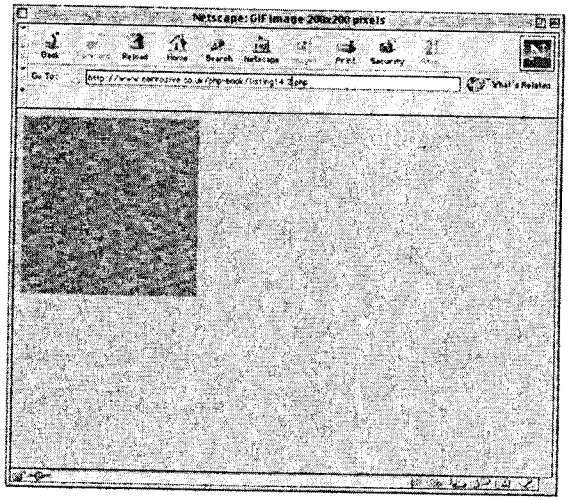
برنامه موجود در لیست ۲- ۱۴، قابلیت ترسیم خط راستی را از یک گوشه مربع به گوشه روبه روی آن به برنامه موجود در لیست ۱- ۱۴ اضافه کرده است.

```
1: <?php
2: header("Content-type: image/gif");
3: $image = imagecreate( 200, 200 );
4: $red = imagecolorallocate($image, 255,0,0);
5: $blue = imagecolorallocate($image, 0,0,255 );
6: imageline( $image, 0, 0, 199, 199, $blue );
7: imagegif($image);
8: ?>
```

لیست ۲- ۱۴ ترسیم خط راست با استفاده از تابع ( ) imageline

همان‌گونه که در این برنامه ملاحظه می‌کنید، از دو مرجع رنگ یکی برای تعیین رنگ قرمز (در خط ۴) و دیگری برای تعیین رنگ آبی (در خط ۵) استفاده کرده‌ایم. سپس در خط ۶ از مرجع رنگ آبی که در متغیر $blue ذخیره شده است، استفاده نموده‌ایم. دقت کنید که موقعیت انتهای خط پیکسلی از تصویر است که در موقعیت 199 و 199 واقع است که برخی به اشتباه آن‌را 200 و 200 فرض می‌کنند. به‌خاطر داشته باشید که شماره‌گذاری پیکسل‌ها از عدد صفر آغاز می‌شود. خروجی حاصل از

این برنامه در شکل ۲- ۱۴ قابل بررسی است.



شکل ۲- ۱۴ ترسیم خط با استفاده از تابع ( ) imageline

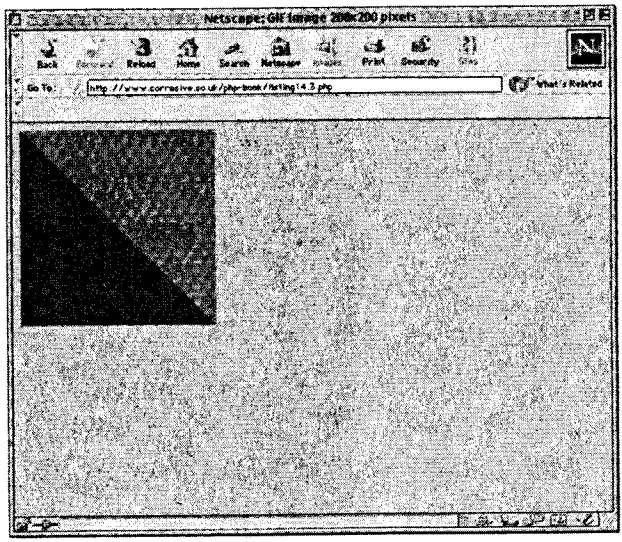
رنگ‌آمیزی تصاویر

به کمک ابزاری که PHP در اختیارتان می‌گذارد، می‌توانید بخشهایی از یک تصویر را همان‌گونه که با استفاده از یک برنامه کاربردی گرافیکی انجام می‌دهید، رنگ آمیزی کنید. این ابزار در زبان برنامه نویسی PHP چیزی نیست جز تابعی با نام ( ) imagefill که جهت انجام عملیات پیش‌بینی شده به چهار آرگومان نیاز دارد. اولین آرگومان این تابع مرجع یک تصویر است و دو آرگومان بعدی موقعیت شروع رنگ آمیزی را مشخص می‌کند. آرگومان آخر نیز مرجعی است که رنگ مورد استفاده در فرآیند رنگ آمیزی را تعیین می‌نماید. کاری که این تابع انجام می‌دهد به سادگی این است که نقطه شروع رنگ آمیزی را به رنگ تعیین شده توسط آرگومان آخر ترسیم کرده و سپس تمام نقاط موجود در همسایگی آن‌را رنگ آمیزی می‌کند. برنامه موجود در لیست ۳- ۱۴ با به کارگیری این تابع شکلی را در مرورگر اینترنت رنگ آمیزی می‌کند.

```
1: <?php
2: header("Content-type: image/gif");
3: $image = imagecreate( 200, 200 );
4: $red = imagecolorallocate($image, 255,0,0);
5: $blue = imagecolorallocate($image, 0,0,255 );
6: imageline( $image, 0, 0, 199, 199, $blue );
7: imagefill( $image, 0, 199, $blue );
8: imagegif($image);
9: ?>
```

لیست ۳- ۱۴ بهره‌گیری از تابع ( ) imagefill

توجه کنید که تنها تغییری که این برنامه نسبت به برنامه لیست ۲- ۱۴ دارد استفاده از تابع ( ) imagefill در خط ۷ است، خروجی این برنامه در شکل ۳- ۱۴ قابل مشاهده است.



شکل ۳- ۱۴ بهره‌گیری از تابع ( ) imagefill جهت رنگ آمیزی تصاویر

ترسیم کمان

با استفاده از تابع ( ) imagearc در زبان PHP می‌توان اقدام به ترسیم بخشی از یک کمان یا ترسیم کل آن نمود. تابع( ) imagearc جهت انجام عملیات خود به هشت آرگومان ضروری نیاز دارد. اولین آرگومان این تابع، مطابق انتظار مرجعی به یک تصویر است. آرگومان دوم و سوم موقعیت مرکز کمان را مشخص می‌کنند. آرگومانهای چهارم و پنجم این تابع تعیین کننده پهنا و ارتفاع کمان هستند. آرگومانهای ششم و هفتم نقاط شروع و پایان کمان را برحسب درجه مشخص می‌کنند و بالاخره آرگومان‌آخر مرجعی است که رنگ مورد استفاده جهت ترسیم کمان توسط این تابع را مشخص می‌کند. زوایای تعیین شده توسط آرگومانهای ششم و هفتم درجهت ساعتگرد و با شروع از ساعت ۳ در نظر گرفته می‌شوند. برای مثال، فراخوانی زیر را که موجب رسم کمانی به اندازه یک چهارم دایره می‌شود درنظر بگیرید:

```
imagearc ($image, 99, 99, 200, 200, 0, 90, $blue) ;
```

فراخوانی تابع مذکور باعث ترسیم بخشی از یک کمان کامل (دایره) می‌شود که مرکز آن در موقعیت 99 و 99 قرار دارد. ارتفاع و پهنای این کمان هریک به اندازه 200 پیکسل است که ترسیم آن از ساعت ۳ آغاز شده و به اندازه ۹۰ درجه در جهت ساعتگرد (ساعت ۶) ادامه پیدا می‌کند.

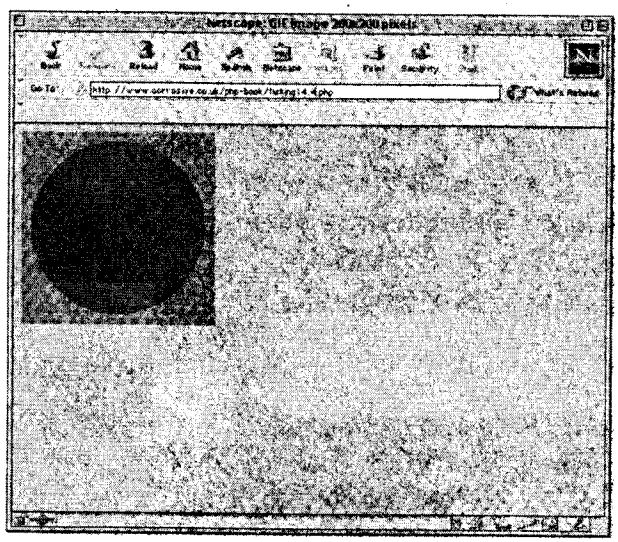
برنامه موجود در لیست ۴- ۱۴ دایره کاملی را با استفاده از تابع ( ) imagearc ترسیم کرده و آن‌را با رنگ آبی رنگ آمیزی می‌کند.

```
1: <?php
2: header("Content-type: image/gif");
3: $image = imagecreate( 200, 200 );
4: $red = imagecolorallocate($image, 255,0,0);
5: $blue = imagecolorallocate($image, 0,0,255 );
6: imagearc( $image, 99, 99, 180, 180, 0, 360, $blue );
7: imagefill( $image, 99, 99, $blue );
8: imagegif($image);
9: ?>
```

لیست ۴- ۱۴ ترسیم کمان با استفاده از تابع ( ) imagearc

همان‌گونه که مشاهده می‌کنید، همانند برنامه‌های گذشته در درس این ساعت، ابتدا در خطوط ۴ و ۵ مراجعی جهت تعیین رنگ مشخص شده‌اند. در خط ۶ از برنامه تابع ( ) imagearc به‌منظور رسم یک دایره کامل فراخوانی شده است. در نهایت با فراخوانی تابع ( ) imagefill در خط ۷ از برنامه، دایره ترسیم شده با رنگ آبی رنگ آمیزی شده است.

خروجی حاصل از اجرای این برنامه در شکل ۴- ۱۴ قابل مشاهده است.



شکل ۴- ۱۴ بهره‌گیری از تابع ( ) imagearc جهت ترسیم کمان

ترسیم چهارضلعی

با استفاده از تابع ( ) imagerectangle در زبان PHP می‌توان چهارضلعی‌هایی را دراندازه و

اشکال مختلف رسم کرد. تابع ( ) imagerectangle جهت اجرای عملیات خود به شش آرگومان ضروری نیاز دارد. بار دیگر، اولین آرگومان تابع فوق نماینده مرجعی به یک تصویر گرافیکی است. دو آرگومان بعدی مشخص‌کننده موقعیت گوشه بالای سمت چپ چهارضلعی است. آرگومانهای چهارم و پنجم موقعیت گوشه پایین و سمت راست چهارضلعی و بالاخره آرگومان آخر نیز رنگ مورد استفاده جهت ترسیم آن‌را تعیین می‌کند. فراخوانی زیر نمونه‌ای از به‌کارگیری این تابع را نشان می‌دهد. این فراخوانی موجب می‌شود تا یک چهارضلعی که مختصات گوشه بالای سمت چپ آن 179 و 179 بوده و مختصات گوشه پایین سمت راست آن 19 و 19 است، بر روی صفحه مرورگر اینترنت با رنگ آبی ترسیم شود:

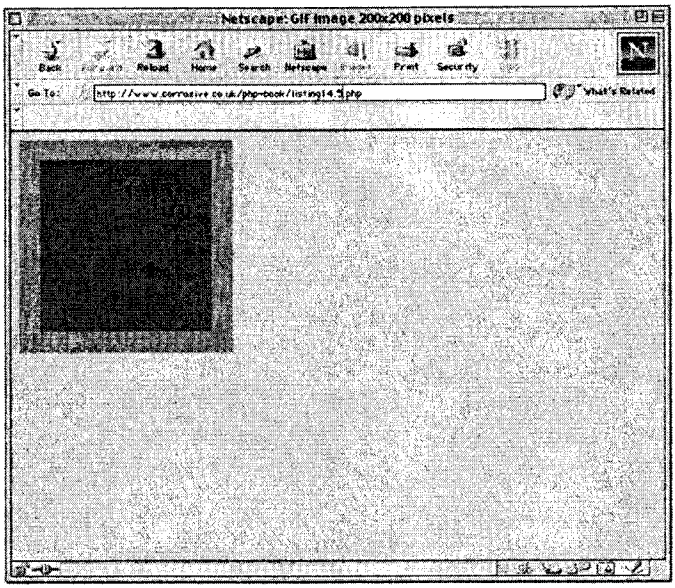
```
imagerectangle ($image, 19, 19, 179, 179, $blue) ;
```

پس از ترسیم چنین چهارضلعی بر روی صفحه می‌توان با استفاده از تابع ( ) imagefill اقدام به رنگ آمیزی آن نمود. از آنجا که دو فرآیند ترسیم چهارضلعی و رنگ آمیزی آن با یک رنگ مشخص معمولاً به‌دنبال هم انجام می‌شوند، PHP روشی را برای انجام آن در قالب یک فرآیند پیش روی ما می‌گذارد. این روش بهره‌گیری از تابع دیگری است که ( ) imagefilledrectangle نام دارد. آرگومانهایی که این تابع می‌پذیرد دقیقاً مشابه آرگومانهای مورد استفاده در تابع ( ) imagerectangl است. تنها تفاوت این دو تابع در عملکرد آنهاست. تابع جدید ( ) imagefilledrectangle چهارضلعی حاصل را با رنگی که توسط آخرین آرگومان آن مشخص شده است، رنگ آمیزی می‌کند. برنامه موجود در لیست ۵- ۱۴ از این تابع در خط ۶ استفاده کرده است. حاصل عملیات این برنامه تصویر یک چهارضلعی است که با رنگهای آبی بر روی صفحه مرورگر اینترنت نقش می‌بندد.

```
1: <?php
2: header("Content-type: image/gif");
3: $image = imagecreate( 200, 200 );
4: $red = imagecolorallocate($image, 255,0,0);
5: $blue = imagecolorallocate($image, 0,0,255 );
6: imagefilledrectangle( $image, 19, 19, 179, 179,  $blue );
7: imagegif( $image );
8: ?>
```

لیست ۵- ۱۴ استفاده از تابع ( ) imagefilledrectangle جهت ترسیم یک چهارضلعی رنگ آمیزی شده

خروجی حاصل از اجرای این برنامه در شکل ۵- ۱۴ قابل مشاهده است.



شکل ۵- ۱۴ترسیم یک چهارضلعی رنگ آمیزی شده با استفاده از تابع ( )imagefilledrectangle

ترسیم چندضلعی

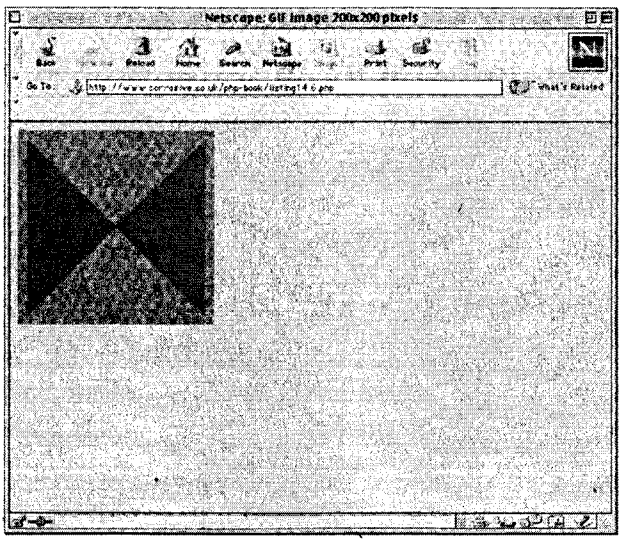
با بهره‌گیری از تابع ( ) imagepolygon می‌توانید اشکال پیچیده‌تری را نیز ترسیم کنید. این تابع‌جهت‌انجام‌عملیات خود به چهارآرگومان ضروری نیاز دارد. آرگومان اول طبق معمول مشخص‌کننده مرجع یک تصویر است. آرگومان دوم موقعیت مختلف (رئوس) چند ضلعی حاصل را مشخص می‌کند و آرگومان چهارم عدد صحیحی است که نماینده تعداد نقاط (رئوس) چندضلعی است. و بالاخره آرگومان آخر مرجعی است که رنگ مورد استفاده جهت ترسیم چند ضلعی را مشخص می‌کند. دومین آرگومان تابع ( ) imagepolygon، آرایه‌ای است که عناصر آن باید به‌صورت عددی شاخص‌گذاری شده باشند (به عبارت دیگر این آرگومان نباید یک آرایه انجمنی باشد). اولین دو عنصر این آرایه نماینده موقعیت اولین رأس چندضلعی است. دو عنصر بعدی این آرایه مشخص کننده موقعیت دومین رأس چندضلعی است و این رویه تا آخر ادامه می‌یابد. عملکرد تابع ( ) imagepolygon به گونه‌ای است که نقاط مشخص‌شده توسط این آرایه را با استفاده از خطوط راست به یکدیگر متصل می‌کند. تابع مذکور به‌طور خودکار جهت تشکیل چندضلعی، آخرین نقطه را به اولین نقطه وصل خواهد کرد. مشابه تابع ( ) imagerectangle در مورد تابع ( ) imagepolygon نیز تابع دیگری در زبان PHP تعریف شده که فرآیند ترسیم چندضلعی و رنگ آمیزی آن‌را به‌طور توأم طی یک فرآیند انجام می‌دهد. نام این تابع، همان‌گونه که حدس زده‌اید ( ) imagefilledpolygon است.

برنامه موجود در لیست ۶- ۱۴ موجب ترسیم یک چند ضلعی رنگ آمیزی شده به رنگ آبی بر روی صفحه مرورگر اینترنت می‌شود.

```
 1: <?php
 2: header("Content-type: image/gif");
 3: $image = imagecreate( 200, 200 );
 4: $red = imagecolorallocate($image, 255,0,0);
 5: $blue = imagecolorallocate($image, 0,0,255 );
 6: $points = array (     10, 10,
 7:             190, 190,
 8:             190, 10,
 9:             10, 190
10:             );
11: imagefilledpolygon( $image, $points, count( $points )/2 , $blue );
12: imagegif($image);
13: ?>
```

لیست ۶- ۱۴ ترسیم یک چندضلعی رنگ آمیزی شده با استفاده از تابع ( ) imagefilledpolygon

همان‌گونه که مشاهده می‌کنید در این برنامه پس از ایجاد دو مرجع مورد نیاز در رابطه با تصویر و رنگ در خطوط ۲ تا ۵ آرایه‌ای از اعداد صحیح در خط ۶ تعریف و مقداردهی شده است. دقت کنید که در فراخوانی تابع ترسیم چند ضلعی در خط ۱۱، یعنی ( ) imagefilledpolygon آرگومان سوم را چگونه مشخص کرده‌ایم. در این شیوه تعداد عناصر آرایه $points که حاوی موقعیت رئوس چندضلعی است به عدد ۲ تقسیم شده است. این شیوه از آن جهت مناسب است که همواره از صحت تعداد رئوس چند ضلعی به ما اطمینان می‌دهد. خروجی حاصل از اجرای این برنامه در شکل ۶- ۱۴ قابل مشاهده است.



شکل ۶- ۱۴ حاصل ترسیم یک چندضلعی رنگ آمیزی شده با استفاده از تابع imagefilledpolygon ( )

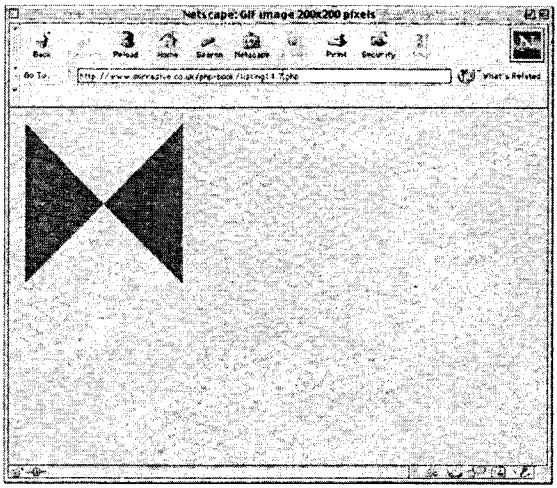
شفاف سازی رنگ

با استفاده از تابع ( ) imagecolortransparent در زبان PHP ، می‌توان برخی از رنگهای یک تصویر را شفاف نمود. ساختار این تابع ساده بوده و جهت فراخوانی تنها به دو آرگومان نیاز دارد. آرگومان اول مطابق معمول مرجع یک تصویر بوده و آرگومان دوم مرجعی است که رنگ موردنظر را مشخص می‌کند. حاصل عملیات تابع( ) imagecolortransparent ، شفاف‌سازی رنگی از تصویر است که توسط دومین آرگومان این تابع مشخص شده است. برنامه موجود درلیست ۷- ۱۴ از این تابع استفاده کرده و شفافیت جالب توجهی را به چند ضلعی واقع در صفحه افزوده است. اکنون به‌جای اینکه چندضلعی آبی بر روی یک پس‌زمینه رنگی به‌گونه‌ای ایستا مقیم شده باشد، چنین به‌نظر می‌آید بر روی صفحه مرورگر شناور شده است.

```
 1: <?php
 2: header("Content-type: image/gif");
 3:
 4: $image = imagecreate( 200, 200 );
 5: $red = imagecolorallocate($image, 255,0,0);
 6: $blue = imagecolorallocate($image, 0,0,255 );
 7:
 8: $points = array (     10, 10,
 9:              190, 190,
10:              190, 10,
11:              10, 190
12:              );
13:
14: imagefilledpolygon( $image, $points, count( $points )/2 , $blue );
15: imagecolortransparent( $image, $red );
16: imagegif($image);
17: ?>
```

لیست ۷- ۱۴ شفاف سازی رنگها با استفاده از تابع ( ) imagecolortransparent

همان‌گونه که ملاحظه می‌کنید، این لیست مشابه لیست ۶- ۱۴ است با این تفاوت که در خط ۱۵ از این برنامه، تابع ( ) imagecolortransparent فراخوانی شده است. خروجی حاصل از این برنامه در شکل ۷- ۱۴ نمایش داده شده است. اما بعید است از این طریق بتوانید تفاوت موجود را که در اثر استفاده از این تابع به‌وجود آمده درک کنید. از این‌رو توصیه می‌کنیم تا کد این برنامه را حتماً بر روی کامپیوترتان اجرا نمایید.



شکل ۷- ۱۴ شفاف‌سازی رنگ‌ها با استفاده از تابع ( ) imagecolortransparent

بهره‌گیری از متن

اگر فونت‌های TrueType بر روی کامپیوترتان موجود باشد، می‌توانید از آنها جهت نوشتن متون مورد نظرتان بر روی صفحه مرورگر اینترنت استفاده نمایید. برای این منظور علاوه بر کتابخانه GD لازم است تا کتابخانه دیگری با عنوان FreeType را نیز بر روی کامپیوترتان نصب کنید. با دراختیار داشتن این دو کتابخانه می‌توانید اقدام به ایجاد نمودارهای گرافیکی نموده و از عناصر گرافیکی به‌گونه مؤثرتری در برنامه‌های خود استفاده کنید. PHP در این راستا ابزار بسیار کارآمدی را در اختیارتان قرار می‌دهد که با بهره‌گیری از آن می‌توانید از وجود فضای کافی جهت نوشتن متون بر روی تصاویر مورد نظرتان اطمینان حاصل کنید.

درج یک دنباله کاراکتری با استفاده از تابع ( ) imageTTFtext

با استفاده از تابع ( ) imageTTFtext می‌توانید متون موردنظرتان را بر روی تصاویر گرافیکی موجود برروی صفحه مرورگر اینترنت درج نمایید. این تابع جهت انجام عملیات مورد نیاز خود به هشت آرگومان نیاز دارد. اولین آرگومان آن طبق انتظار مرجعی به یک تصویر است. دومین آرگومان اندازه ارتفاع کاراکترهایی را که با استفاده از این تابع بر روی تصویر موردنظر حک می‌شوند، مشخص می‌کند. آرگومان بعدی عدد صحیحی است که زاویه نوشتن را تعیین می‌کند. دو آرگومان چهارم و پنجم مختصات نقطه شروع نوشتن را تعیین می‌کند (آرگومان چهارم موقعیت نقطه شروع را در راستای افقی وآرگومان‌پنجم‌در راستای عمودی مشخص می‌کند). آرگومان بعدی مرجعی است که رنگ مورد استفاده جهت‌نوشتن‌متن را تعیین می‌کند. آرگومان هفتم مسیر فایل حاوی فونت TrueType موردنظر را برروی

کامپیوتر مشخص می‌کند و بالاخره آرگومان نهایی متن مورد نظر جهت نوشتن بر روی تصویر را تعیین می‌نماید.

نقطه شروع نوشتن متن (آرگومانهای چهارم و پنجم در تابع imageTTFtext) در حقیقت موقعیت خط زمینه اولین کاراکتر از متن موردنظر را مشخص می‌کند.

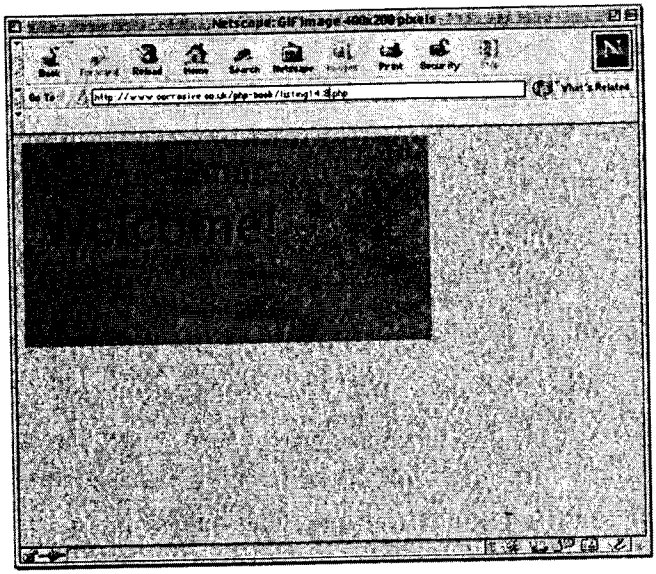
برنامه موجود در لیست ۸- ۱۴، متن ساده‌ای را بر روی یک تصویر (یک چهارضلعی) درج کرده و نتیجه حاصل را به‌عنوان خروجی بر روی صفحه مرورگر نمایش می‌دهد.

```
 1: <?php
 2: header("Content-type: image/gif");
 3:
 4: $image = imagecreate( 400, 200 );
 5: $red = imagecolorallocate($image, 255,0,0);
 6: $blue = imagecolorallocate($image, 0,0,255 );
 7: $font = "/usr/local/jdk121_pre-v1/jre/lib/fonts/LucidaSansRegular.ttf";
 8:
 9: imageTTFtext( $image, 50, 0, 20, 100, $blue, $font, "Welcome!" );
10:
11: imagegif($image);
12: ?>
```

لیست ۸- ۱۴ درج متن ساده‌ای بر روی یک تصویر با استفاده از تابع ( ) imageTTFtext

همان‌گونه که مشاهده می‌کنید، ابتدا در خط ۴ از برنامه با استفاده از تابع ( ) imageTTFtext یک چهارضلعی به ابعاد ۴۰۰ پیکسل در ۲۰۰ پیکسل ایجاد شده‌است. در خطوط ۵ و ۶ از برنامه نیز دو مرجع جهت استفاده از دو رنگ قرمز و آبی ایجاد شده است. درخط ۷ مسیر فونت TrueType موردنظر در کامپیوتر در متغیری با نام $font ذخیره شده است. توجه کنید که فایل‌های مربوط به فونت‌ها ممکن است در فهرستهای مختلفی از سیستم شما نصب شده باشند. اگر در این مورد اطمینان ندارید، به‌سادگی می‌توانید فایل‌های با پسوند ttf. را در سیستم فایل کامپیوترتان مورد جستجو قراردهید (فایل‌های با پسوند فوق، فونت‌های TrueType را مشخص می‌کنند). خط ۹ از برنامه با به‌کارگیری تابع ( ) imageTTFtext دنباله کاراکتری " ! Welcome " را برروی تصویر موجود حک می‌کند.

چنانچه در فراخوانی تابع ( ) imageTTFtext درخط ۹ از این برنامه مشاهده می‌کنید، ما از اندازه 50، زاویه صفر و موقعیت شروع 20 پیکسل در راستای افقی و 100 پیکسل در راستای عمودی استفاده کرده‌ایم؛ همچنین مرجع رنگ موردنظر را که در متغیر $blue و مسیر فایل مربوط به فونت موردنظر را که در متغیر $font ذخیره شده است را به‌عنوان آرگومانهای ششم و هفتم به این تابع ارسال کرده‌ایم و در نهایت متن موردنظر خود، به‌عنوان آخرین آرگومان مشخص نموده‌ایم. خروجی حاصل از اجرای این برنامه را می‌توانید در شکل ۸- ۱۴ مشاهده کنید.



شکل ۸- ۱۴ نوشتن متن موردنظر برروی یک تصویر با استفاده از تابع ( ) imageTTFtext

باوجود تمامی امکانات پیش‌بینی شده در تابع ( ) imageTTFtext، همان‌گونه که احتمالاً تابه حال‌برای‌شماسؤال‌برانگیز شده است مجبوریم موقعیت متن موردنظرمان بر روی تصویر را به‌طور حدسی مشخص کنیم؛ چراکه آرگومان دوم مربوط به تعیین فونت ارتفاع دقیق کاراکترها را مشخص نمی‌کند. ضمن اینکه طول متن نیز خود معضل دیگری است. اما خبر خوشایند این است که تابع ( ) imageTTFtext، اطلاعاتی را درمورد ابعاد متن به برنامه فراخواننده بازمی‌گرداند. هرچند که این اطلاعات می‌تواند مفید واقع شود، ما در اینجا آنها را نادیده گرفته و به آنچه که داریم، قناعت می‌کنیم. درعوض تابع دیگری را در قسمت بعد بررسی می‌کنیم که امکانات و اطلاعات مفیدی را پیش از نوشتن متن موردنظر بر روی تصویر موجود بر روی صفحه در اختیارمان قرار می‌دهد.

بررسی ابعاد متن با استفاده از تابع ( ) imageTTFbox

با استفاده از تابع ( ) imageTTFbox می‌توانیم به اطلاعات مفید و مهمی درباره ابعاد متن موردنظرمان دست پیدا کنیم. این تابع از آن جهت بدین گونه نامیده شده است که اطلاعاتی را در مورد ابعاد کادری که متن در درون آن جای می‌گیرد، در اختیار ما قرار می‌دهد. تابع ( ) imageTTFbox جهت‌انجام عمیات خود به چهار آرگومان ضروری نیاز دارد. آرگومان اول اندازه فونت موردنظر، آرگومان دوم زاویه نوشتن متن، آرگومان سوم مسیر فایل مربوط به فونت مورد استفاده در سیستم فایل کامپیوتر و بالاخره آرگومان چهارم متن مورد نظر را مشخص می‌کنند. این تابع همان‌گونه که متوجه شده‌ایدیکی از معدود توابعی است که مرجع یک تصویر را به‌عنوان آرگومان مورد استفاده قرار نمی‌دهد.

حاصل عمیات این تابع یک آرایه هشت عضوی است که توصیف هریک از آنها در جدول ۱-۱۴ آمده است.

جدول ۱- ۱۴ عناصر آرایه بازگشتی از تابع ( ) imageTTFbox

شماره شاخص	توصیف
0	پایین سمت چپ (محور افقی)
1	پایین سمت چپ (محور عمودی)
2	پایین سمت راست (محور افقی)
3	پایین سمت راست (محور عمودی)
4	بالا سمت راست (محور افقی)
5	بالا سمت راست (محور عمودی)
6	بالا سمت چپ (محور افقی)
7	بالا سمت چپ (محور عمودی)

ابعاد متن موردنظر در راستای عمودی نسبت به خط زمینه متن که مبدأ سنجش بوده و مقدار صفر به آن نسبت داده می‌شود، سنجیده می‌گردد؛ بدین ترتیب که بخش بالای خط زمینه یک متن نمونه از بالا به پایین (تا خط زمینه یا همان مبدأ سنجش) مورد شمارش قرار می‌گیرد و بنابراین معمولاً مقداری منفی هستند. همچنین بخش پایین خط زمینه متن از پایین زمینه متن از پایین به بالا (باز هم تا خط زمینه) مورد شمارش قرار می‌گیرد و بنابراین مقداری مثبت می‌باشند (به‌عبارت بهتر بخش بالای خط زمینه "تا" صفر شمارش می‌شود ولی شمارش بخش پایین آن " از" صفر آغاز می‌شود).

بنابراین به‌عنوان مثال ارزیابی متنی که در آن حرف "y" استفاده شده است، آرایه‌ای هشت عضوی از اعداد صحیح را به دست می‌دهد که دومین عنصر آن (دومین سطر از جدول ۱- ۱۴) حاوی مقدار 3 خواهد بود؛ چراکه بخش پایینی حرف y به اندازه سه پیکسل پایین خط زمینه واقع می‌شود. همچنین آخرین عضو این آرایه (آخرین سطر از جدول ۱- ۱۴) حاوی مقدار 10 - خواهد بود و زیر بخش فوقانی کاراکتر مورد بحث به اندازه 10 پیکسل بالای خط زمینه متن واقع می‌گردد.

با این حال در ارزیابی واقعی چنین به‌نظر می‌رسد که یک اختلاف جزئی به اندازه دو پیکسل مابین فاصله‌ای که تابع imageTTFbox ( ) در قالب عناصر یک آرایه باز می‌گرداند با فاصله چشمی

مابین سطح بالا و پایین کاراکتر موردنظر تا خط زمینه وجود داشته باشد. شاید لازم باشد تا در برخی موارد حساس این فاصله را نیز با درنظرگرفتن این مطلب که ارتفاع خط زمینه به اندازه دو پیکسل بیش از آن‌چیزی است که توسط تابع imageTTFbox ( ) در قالب یک آرایه مشخص می‌شود، در محاسبات خود منظور نمایید.

اما درمورد ابعاد مربوط به سمت چپ محور مختصات در راستای افقی (یعنی عناصر اول و هفتم از آرایه بازگشتی توسط تابع imageTTFbox ( ) یا به عبارت دیگر، سطرهای اول و هفتم جدول ۱-۱۴) وضعیت به‌مانند ابعاد مربوط به سمت چپ محور مختصات در راستای عمودی است؛ چراکه در اینجا نیز شمارش از نقطه شروع کاراکتر مورد نظر آغاز شده و تا رسیدن به مبدأ صفر ادامه می‌یابد از این جهت مقادیر مربوط به این ابعاد یک عدد منفی خواهد بود. از آنجا که این عدد منفی اغلب کوچک‌تر از آن‌است که بسیار جدی تلقی شود، تصمیم اینکه چگونه آن‌را در محاسبات خود دخیل می‌کنید با خود شماست.

از اطلاعات بازگشتی توسط تابع imageTTFbox ( ) که یک آرایه عددی هشت عضوی بیش نیست، می‌توانید جهت تنظیم فواصل متن موجود در یک تصویر از پیرامون آن تصویر استفاده نمایید. برنامه موجود در لیست ۹- ۱۴ یک متن ساده را که تنظیمات فواصل آن از پیرامون تصویر زمینه به‌طور خودکار انجام می‌شود، نشان می‌دهد. این متن به‌سادگی در مرکز تصویر زمینه خود واقع می‌شود.

```
1: <?php
2: header("Content-type: image/gif");
3: $height = 100;
4: $width = 200;
5: $fontsize = 50;
6: if ( ! isset ( $text ) )
7:     $text = "Change me!";
8: $image = imagecreate( $width, $height );
9: $red = imagecolorallocate($image, 255,0,0);
10: $blue = imagecolorallocate($image, 0,0,255 );
11: $font = "/home/usr/local/jdk1.3.1/jre/lib/fonts/LucidaSansRegular.ttf";
12: $textwidth = $width;
13: $textheight;
14: while ( true ) {
15:     $box = imageTTFbbox( $fontsize, 0, $font, $text );
16:     $textwidth =  abs( $box[2] );
17:     $textbodyheight = ( abs($box[7]) )-2;
18:     if ( $textwidth < $width - 20 )
19:         break;
20:     $fontsize--;
21: }
22: $gifXcenter = (int) ( $width/2 );
23: $gifYcenter = (int) ( $height/2 );
24: imageTTFtext(    $image, $fontsize, 0,
```

لیست ۹- ۱۴ تنظیم متن موجود بر روی تصویر با استفاده از تابع imageTTFbox ( )

```
25:         (int) ($gifXcenter-($textwidth/2)),
26:         (int) ($gifYcenter+(($textbodyheight)/2) ),
27:         $blue, $font, $text );
28: imagegif($image);
29: ?>
```

دنباله لیست ۹- ۱۴

همان‌گونه که در این برنامه مشاهده می‌کنید ما متغیرهای $height و $width را درخطوط ۳ و ۴ جهت ذخیره ارتفاع و پهنای تصویر مورد استفاده قرار داده و در خط ۵ نیز اندازه فونت مورد استفاده را برابر با ۵۰ واحد انتخاب کرده‌ایم. در خط ۶ بررسی را در مورد وجود متغیری با نام $text صورت داده و درصورتی که این متغیر موجود نباشد، در خط ۷ آن‌را با یک مقدار پیش‌فرض تعریف کرده‌ایم. به این روش می‌توانیم ترتیبی دهیم تا تصویر مورد استفاده بتواند داده‌هایی را از یک صفحه وب بپذیرد (از طریق دنباله پرس و جوی ضمیمه شده به آدرس URL تصویر یا از طریق فرم ارسالی به سرور). در خط ۸ جهت دستیابی به یک مرجع تصویر از تابع ( ) imagecreate استفاده کرده‌ایم. همچنین در خطوط ۹ تا ۱۱ برنامه طبق روش معمول جهت دستیابی به مرجعی برای استفاده از رنگ و نیز ثبت مسیر فونت TrueType در متغیری با نام $font اقدام شده است.

تمایل داریم تا دنباله کاراکتری ذخیره شده در متغیر $text را درفضایی که در اختیار داریم، جای‌دهیم؛ اماهیچ روشی جهت اطلاع از اینکه آیا چنین چیزی اصلاً امکان‌پذیر است یا خیر در دسترس قرار ندارد. لذا باید این مشکل را به‌گونه‌ای ابتکاری حل و فصل نماییم. همان‌گونه که مشاهده می‌کنید، در ساختار تکرار while که از خط ۱۴ تا ۲۱ ادامه دارد مسیر فایل مربوط به فونت موردنظر و همچنین متن موردنیاز خود را به تابع ( ) imageTTFbox ارسال کرده (خط ۱۵) و آرایه حاصل از عملیات این تابع را جهت مراجعات بعدی در متغیری با نام $box نگه داشته‌ایم. عنصر سومین از این آرایه که اکنون به صورت $box[2] آن‌را مورد دستیابی قرار می‌دهیم شامل موقعیت گوشه پایین و سمت راست از متن بر روی محور افقی می‌باشد. این مقدار را به‌عنوان طول دنباله کاراکتری جهت مراجعات بعدی در متغیری با نام $textwidth ذخیره می‌کنیم (خط ۱۶).

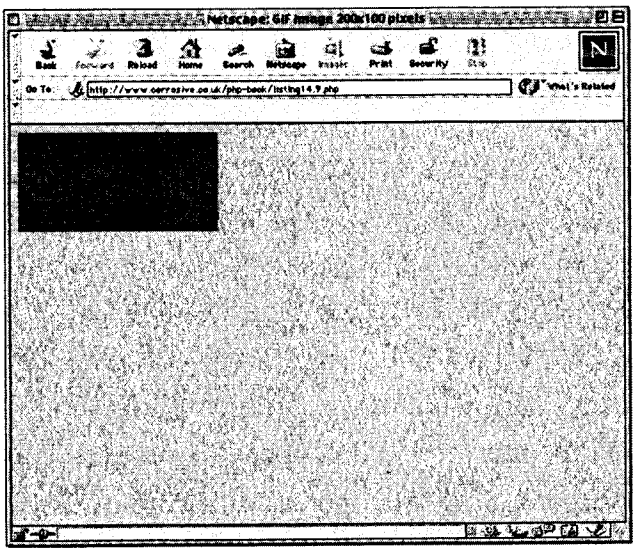
همچنین مایلیم تا متن را در وسط تصویر نسبت به خط عمود درج کنیم با این توضیح که تنها بخش موجود در بالای خط زمینه را درنظر می‌گیریم. برای این‌کار ابتدا مقدار قدر مطلق عضو $box[7] از آرایه $box را جهت پی بردن به ارتفاع متن موجود در بالای خط زمینه محاسبه کرده و جهت تنظیم بهتر دو واحد از حاصل به‌دست آمده کم می‌کنیم (علت این اختلاف را پیشتر توضیح دادیم). آن چه که به‌دست می‌آید در متغیر $textbodyheight ذخیره می‌کنیم.

اکنون با در دست داشتن طول متن می‌توانیم آن‌را با طول تصویر زمینه مقایسه کنیم (البته با درنظرگرفتن حاشیه تصویر که اندازه‌ای برابر با ۱۰ پیکسل دارد). چنانچه متن مورد نظر کوچک‌تر از طول تصویر باشد، با استفاده از دستورالعمل break درخط ۱۹ به اجرای عملیات حلقه پایان می‌دهیم.

درغیر این‌صورت اندازه فونت مورد استفاده را توسط دستورالعمل خط ۲۰ کاهش داده و عملیات را مجدداً تکرار می‌کنیم.

با تقسیم مقادیر متغیرهای $height و $width بر عدد ۲ (خطوط ۲۲ و ۲۳ برنامه) توانستیم موقعیت تقریبی وسط تصویر را به‌دست آوریم. همان‌گونه که مشاهده می‌کنید با محاسبه ابعاد متن موردنظر و موقعیت مرکز تصویر و همچنین ارتفاع و طول آن در خط ۲۴ با فراخوانی تابع imageTTFtext ( ) توانستیم متن مذکور را با موفقیت بر روی تصویر زمینه درج نماییم.

در نهایت توسط فراخوانی تابع imagegif ( ) در خط ۲۸ از برنامه تصویر حاوی متن را بر روی صفحه مرورگر اینترنت ارسال کردیم. خروجی حاصل از این برنامه در شکل ۹- ۱۴ قابل مشاهده است.



شکل ۹- ۱۴ بهره‌گیری از تابع imageTTFbox ( ) جهت تعیین ابعاد متن

اکنون به‌راحتی می‌توانیم برنامه موجود در لیست ۹- ۱۴ را از درون یک صفحه دیگر در قالب نشانه < img > مورد بهره‌گیری قرار دهیم. لیست ۱۰- ۱۴، نحوه انجام این عمل را نشان می‌دهد. این برنامه امکان واردکردن متن را جهت درج بر روی تصویر از طریق یک فرم HTML در اختیار کاربر قرار می‌دهد:

```
1: <?php
2: header("Content-type: image/gif");
3: $cells = array ( 'liked'=>200, 'hated'=>400, 'indifferent'=>900 );
4: $max = max( $cells );
5: $total = count ( $cells );
6: $totalwidth = 300;
```

لیست ۱۰- ۱۴ بهره‌گیری از برنامه لیست ۹- ۱۴ در یک برنامه دیگر

```
 7: $totalheight = 200;
 8: $xgutter = 20; // left/right margin
 9: $ygutter = 20; // top/bottom margin
10: $internalgap = 10; // space between cells
11: $bottomspace = 30; // gap at the bottom (in addition to margin)
12: $font = "/home/usr/local/jdk1.3.1/jre/lib/fonts/LucidaSansRegular.ttf";
13: $graphCanX = ( $totalwidth - $xgutter*2 );
14: $graphCanY = ( $totalheight - $ygutter*2 - $bottomspace );// starting draw
➥position x - axis
15: $posX = $xgutter; // starting draw pos - y - axis
16: $posY = $totalheight - $ygutter - $bottomspace;
17: $cellwidth = (int) (( $graphCanX -                    ( $internalgap * ( $total-1
➥) )) / $total) ;
18: $textsize = (int)($bottomspace);
19: // adjust font size
20: foreach ( $cells as $key=>$val ) {
21:     while ( true ) {
22:         $box = ImageTTFbBox( $textsize, 0, $font, $key );
23:         $textWidth = abs( $box[2] );
24:         if ( $textWidth < $cellwidth )
25:             break;
26:         $textsize--;
27:     }
28: }
29: $image = imagecreate( $totalwidth, $totalheight );
30: $red = ImageColorAllocate($image, 255, 0, 0);
31: $blue = ImageColorAllocate($image, 0, 0, 255 );
32: $black = ImageColorAllocate($image, 0, 0, 0 );
33: $grey = ImageColorAllocate($image, 100, 100, 100 );
34: foreach ( $cells as $key=>$val ) {
35:     $cellheight = (int) (($val/$max) * $graphCanY);
36:     $center = (int)($posX+($cellwidth/2));
37:     imagefilledrectangle( $image, $posX, ($posY-$cellheight),
➥($posX+$cellwidth), $posY, $blue );
38:     $box = ImageTTFbBox( $textsize, 0, $font, $key );
39:     $tw = $box[2];
40:     ImageTTFText( $image, $textsize, 0, ($center-($tw/2)),
41:             ($totalheight-$ygutter), $black, $font, $key );
42:     $posX += ( $cellwidth + $internalgap);
43: }
44: imagegif( $image );
45: ?>
```

دنباله لیست ۱۰-۱۴

همان‌گونه که در این لیست مشاهده می‌کنید، هنگام فراخوانی برنامه لیست ۹-۱۴ در خط ۹ یک دنباله پرس و جو را که حاوی متن موردنظر کاربر است به آن ضمیمه کرده‌ایم. در ساعت نوزدهم با عنوان " ثبت وضعیت با استفاده از کوکی‌ها و دنباله‌های پرس و جو" به بررسی این تکنیک جهت ارسال اطلاعات از یک برنامه به برنامه دیگر خواهیم پرداخت.

بررسی یک مثال کامل

اجازه دهید تا در این قسمت از درس به بررسی مثالی بپردازیم که برخی از توابع موردبحث در این ساعت را جهت انجام عملیات خود به کار می‌گیرد. فرض کنید از ما خواسته شده است تا یک نمودار میله‌ای پویا جهت مقایسه محدوده‌ای از اعداد مشخص ایجاد نماییم. نمودار میله‌ای حاصل باید آن‌چنان باشد که برچسب مربوط به هر میله در پایین آن (بر روی محور افقی) واقع شود. کاربر باید در این میان قادر باشد تا تعداد میله‌های موجود در نمودار را به‌دلخواه تغییر دهد. همچنین باید بتواند عرض ارتفاع میله‌ها و اندازه حاشیه پیرامون نمودار را به‌دلخواه تعیین نماید. این نمودار میله‌ای جهت ارزیابی آرای مشتریان مورد استفاده قرار می‌گیرد. با این حال به جزئیات آرا علاقه‌مند نبوده بلکه آنچه مورد نظرمان است یک دید کلی از آن است تا بتوانیم به سرعت مابین موارد موجود مقایسه مورد نظرمان را صورت دهیم. درصورت نیاز می‌توانیم توضیحات بیشتر در مورد جزئیات را در درون سند HTML، یعنی جایی که نمودار واقع می‌شود، جای دهیم.

ساده‌ترین روشی که در اینجا برای ذخیره برچسبها و مقادیر مربوطه به ذهن می‌رسد، استفاده از یک آرایه انجمنی است. پس از ایجاد این آرایه انجمنی لازم است تا تعداد میله‌های موجود در نمودار و همچنین بزرگ‌ترین مقدار موجود در آرایه را محاسبه نماییم. لیست کوتاه ۱۱- ۱۴ شامل کد لازم جهت انجام این عملیات را نشان می‌دهد.

```
$cells = array( 'linked' => 200, 'hated' => 400,
'indifferent' => 900 );
$max = max( $cells );
$total= count( $cells );
```

لیست ۱۱- ۱۴ کد لازم جهت محاسبه تعداد میله‌های نمودار و بزرگ‌ترین مقدار موجود در آرایه انجمنی مربوطه

گام بعدی این است که به معرفی متغیرهایی بپردازیم که کاربر از طریق آنها بتواند تصویر موجود بر روی صفحه شامل فونت مورد استفاده و اندازه حاشیه‌ها را به‌دلخواه مشخص نماید. لیست ۱۲ - ۱۴ متغیرهای لازم جهت این کار را به‌همراه مقادیر اولیه آنها نشان می‌دهد.

```
$totalwidth = 400;
$totalheight = 200;
$xgutter = 20;
$ygutter = 20;          // left/right margin
$internalgap = 5; // space between cells
$bottomspace = 40;      / gap at the bottom( in addition to
margin )
$font =
"/home/usr/local/jdk1.3.1/jre/lib/fonts/LucidaSansRegular.ttf
";
```

لیست ۱۲- ۱۴ متغیرهای مورد نیاز جهت تنظیم دلخواه تصویر بر روی صفحه مرورگر اینترنت

به واسطه متغیرهای معرفی شده در این لیست، کاربر می‌تواند به‌راحتی اندازه ارتفاع و طول تصویر را مشخص کند. متغیرهای $xgutter و $ygutter به ترتیب حاشیه‌های افقی و عمودی پیرامون تصویر را مشخص می‌نماید. متغیر $internalgap فضای مابین میله‌های نمودار را مشخص کرده و متغیر $bottomspace نیز شامل اندازه فضای قابل استفاده جهت درج برچسب هریک از میله‌هاست.

اکنون با در دست داشتن این مقادیر می‌توانیم محاسبات دیگری را جهت تعیین مقادیر سایر متغیرها انجام دهیم. لیست ۱۳- ۱۴ این محاسبات را نشان می‌دهد.

```
$graphCanX = ( $totalwidth - $xgutter * 2 );
$graphCanY = ( $totalheight - $ygutter * 2 - $bottomspace );
$posX = $xgutter; // starting draw position x - axis
$posY = $totalheight - $ygutter - $bottomspace;     //
starting draw pos - y - axis
$cellwidth = ( int )( ( $graphCanX - ( $internalgap * (
$total - 1 ) ) ) / $total );
$textsize = ( int )( $bottomspace );
```

لیست ۱۳- ۱۴ محاسبات مربوط به ابعاد نمودار

همان‌گونه که مشاهده می‌کنید، بخشی از این لیست مربوط به محاسبه زمینه نمودار (یعنی فضایی که میله‌های نمودار در آنجا به نمایش در می‌آیند) می‌شود. طول این زمینه در راستای محور افقی برابر با طول کل تصویر منهای دوبرابر اندازه حاشیه است. همچنین ارتفاع آن در راستای محور عمودی برابر با ارتفاع کل تصویر منهای دوبرابر اندازه حاشیه است که البته باید مقدار متغیر $bottomspace نیز جهت پیش بینی فضایی برای درج برچسب میله‌های نمودار از حاصل آن کم شود. متغیر $posX موقعیتی از محور افقی را که فرآیند رسم میله‌های نمودار از آنجا آغاز می‌شود، تعیین می‌کند. از این‌رو مقدار این متغیر برابر با متغیر $xgutter درنظر گرفته شده است (این مقدار شامل فضای مربوط به حاشیه تصویر در راستای افقی نیز می‌شود). متغیر $posY موقعیت پایین میله‌های نمودار را مشخص می‌کند. همان‌گونه که از محاسبات نیز پیداست، مقدار این متغیر برابر است با اندازه کل ارتفاع تصویر که اندازه یک حاشیه و فضای مربوط به درج برچسبها (مقدار موجود در متغیر $bottomheight) از آن کم شده باشد.

متغیر $cellwidth شامل اندازه عرض هر میله از نمودار است. جهت تعیین مقدار این متغیر ابتدا فضای کل مابین میله‌ها محاسبه شده و حاصل آن از عرض کل تصویر کم می‌شود. از تقسیم نتیجه به دست آمده بر تعداد میله‌های نمودار عرض یک میله به دست می‌آید.

همان‌گونه که ملاحظه می‌کنید اندازه ابتدایی فونت مورد استفاده در این نمودار برابر با ارتفاع فضای پیش‌بینی شده برای درج برچسب میله‌ها فرض شده است (این مقدار در متغیر $bottomspace ذخیره شده است).

پیش از ایجاد تصویر و کار بر روی آن ابتدا لازم است تا اندازه ابعاد متن موردنظر را با استفاده از آرایه بازگشتی از تابع ( ) imageTTFbox تعیین کنیم. مشکل اینجاست که طول برچسبها را نمی‌دانیم، در حالی‌که مایلیم این برچسبها دقیقاً در فضایی که در پایین میله‌های نمودار برای هریک درنظر گرفتیم، جای بگیرند. برای حل این مشکل مقادیر موجود در آرایه انجمنی حاوی برچسبها را که در متغیری با نام $cells ذخیره کرده‌ایم جهت محاسبه بزرگ‌ترین اندازه‌ای که می‌توانیم برای فونت متن درنظر بگیریم در قالب یک ساختار تکرار مورد پردازش و بررسی قرارمی‌دهیم. لیست ۱۴- ۱۴ کد مربوط به چنین فرآیندی را نشان می‌دهد.

```
foreach( $cells as $key => $val ) {
      while( true ) {
            $box = ImageTTFbBox( $textsize, 0, $font, $key );
            $textWidth = $box[ 2 ];

            if( $textWidth < $cellwidth )
                  break;

            $textsise--;
      }
}
```

لیست ۱۴- ۱۴ کد مربوط به تعیین بزرگ‌ترین اندازه قابل استفاده برای درج متن در نمودار میله‌ای

همان‌گونه که در لیست فوق نیز مشاهده می‌کنید، به‌ازای هریک از عناصر موجود در آرایه پردازشی جهت تعیین ابعاد متن موردنظر صورت می‌پذیرد. طی این پردازش اطلاعات مربوط به ابعاد فعلی هر یک از برچسبها به عنوان آرگومان به تابعی که توانایی ارزیابی آن را دارد، یعنی تابع imageTTFbox ( ) ارسال می‌گردد. درهربار عبور از حلقه، سومین عنصر از آرایه بازگشتی از تابع مذکور که با عنوان $box[2] مشخص شده با مقدار موجود در متغیر $cellwidth مقایسه می‌شود (این متغیر شامل عرض یک میله نمودار است). چنانچه برچسب مورد بررسی اندازه‌ای کوچک‌تر از مقدار متغیر فوق داشته باشد، با استفاده از دستورالعمل break به عملیات حلقه پایان داده می‌شود. درغیر این‌صورت با استفاده از عملگر کاهش یک واحد از مقدار متغیر $textsize کم شده و عملیات حلقه از سرگرفته می‌شود. فرآیند کاهش و به‌دنبال آن بررسی تا زمانی ادامه پیدا می‌کند که تمام برچسبهای موجود در آرایه $cells در محل تعیین شده در پایین میله‌های نمودار جای بگیرند.

گام نهایی ایجاد یک مرجع به تصویر مورد نظر و بهره‌گیری از توابع بررسی شده در این درس جهت تکمیل فرآیند است.

لیست ۱۵- ۱۴ کد باقیمانده جهت این‌کار را نشان می‌دهد.

```
$image = imagecreate( $totalwidth, $totalheight );
$red = ImageColorAllocate( $image, 255, 0, 0 );
blue = ImageColorAllocate( $image, 0, 0, 255 );
black = ImageColorAllocate( $image, 0, 0, 0 );
reset( $cells );

foreach( $cells as $key => $val ) {
      $cellheight = ( int )( ( $val / $max ) * $graphCanY );
      $center = ( int )($posX + ( $cellwidth / 2 ) );
      imagefilledrectangle( $image, $posX, ( $posY -
$cellheight ),
            ( $posX + $cellwidth ), $posY, $blue );
      $box = ImageTTFbBox( $textsize, 0, $font, $key );
      $tw = $box[ 2 ];
      ImageTTFText( $image, $textsize, 0, ( $center - ( $tw /
2 ) ),
            ( $totalheight - $ygutter ), $black, $font, $key
);
      $posX += ( $cellwidth + $internalgap );
}

imagegif( $image );
```

لیست ۱۵- ۱۴ استفاده از توابع پردازش تصویر جهت تکمیل فرآیند

همان‌گونه که مشاهده می‌کنید این لیست کار خود را با ایجاد یک مرجع به تصویر مورد استفاده آغاز می‌کند. چنانکه می‌دانید، این فرآیند با استفاده از تابع ( ) imagecreate انجام می‌گیرد. پس از آن سه مرجع مختلف با استفاده از تابع ( ) imagecolorallocate جهت استفاده در سایر توابع ایجاد می‌شود. باردیگر از یک ساختار تکرار جهت پردازش عناصر آرایه $cells استفاده شده است. درهربار عبور از حلقه ارتفاع یک میله از نمودار محاسبه شده و حاصل در متغیری با نام $cellheight ذخیره می‌شود. همچنین مرکز موقعیت میله‌ها در راستای محور افقی نیز با اضافه کردن مقدار متغیر $posX به نصف عرض میله محاسبه می‌گردد.

در هربار عبور از حلقه با استفاده از تابع ( ) imageFilledectangle و بهره‌گیری از متغیرهای $posX ، $posY ، $cellheight و $cellwidth به‌عنوان آرگومانهای تابع فوق، تصویر یک میله بر روی زمینه ترسیم می‌شود.

باردیگر به‌منظور تنظیم متن از تابع ( ) imageTTFbox استفاده شده است. این مقدار بازگشتی از این تابع که یک آرایه عددی است در متغیری با نام $box ذخیره می‌شود. درهرگذر از حلقه، مقدار سومین عنصر از این آرایه که با عنوان $box [2] مورد دستیابی قرار می‌گیرد در متغیری با نام $tw جهت مراجعات بعدی نگهداری می‌شود. هم‌اکنون اطلاعات کافی برای درج یک برچسب در پایین میله مربوطه از نمودار موجود می‌باشد. موقعیت محل درج در راستای افقی با تفریق نصف طول متن از مقدار

ذخیره شده در متغیر $center به دست می‌آید. همچنین موقعیت محل درج در راستای عمودی را می‌توان با کم کردن اندازه حاشیه از اندازه ارتفاع تصویر به‌راحتی به دست آورد.

در انتهای حلقه مقدار متغیر $bosX جهت تکرار فرآیند در مورد میله بعدی محاسبه می‌شود.

در انتهای برنامه، تابع ( ) imagegif جهت درج تصویر نهایی بر روی صفحه مرورگر اینترنت فراخوانی می‌شود.

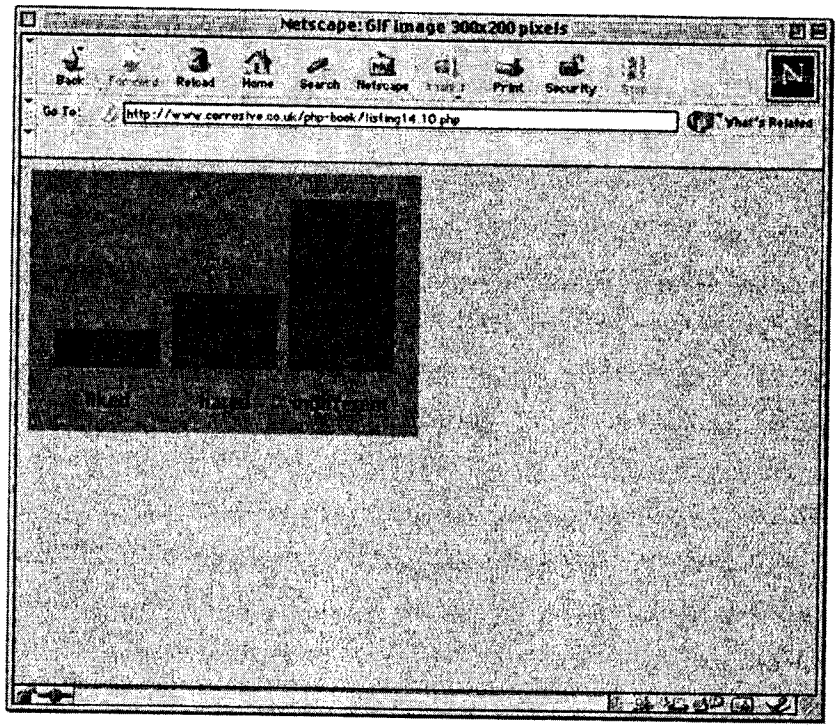
برنامه کامل این فرآیند در لیست ۱۶- ۱۴ و خروجی حاصل از اجرای آن نیز در شکل ۱۰- ۱۴ قابل توجه و بررسی است.

```
<?php
header("Content-type: image/gif");
$cells = array ( 'liked'=>200, 'hated'=>400,
'indifferent'=>900 );
$max = max( $cells );
$total = count ( $cells );
$totalwidth = 300;
$totalheight = 200;
$xgutter = 20; // left/right margin
$ygutter = 20; // top/bottom margin
$internalgap = 10; // space between cells
$bottomspace = 30; // gap at the bottom (in addition to
margin)
$font =
"/home/usr/local/jdk1.3.1/jre/lib/fonts/LucidaSansRegular.ttf
";
$graphCanX = ( $totalwidth - $xgutter*2 );
$graphCanY = ( $totalheight - $ygutter*2 - $bottomspace );//
starting draw position x - axis
$posX = $xgutter; // starting draw pos - y - axis
$posY = $totalheight - $ygutter - $bottomspace;
$cellwidth = (int) (( $graphCanX -                (
$internalgap * ( $total-1 ) )) / $total) ;
$textsize = (int)($bottomspace);
// adjust font size
foreach ( $cells as $key=>$val ) {
    while ( true ) {
        $box = ImageTTFbBox( $textsize, 0, $font, $key );
        $textWidth = abs( $box[2] );
        if ( $textWidth < $cellwidth )
            break;
        $textsize--;
    }
}
$image = imagecreate( $totalwidth, $totalheight );
```

لیست ۱۶- ۱۴ برنامه کامل رسم یک نمودار میله‌ای

```
$red = ImageColorAllocate($image, 255, 0, 0);
$blue = ImageColorAllocate($image, 0, 0, 255 );
$black = ImageColorAllocate($image, 0, 0, 0 );
$grey = ImageColorAllocate($image, 100, 100, 100 );
foreach ( $cells as $key=>$val ) {
    $cellheight = (int) (($val/$max) * $graphCanY);
    $center = (int)($posX+($cellwidth/2));
    imagefilledrectangle( $image, $posX, ($posY-$cellheight),
($posX+$cellwidth), $posY, $blue );
    $box = ImageTTFbBox( $textsize, 0, $font, $key );
    $tw = $box[2];
    ImageTTFText( $image, $textsize, 0, ($center-($tw/2)),
            ($totalheight-$ygutter), $black, $font, $key );
    $posX += ( $cellwidth + $internalgap);
}
imagegif( $image );
?>
```

دنباله لیست ۱۶-۱۴ برنامه کامل رسم یک نمودار میله‌ای



شکل ۱۰-۱۴ خروجی حاصل از برنامه رسم نمودار

توجه کنید که در فاصله خطوط ۳ تا ۱۸ از این برنامه متغیرهایی را که مورد استفاده قرار

خواهیم داد، معرفی و مقداردهی کرده‌ایم. ساختار تکرار موجود بین خطوط ۲۰ تا ۲۸ فضای مورد نیاز جهت استفاده از متن را به‌خوبی تعیین می‌کند. درنهایت کد موجود در خطوط ۲۹ تا ۴۴ نمودار موردنظر را شکل می‌دهد.

جمع‌بندی

با پشتیبانی به‌عمل آمده از کتابخانه کدباز GD در زبان برنامه‌نویسی PHP به‌راحتی می‌توانید نمودارهای قابل توجهی ایجاد کرده و از قابلیتهای گرافیکی موجود جهت ایجاد برنامه‌های جذاب‌تر استفاده نمایید.

در درس این ساعت درمورد بهره‌گیری از این قابلیتها و نحوه استفاده از توابع مربوطه بحث کردیم. در این درس با چگونگی بهره‌برداری از توابع ( ) imagecreate و ( ) imagegif جهت ایجاد تصویر و نمایش آن در خروجی آشنا شدید. چگونگی استفاده از تابع ( ) imagecolorallocate به‌منظور دستیابی به مرجع رنگ موردنظر و نحوه به‌کارگیری این مرجع در تابع ( ) imagefill جهت رنگ آمیزی تصاویر یا بخشهایی از آنها مطالب دیگری بود که در این درس با آنها آشنا شدید، همچنین در درس این ساعت مطالب مفیدی درمورد چگونگی استفاده از توابع مربوط به ترسیم خطوط و اشکال مختلف جهت ایجاد یک طرح و نحوه رنگ آمیزی آنها فراگرفتید. علاوه بر اینها مطالبی را درباب چگونگی استفاده از کتابخانه کدباز دیگری با نام FreeType که جهت کار با فونت‌های TrueType طراحی شده و زبان برنامه‌نویسی PHP نیز به‌خوبی از آن پشتیبانی می‌کند، عنوان کردیم. با استفاده از امکانات این کتابخانه مفید همان‌گونه که مشاهده کردید، توانستیم متون موردنظرمان را بر روی تصاویر حک کنیم.

در پایان درس نیز با ارائه مثال کاملی شامل ترسیم یک نمودار میله‌ای برروی صفحه مرورگر سعی کردیم تا سودمند بودن توابع کتابخانه‌های کدباز GD و FreeType را درعمل نشان دهیم. اعتقاد ما این است که درصورت دنبال‌کردن این مطالب و بررسی مثالهای بیشتر می‌توانید مهارت خود را در ترسیم نمودارهای پیچیده‌تر افزایش دهید.

درس ساعت بعد با آنچه که در این درس مورد بررسی قرار دادیم، به کلی تفاوت دارد. درس ساعت آینده به بررسی استفاده از تاریخ و ساعت پرداخته و توابع مفیدی را که در زبان برنامه‌نویسی PHP جهت بهره‌برداری از آنها پیش‌بینی شده است، مورد بحث و گفتگو قرار می‌دهد.

پرسش و پاسخ

پرسش: آیا مسأله سرعت مانعی در بهره‌گیری از تصاویر گرافیکی پویا در صفحات و اسناد وب محسوب نمی‌شود؟

پاسخ: واضح است که ایجاد و درج تصاویر در اسناد وب به‌صورت پویا (یعنی تصاویری که از قبل آماده نشده باشند) سرعت بارگذاری را درمقایسه با اسنادی که از تصاویر گرافیکی از پیش آماده شده استفاده می‌کنند، کاهش می‌دهد. با این حال بسته به چگونگی برنامه‌نویسی، رعایت اصول مهم در این زمینه و سرعت بارگذاری مورد انتظارتان و البته میزان سادگی یا پیچیدگی تصاویر تولید شده، صفحات و اسناد حاوی این‌گونه تصاویر ممکن است تأثیر منفی فوق‌العاده‌ای بر روند بارگذاری نداشته باشند.

تمرینها

هدف از این بخش ارائه تمرینهایی در قالب آزمون است. پاسخ بخش آزمون بلافاصله پس از آن آمده است. بخش فعالیتها شامل تمرینهایی است که به قصد افزایش مهارت برنامه‌نویسی خواننده طراحی شده و البته فاقد پاسخ می‌باشد.

آزمون

۱- پیش از ارسال یک تصویر گرافیکی از نوع GIF به برنامه مرورگر کاربر ارسال کدام هدر پاسخ برای وی ضروری است؟

۲- کدام تابع جهت دستیابی به یک مرجع تصویری برای استفاده در سایر توابع گرافیکی از کتابخانه کدباز GD مورد استفاده قرار می‌گیرد؟

۳- کدام تابع تصویر ایجاد شده را به‌عنوان خروجی به صفحه مرورگر اینترنت ارسال می‌کند؟

۴- از کدام تابع می‌توان جهت دستیابی به مرجعی که نماینده یک رنگ مشخص است، استفاده کرد؟

۵- از کدام تابع می‌توان جهت ترسیم یک خط برروی یک تصویر ایجاد شده به صورت پویا استفاده نمود؟

۶- از کدام تابع می‌توان جهت رنگ آمیزی بخشی از یک تصویر پویا استفاده کرد؟

۷- کدام تابع را می‌توان جهت ترسیم کمان مورد استفاده قرار داد؟

۸- از کدام تابع می‌توان جهت ترسیم یک چهارضلعی استفاده کرد؟

۹- از کدام تابع می‌توان جهت ترسیم یک چندضلعی استفاده کرد؟

۱۰- کدام تابع را می‌توان جهت درج یک دنباله کاراکتری بر روی یک تصویر گرافیکی پویا مورد استفاده قرار داد؟ درج متون در تصاویر پویا با استفاده از امکانات کدام کتابخانه صورت می‌پذیرد؟

پاسخ آزمون

۱- پیش از ارسال یک تصویر GIF لازم است تا با استفاده از تابع ( ) header خط زیر را به برنامه مرورگر ارسال کنید:
Content _ type: image / gif

۲- با استفاده از تابع ( ) imagecreate، می‌توان مرجعی را به یک تصویر گرافیکی پویا ایجاد کرد.

۳- به‌کمک تابع ( ) imagegif، می‌توان‌تصویر پویای ایجاد شده‌را بر روی مرورگر اینترنت ارسال کرد.

۴- بابهره‌گیری از تابع ( ) imagecolorallocate، می‌توان مرجعی را به رنگ مورد نظر ایجاد نمود.

۵- به کمک تابع ( ) imageline، می‌توان یک خط راست ترسیم نمود.

۶- با استفاده از تابع ( ) imagefill، می‌توان بخشی از تصویر را رنگ آمیزی کرد.

۷- با بهره‌گیری از تابع ( ) imagearc، می‌توان کمانی را در صفحه ترسیم کرد.

۸- با استفاده از تابع ( ) imagerectangle، می‌توان اقدام به ترسیم یک چهارضلعی کرد. همچنین با بهره‌گیری از تابع ( ) imagefilledrectangle، می‌توان یک چهارضلعی رنگ آمیزی شده را بر روی صفحه ترسیم کرد.

۹- با استفاده از تابع ( ) imagepolygon، می‌توان یک چندضلعی را بر روی صفحه رسم کرد. همچنین با بهره‌گیری از تابع ( ) imagefilledpolygon می‌توان یک چند ضلعی رنگ آمیزی شده را بر روی صفحه ترسیم نمود.

۱۰- به کمک تابع ( ) imageTTFtext می‌توان متنی را بر روی یک تصویر پویا درج کرد. جهت استفاده از متون به‌همراه تصاویر پویا بهره‌گیری از امکانات کتابخانه کدباز FreeType ضروری است.

فعالیتها

۱- برنامه‌ای جهت نمایش یک progress bar بر روی صفحه مرورگر ایجاد نمایید. نمونه چنین چیزی را احتمالاً هنگام نصب برنامه‌های تحت ویندوز و یا هنگام بارگذاری یک انیمیشن flash بر روی صفحه مرورگر دیده‌اید. نوار progress bar روند نصب یا بارگذاری را به اطلاع کاربر می‌رساند.

۲- برنامه‌ای بنویسید که عنوان یک تصویر برروی صفحه مرورگر اینترنت را بر مبنای اینکه از طریق یک فرم یا یک دنباله پرس و جو مورد درخواست قرارگرفته است مشخص نماید. کاربر باید بتواند اندازه تصویر، رنگ پس‌زمینه و رنگ مورد استفاده در خود زمینه و همچنین وجود و اندازه سایه فرضی زمینه را مشخص نماید.

ساعت پانزدهم

بهره‌گیری از تاریخ و زمان

تاریخ و ساعت آن‌چنان در زندگی روزمره ما وارد شده‌است که بدون هیچ تفکری همه مردم در تمام نقاط دنیا از آن بهره می‌گیرند. بااین وجود ابهامات موجود در تقویمهای مورد استفاده ما مانع از آن‌است که بتوانیم آنها را در برنامه‌های کامپیوتری مورد بهره‌برداری قرار دهیم. خوشبختانه زبان برنامه‌نویسی PHP ابزارها و امکانات بسیار کارآمدی دررابطه با محاسبات تاریخ، پیش روی برنامه‌نویسان قرار داده که این امر استفاده از عنصر زمان را در برنامه‌های ایجاد شده با این زبان بسیار سهل و ساده می‌کند.

در درس این ساعت موارد زیر را در این رابطه مورد بررسی قرار می‌دهیم:

- **نحوه دستیابی به تاریخ و ساعت جاری**
- **نحوه دستیابی به اطلاعاتی درمورد یک تاریخ مشخص**
- **نحوه قالب‌بندی اطلاعات تاریخ**
- **نحوه اعتبارسنجی یک تاریخ**
- **نحوه تنظیم تاریخ**
- **نحوه ایجاد یک تقویم ساده با استفاده از یک برنامه کامپیوتری**
- **نحوه ایجاد کتابخانه‌ای از کلاسها جهت بهره‌گیری از تاریخ در فرمهای HTML**

در ادامه درس به بررسی هریک از موارد فوق می‌پردازیم.

دستیابی به زمان (شامل تاریخ و ساعت) از طریق تابع ( ) time

تابع ( ) time در زبان برنامه‌نویسی PHP کلیه اطلاعاتی را که درمورد تاریخ و ساعت جاری به آنها نیاز دارید، در اختیارتان قرار می‌دهد. این تابع جهت اجرای عملیاتش به هیچ آرگومانی نیاز ندارد اما به‌عنوان نتیجه این عملیات یک عدد صحیح را به برنامه فراخواننده باز می‌گرداند. این عدد صحیح به‌سبب بزرگی نامأنوس به‌نظر می‌رسد اما در عین حال مفهوم آن برای PHP بسیار مفید و قابل استفاده است. به نمونه زیر که کاربردی از این تابع را نشان می‌دهد، توجه کنید:

```
print time ( ) ;
// sample output : 1127732399
```

عددصحیح بازگشتی از تابع ( ) time نماینده تعداد ثانیه‌های سپری شده از نیمه شب اولین روز ژانویه سال ۱۹۷۰ میلادی است (مرجع این ساعت استاندارد GMT است). نیمه شب فوق در میان برنامه‌نویسان و جامعه کامپیوتری به " UNIX epoch " معروف است. معمولاً چنین متداول است که به تعداد ثانیه‌های سپری شده از آن لحظه تا لحظه جاری timestamp یا برچسب زمان گفته می‌شود. در زبان PHP امکانات بسیار قابل توجهی برای تبدیل یک برچسب زمان به فرم قابل درک توسط انسان پیش‌بینی شده است.

با وجود این، شاید چنین پرسشی در این مقطع ذهن را اندکی آزار دهد که آیا برچسب زمان یک روش منفی بوده و عکس گفته بالا صحیح است. جالب است بدانید که تنها از این عدد صحیح ساده می‌توان اطلاعات بسیار زیاد و با ارزشی را استخراج نمود. نکته جالب‌تر این است که انجام عملیات ریاضی و محاسباتی با استفاده از برچسب زمان بیش از آنچه تصورش را می‌کنید، ساده و سرراست است.

سیستم ثبت زمانی را تصور کنید که طراح آن روزها و ماهها و سالها را به‌عنوان مقادیری مجزا از یکدیگر ثبت و نگهداری می‌کند. حال وضعیتی را تصور کنید که لازم باشد تا در برنامه‌ای که از این سیستم جهت ثبت زمان استفاده می‌کند، یک روز به تاریخ مشخصی اضافه شود. درصورتی که تاریخ فعلی برای مثال روز ۳۱ دسامبر سال ۱۹۹۹ میلادی باشد با افزودن تنها یک روز به تاریخ فوق برنامه مجبور خواهد بود تا علاوه بر تنظیم روز جدید با عدد یک عنصر ماه را به عدد یک (به نشانه ماه ژانویه) و عنصر سال را نیز به عدد ۲۰۰۰ تغییر دهد. این در حالی‌است که با بهره‌گیری از طرز نمایش برچسب زمان تنها کافی است که عددی معادل با تعداد ثانیه‌های موجود در یک روز (که برابر با ۳۶۰۰ × ۲۴ است) را به آن اضافه کنیم. پس از آن با استفاده از امکانات پیش‌بینی شده در PHP می‌توانید حاصل عددی به دست آمده را به فرم قابل خواندن و کاربر پسند تبدیل نمایید.

تبدیل برچسب زمان با استفاده از تابع ( ) getdate

اکنون با در اختیارداشتن یک برچسب زمان می‌توانید به استفاده از آن بپردازید. یکی از عملیاتی که هر کاربری در نگاه اول مایل به انجام آن است، تبدیل این برچسب به‌صورتی است که قابل خواندن باشد (نه برای PHP بلکه برای انسان). تابع ( ) getdate در زبان PHP یک برچسب زمان را به‌عنوان آرگومان ورودی دریافت کرده و یک آرایه انجمنی شامل اطلاعات دقیق درمورد زمان مربوطه را به برنامه فراخواننده باز می‌گرداند. درصورتی که هیچ آرگومانی را هنگام فراخوانی این تابع به‌عنوان ورودی مورد استفاده قرار ندهید، تابع مذکور از برچسب زمان جاری که حاصل فراخوانی تابع ( ) time است به‌عنوان آرگومان استفاده خواهد کرد (به‌عبارت دیگر، درصورت عدم استفاده از آرگومان، تابع ( ) getdate جهت دستیابی به برچسب زمان جاری اقدام به فراخوانی تابع ( ) time خواهد کرد). جدول ۱- ۱۵ جزئیات مربوط به عناصر آرایه بازگشتی توسط تابع ( ) getdate را مورد بررسی قرار داده است.

جدول ۱- ۱۵ عناصر بازگشتی ازتابع ( ) getdate

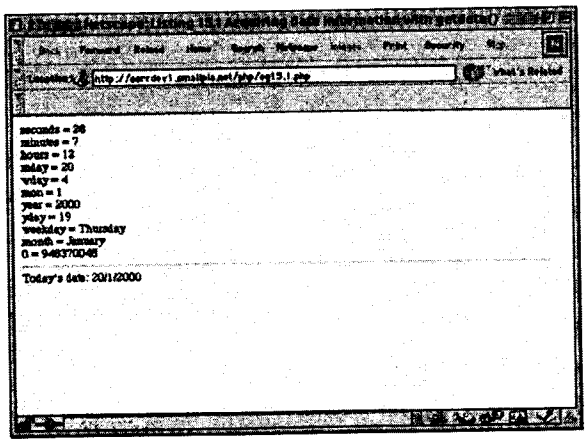
نام کلید دستیابی	توصیف	مثال
seconds	ثانیه‌های سپری شده (از صفر تا ۵۹)	28
minutes	دقیقه‌های سپری شده (از صفر تا ۵۹)	7
hours	ساعات سپری شده (از صفر تا ۲۳)	12
mday	روزی از ماه (از یک تا ۳۱)	20
Wday	روزی از هفته (از صفر تا ۶)	4
mon	ماهی از سال (از یک تا ۱۲)	1
Year	سال (چهار رقم)	2000
Yday	روزی از سال (از صفر تا ۳۶۵)	19
Weekday	نام روزی از روزهای هفته	Thursday
month	نام ماهی از سال	January
0	برچسب زمان	948370048

برنامه موجود در لیست ۱- ۱۵، تابع ( ) getdate را در خط ۷ جهت استخراج اطلاعات زمانی از برچسب زمان مورد استفاده را به‌عنوان آرگومان مورد فراخوانی قرار می‌دهد. نمایش عناصر تشکیل دهنده زمان با بهره‌گیری از یک ساختار تکرار foreach و فراخوانی تابع print صورت می‌پذیرد.

```
1: <html>
2: <head>
3: <title>Listing 15.1 Acquiring date information with getdate()</title>
4: </head>
5: <body>
6: <?php
7: $date_array = getdate(); // no argument passed so today's date will be used
8: foreach ( $date_array as $key => $val ) {
9:     print "$key = $val<br>";
10: }
11: ?>
12: <hr>
13: <?
14: print "Today's date:
➥".$date_array['mday']."/".$date_array['mon']."/".$date_array['year']."<p>";
15: ?>
16: </body>
17: </html>
```

لیست ۱- ۱۵ بهره‌گیری از تابع ( ) getdate جهت دستیابی به اطلاعات زمانی مورد نیاز

خروجی حاصل از اجرای این برنامه در شکل ۱- ۱۵ قابل مشاهده و بررسی است. همان‌گونه که مشاهده می‌کنید، تابع ( ) getdate اطلاعات زمانی را با توجه به موقعیت جغرافیایی کامپیوتر میزبان باز می‌گرداند.



شکل ۱- ۱۵ نتیجه استفاده از تابع ( ) getdate

تبدیل یک برچسب زمان با بهره‌گیری از تابع ( ) date

علاوه بر تابع ( ) getdate از تابع ( ) date نیز می‌توان جهت تبدیل برچسب زمان استفاده کرد.

با این حال معمولاً هنگامی از تابع ( ) getdate استفاده می‌شود که دستیابی به اجزای مختلف زمان مدنظر باشد درصورتی که قصد ما از تبدیل برچسب زمان صرفاً نمایش دنباله کاراکتری متناظر با آن در یک قالب خوانا باشد از تابع ( ) date استفاده می‌کنیم. این تابع یک برچسب زمان را به‌عنوان آرگومان ورودی پذیرفته و دنباله کاراکتری متناظر با آن که در قالبی کاربر پسند درآمده را به‌عنوان خروجی برنامه فراخواننده باز می‌گرداند. با استفاده از یک دنباله کاراکتری که به عنوان اولین آرگومان تابع ( )date مورد بهره‌برداری قرار می‌گیرد، برنامه نویس می‌تواند قالب خروجی (چگونگی نمایش زمان) را به‌طور دقیق و کامل کنترل نماید. جدول ۲- ۱۵ لیستی از کدهای قابل استفاده جهت ایجاد دنباله کاراکتری قالب‌بندی را نشان می‌دهد. هرنوع داده دیگری را که غیر از این کدها در دنباله کاراکتری فوق مورد استفاده قرار دهید، تابع ( ) date عیناً در خروجی نمایش خواهد داد.

جدول ۲- ۱۵ کدهای قابل استفاده در دنباله کاراکتری قالب‌بندی از تابع ( ) date

کد قابل استفاده	توصیف	مثال
a	am یا pm با حروف کوچک انگلیسی	pm
A	AM یا PM با حروف بزرگ انگلیسی	PM
d	روزی از ماه (عددی که در موارد کوچک‌تر از ۱۰ پیش از آن از رقم صفر استفاده می‌شود)	08
D	روزی از هفته (شامل تنها سه حرف اول)	Thu
F	نام ماهی از سال	January
h	ساعت (عددی از صفر تا ۱۱ که درصورت تک رقمی بودن پیش از آن از رقم صفر استفاده می‌شود)	02
H	ساعت (عددی از صفر تا ۲۳ که درصورت تک رقمی بودن پیش از آن از رقم صفر استفاده می‌شود)	17
g	ساعت (عددی از صفر تا ۱۱ که درصورت تک رقمی بودن از رقم پیش از آن استفاده نمی‌شود)	7
G	ساعت (عددی از صفر تا ۲۳ که درصورت تک رقمی بودن از رقم صفر پیش از آن استفاده نمی‌شود)	8
i	دقیقه	47
j	روزی از ماه (درصورت تک رقمی بودن از رقم صفر پیش از آن استفاده نمی‌شود)	20

l	نام روزی از ایام هفته	Thursday
L	سال کبیسه (عدد 1 به‌معنی اینکه سال کبیسه است و عدد صفر به‌معنی اینکه سال کبیسه نیست)	1
m	ماهی از سال (عددی از یک تا ۱۲ که درصورت تک رقمی بودن پیش از آن از رقم صفر استفاده می‌شود)	01
M	ماهی از سال (تنها سه حرف اول نام ماه)	Jan
n	ماهی از سال (عددی از یک تا ۱۲ که درصورت تک رقمی بودن از رقم صفر پیش از آن استفاده نمی‌شود)	1
s	ثانیه‌ای از ساعت	24
r	زمان کامل مطابق با استاندارد RFC 822 (جهت اطلاع بیشتر مراجعه شود به سایت زیر: (http: // www. Faqs. Org / rfcs / rfc822. html	Web, 26 Sep 2001 15:15:14 t 0100
U	برچسب زمان	948372444
y	سال (عددی دو رقمی)	00
Y	سال (عددی چهاررقمی)	2000
z	روزی از سال (عددی از صفر تا ۳۶۵)	19
Z	تفاوت زمانی از مرجع GMT برحسب ثانیه	0

دنباله جدول ۲- ۱۵

با استفاده از دو دنباله کاراکتری قالب‌بندی در خطوط ۷ و ۱۱ کاربرد این کدها را در نحوه قالب‌بندی برچسب زمان نشان می‌دهد.

```
1: <html>
2: <head>
3: <title>Listing 15.2 Formatting a date with date()</title>
4: </head>
5: <body>
6: <?php
7: print date("m/d/y G.i:s<br>", time());
8: // 09/26/01 15.46:30
9: print "<br>";
10: print "Today is ";
11: print date("j of F Y, \a\\t g.i a", time());
12: // Today is 26 of September 2001, at 3.46 pm
13: ?>
14: </body>
15: </html>
```

لیست ۲- ۱۵ قالب‌بندی زمان با استفاده از تابع ( ) date و بهره‌گیری از کدهای جدول ۲- ۱۵

همان‌گونه که در این برنامه مشاهده می‌کنید، تابع ( ) date یک بار در خط ۷ جهت نمایش

زمان به فرم کوتاه و باردیگر در خط ۱۱ برای نمایش زمان در فرم کامل فراخوانی شده است.

هرچند که درک دنباله کاراکتری قالب‌بندی در نگاه اول اندکی مشکل می‌نماید اما در حقیقت استفاده از آن بسیار ساده است. ممکن است بخواهید در خروجی علاوه بر زمان موردنظرتان یک دنباله کاراکتری را نیز قبل یا بعد از آن نمایش دهید. همان‌گونه که پیشتر نیز اشاره شد، این‌کار با درج متن موردنظرتان در درون دنباله کاراکتری قالب‌بندی (آرگومان اول تابع ( ) date) میسر است. درمواردی که متن مورد نظرتان شامل کاراکترهای قالب‌بندی جدول ۲- ۱۵ باشد، با استفاده از یک علامت \ پیش از آنها می‌توانید ترتیبی دهید که آن کاراکتر توسط PHP به‌عنوان کاراکتر قالب‌بندی فرض نشده و بنابراین عیناً در خروجی چاپ شود. درصورتی که متن موردنظرتان شامل کاراکترهایی باشد که جهت کنترل خروجی از آنها استفاده می‌شود، با استفاده از دو علامت \ متوالی می‌توانید تاثیر آنها را در متن حفظ نمایید. بدین ترتیب جهت استفاده از کاراکتر n در متن موردنظرتان کافی است تا از \\ n استفاده کنید؛ چرا که \ n خود کاراکتر کنترلی بوده و موجب درج یک خط جدید در متن خواهد شد. تابع date ( ) اطلاعاتی را باز می‌گرداند که به موقعیت جغرافیایی شما نسبت به مبدا سنجش زمان (GMT) بستگی خواهد داشت. اگر بخواهید زمان را در قالب GMT به برنامه فراخواننده باز گردانید، به‌جای استفاده از این تابع می‌توانید از تابع ( ) gmdate بهره بگیرید. شرایط استفاده از این تابع دقیقاً مشابه شرایط استفاده از تابع ( ) date است.

ایجاد برچسب زمان با بهره‌گیری از تابع ( ) mktime

تا بدین جا توانستیم اطلاعات مربوط به زمان فعلی را در قالب برچسب زمان مورد دستیابی قرار داده و آنها را در قالب خوانا نمایش دهیم. اما هیچ روشی را تا به حال برای ایجاد یک برچسب زمان دلخواه عنوان نکردیم. با استفاده از تابع ( ) mktime، می‌توانیم برچسب زمان موردنظرمان را ایجاد کرده و با بهره‌گیری از تابع ( ) date یا ( ) gmdate آن‌را به قالبی کاربر پسند و خوانا تبدیل نماییم. تابع ( ) mktime به‌عنوان ورودی تعداد شش آرگومان از نوع عدد صحیح دریافت می‌کند که ترتیب آنها چنین است:

hour	(ساعت)
minute	(دقیقه)
second	(ثانیه)
month	(ماه)
day	(روزی از سال)
year	(سال)

برنامه موجود در لیست ۳- ۱۵، به سادگی تابع ( ) mktime را جهت ایجاد یک برچسب زمان فراخوانی کرده و حاصل را با بهره‌گرفتن از تابع ( ) date به قالبی خوانا تبدیل نموده است.

```
 1: <html>
 2: <head>
 3: <title>Listing 15.3 Creating a timestamp with mktime()</title>
 4: </head>
 5: <body>
 6: <?php
 7: // make a timestamp for 1/5/99 at 2.30 am
 8: $ts = mktime( 2, 30, 0, 5, 1, 1999 );
 9: print date("m/d/y G.i:s<br>", $ts);
10: // 05/01/99 2.30:00
11: print "<br>";
12: print "The date is ";
13: print date("j of F Y, \a\\t g.i a", $ts );
14: // The date is 1 of May 1999, at 2.30 am
15: ?>
16: </body>
17: </html>
```

لیست ۳- ۱۵ ایجاد یک برچسب زمان با بهره گیری از تابع ( ) mktime

همان‌گونه که در کد این برنامه مشاهده می‌کنید تابع ( ) mktime در خط ۸ به منظور ایجاد یک برچسب زمان فراخوانی شده و حاصل این فراخوانی به متغیری با نام $ts نسبت داده شده است. در خطوط ۹ و ۱۳ با فراخوانی تابع ( ) date و ارسال برچسب زمان ذخیره شده در $ts به‌عنوان آرگومان دوگونه خروجی کاربر پسند از زمان ایجاد شده است. درصورتی که هنگام فراخوانی تابع ( ) mktime از هیچ آرگومانی استفاده نشود، تابع مذکور از اجزای زمان فعلی جهت ایجاد برچسب زمان استفاده خواهد کرد. تابع ( ) mktime علاوه بر این قادر است تا به‌طور هوشمندانه‌ای خطای ناشی از آرگومانهای ورودی را تصحیح نماید. به عنوان مثال، درصورتی که از عدد صحیح 25 به‌عنوان اولین آرگومان این تابع که بیانگر ساعت است استفاده شود، تابع ( ) mktime به‌طور خودکار از عدد 1 (یا دقیق‌تر بگوییم 1.00am) استفاده خواهد کرد و همین روند درمورد سایر آرگومانها یعنی دقیقه، ثانیه، ماه روز و سال توسعه پیدا می‌کند، بدین معنی که درصورت نیاز مقادیر آنها نیز به‌گونه‌ای متناسب تغییر خواهند کرد.

بررسی یک تابع مشخص با استفاده از تابع ( ) checkdate

ممکن است گاهی لازم باشد تا اطلاعات مربوط به تاریخ را از طریق یک فرم ورودی از کاربر دریافت کنیم. در چنین مواردی پیش از ذخیره تاریخ دریافتی در یک بانک اطلاعاتی یا انجام هرگونه عملیاتی بر روی آن لازم است تا از صحت و اعتبار اجزای تشکیل دهنده آن اطمینان حاصل نماییم. تابع ( ) checkdate در زبان PHP به همین منظور پیش‌بینی شده است. این تابع سه عدد صحیح را به‌عنوان آرگومان ورودی از برنامه فراخواننده دریافت می‌کند. این آرگومانها به ترتیب نماینده اجزای ماه، روز و سال تاریخ مورد بررسی می‌باشند. تابع ( ) checkdate درصورت اعتبار داشتن آرگومانهای ورودی، یعنی

درصورتی که عدد ماه بین یک تا ۱۲ بوده و عدد روز در آن ماه مشخص شده وجود داشته باشد (با درنظر گرفتن موارد استثنا برای سالهای کبیسه) و همچنین عدد سال مابین صفر تا ۳۲۷۶۷ باشد، مقدار true را به برنامه فراخواننده باز می‌گرداند. با وجود این دقت کنید که علیرغم معتبر بودن تاریخ مورد بررسی توسط این تابع و بازگشت مقدار true از آن، ممکن است توسط سایر توابع تاریخ به‌عنوان یک تاریخ معتبر مورد قبول واقع نشود. برای نمونه، فراخوانی زیر را که مربوط به اعتبارسنجی تاریخ 04 / 04 / 1066 است، درنظر بگیرید:

```
Checkdate (4, 4, 1066) ;
```

با وجودی که فراخوانی فوق حاصل true را در پی دارد، استفاده از اجزای تاریخ فوق جهت ایجاد یک برچسب زمان با بهره‌گیری از تابع ( ) mktime نتیجه‌ای جز عدد 1 - را در پی نخواهد داشت. به‌عنوان یک قاعده عمومی هرگز از تابع ( ) mktime جهت ایجاد برچسبهای زمانی مربوط به سالهای قبل از ۱۹۰۲ میلادی استفاده نکنید. همچنین هنگام به‌کارگیری هرگونه تاریخی که قبل از سال ۱۹۷۰ میلادی باشد، دقت مضاعف به خرج دهید.

بررسی یک مثال

اجازه دهید تا در این قسمت مثالی را ارائه دهیم که در آن از بیشتر توابع مورد بررسی در این درس استفاده کرده‌ایم. قصد ما از ارائه این مثال این است که تقویمی بسازیم که قابلیت نمایش هرگونه تاریخی را برای ماههای مربوط به سال‌های ۱۹۸۰ تا ۲۰۱۰ میلادی را داشته باشد. در این مثال کاربر قادر خواهد بود تا ماه و سال مورد نظر خود را از دو لیست مجزا که به همین منظور پیش‌بینی شده‌اند، انتخاب نماید. با این اقدام او، برنامه در پاسخ، تاریخهای موجود در آن ماه را که بر مبنای روزهای هفته سازماندهی شده‌اند در خروجی نمایش می‌دهد. در این برنامه از دو متغیر با اسامی $month و $year جهت نگهداری مقادیر انتخاب شده توسط کاربر استفاده شده است. برنامه از مقادیر این متغیرها جهت ایجاد یک برچسب زمانی بر مبنای اولین روز از ماه استفاده خواهد کرد. اگر کاربر از انتخاب ماه سرباز زند یا مقدار ورودی معتبر نباشد، برنامه به‌طور پیش‌فرض از اولین روز ماه جاری جهت انجام محاسبات مورد نیازش استفاده خواهد کرد.

بررسی مقادیر وارد شده در برنامه

هنگامی که کاربر برای اولین مرتبه از این برنامه استفاده می‌کند، قادر به ارسال هیچ‌گونه اطلاعاتی به سرور نخواهد بود. بنابراین لازم است تا برنامه به نوعی این وضعیت را که طی آن متغیرهای $month و $year فاقد مقدار هستند، کنترل نماید. برای انجام چنین کاری می‌توانیم از تابع ( ) isset که در درسهای گذشته با آن آشنا شدید، استفاده کنیم. این تابع جهت بررسی این مطلب که آیا متغیری

دارای مقدار است یا خیر، ایده‌آل است. درصورتی که متغیر ارسالی به این تابع به عنوان آرگومان فاقد مقدار باشد یا اصلاً تعریف نشده باشد، تابع مذکور به‌عنوان نتیجه بررسی مقدار false را به برنامه فراخواننده باز می‌گرداند. با این وجود ترجیح می‌دهیم تا برای انجام این بررسی از تابع ( ) checkdate که در قسمت قبل آن‌را مورد بحث قرار دادیم، استفاده نماییم. برنامه موجود در لیست ۴- ۱۵ بخشی از برنامه را که مسئول بررسی مقادیر متغیرهای $month و $year است، نشان می‌دهد. این بخش از برنامه همچنین بر مبنای مقادیر این متغیرها اقدام به ایجاد یک برچسب زمان می‌کند.

```
1: <?php
2: if ( ! checkdate( $month, 1, $year ) ) {
3:     $nowArray = getdate();
4:     $month = $nowArray['mon'];
5:     $year = $nowArray['year'];
6: }
7: $start = mktime ( 12, 0, 0, $month,     1, $year );
8: $firstDayArray = getdate($start);
9: ?>
```

لیست ۴- ۱۵ بررسی مقادیر ورودی برنامه

از آنجا که کد موجود در این لیست بخشی از برنامه اصلی است، انتظار نداریم که خروجی را بر روی صفحه مرورگر نمایش دهد. در عبارت تصمیم‌گیری if در خط ۲ از برنامه همان‌گونه که مشاهده می‌کنید، از تابع ( ) checkdate جهت ارزیابی مقادیر متغیرهای $month و $year استفاده شده است. درصورتی که این متغیرها تعریف نشده باشند تابع مذکور مقدار false را به برنامه فراخواننده باز می‌گرداند. این بدان دلیل است که ایجاد یک تاریخ معتبر با بهره‌گیری از متغیرهایی که هنوز تعریف نشده‌اند امکان‌پذیر نمی‌باشد. این رویکرد یک فایده جانبی را نیز به‌همراه دارد و آن اینکه می‌توان از معتبر بودن داده‌های ارسالی توسط کاربر اطمینان حاصل کرد.

در صورتی که داده‌های ارسالی توسط کاربر معتبر نباشند، آن‌گاه فراخوانی تابع ( ) getdate در خط ۳ از این لیست موجب ایجاد یک آرایه انجمنی بر مبنای تاریخ فعلی خواهد شد. با در دست داشتن این آرایه می‌توان مقادیر متغیرهای $month و $year را با بهره‌گیری از مقادیر عناصر mon و year از آن تعیین نمود. همان‌گونه که مشاهده می‌کنید چنین فرآیندی در خطوط ۴ و ۵ برنامه ایجاد شده است.

پس از اطمینان از این مطلب که متغیرهای $month و $year شامل مقادیر معتبر هستند با بهره‌گیری از تابع ( ) mktime در خط ۷ از برنامه می‌توانیم برچسب زمان موردنظرمان را برای اولین روز ماه ایجاد نماییم. از آنجا که به اطلاعات مربوط به این برچسب زمان نیاز پیدا خواهیم کرد، در خط ۸ متغیری با نام ( ) firstdayarray ایجاد کرده و آرایه انجمنی بازگشتی از تابع ( ) getdate را که بر مبنای این برچسب زمان ایجاد می‌شود، در آن متغیر ذخیره می‌کنیم.

ایجاد فرم HTML

در این قسمت قصد داریم تا با ایجاد یک فرم HTML تسهیلاتی را جهت تعیین مقادیر ماه و سال (مقادیر متغیرهای $month و $year) و مشاهده نتایج در اختیار کاربر قرار دهیم. برای این منظور ما از المان SELECT استفاده خواهیم کرد. هرچند که می‌توانیم این فرآیند را به روش دیگری با استفاده از کد HTML نیز انجام دهیم. اما باید به‌گونه‌ای اطمینان حاصل کنیم که مقادیر پیش‌فرض لیستهای انتخاب منطبق بر ماهی است که توسط کاربر انتخاب شده است. از این‌رو لیستهای مورد نظر را به صورت پویا (با برنامه‌نویسی) ایجاد کرده و در صورت نیاز خصلت SELECT را به المان OPTION اضافه می‌کنیم. برنامه لیست ۵- ۱۵ کد این برنامه را نشان می‌دهد.

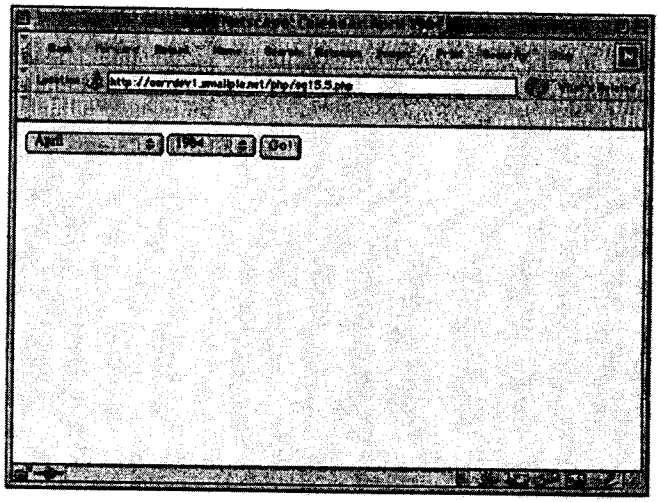
```
 1: <?php
 2: if ( ! checkdate( $month, 1, $year ) ) {
 3:     $nowArray = getdate();
 4:     $month = $nowArray['mon'];
 5:     $year = $nowArray['year'];
 6: }
 7: $start = mktime ( 12, 0, 0, $month, 1, $year );
 8: $firstDayArray = getdate($start);
 9: ?>
10: <html>
11: <head>
12: <title><?php print "Calendar: ".$firstDayArray['month']
13:          ." ".$firstDayArray['year'] ?></title>
14: <head>
15: <body>
16: <form method="post">
17: <select name="month">
18: <?php
19: $months = Array("January", "February", "March", "April",
20:                  "May", "June", "July", "August", "September",
21:                  "October", "November", "December");
22: for ( $x=1; $x <= count( $months ); $x++ ) {
23:     print "\t<option value=\"$x\"";
24:     print ($x == $month)?" SELECTED":"";
25:     print ">".$months[$x-1]."\n";
26: }
27: ?>
28: </select>
29: <select name="year">
30: <?php
31: for ( $x=1980; $x<2010; $x++ ) {
32:     print "\t<option";
33:     print ($x == $year)?" SELECTED":"";
34:     print ">$x\n";
35: }
36: ?>
37: </select>
38: <input type="submit" value="Go!">
39: </form>
40: </body>
41: </html>
```

لیست ۵- ۱۵ ایجاد یک فرم HTML جهت برنامه تقویم

همان‌گونه که در این لیست مشاهده می‌کنید پس از ایجاد برچسب زمان نظر و ذخیره آن در متغیر $start و ایجاد آرایه $firstdayarray در خط ۸ ، می‌توانیم طراحی فرم HTML را آغاز کنیم. توجه کنید که در خطوط ۱۲ و ۱۳ از این برنامه با استفاده از آرایه $firstDayArray مقادیر ماه و سال را به المان TITLE اضافه کرده‌ایم (این مقادیر در نوار عنوان پنجره مرورگر اینترنت کاربر خواهند شد). در خط ۱۶ روش ارسال اطلاعات فرم به وب سرور (یا به‌طور دقیق‌تر به برنامه PHP) تعیین شده است اما چنانچه می‌بینید نام هیچ برنامه‌ای جهت پردازش اطلاعات مذکور تعیین نشده است. به‌خاطر بیاورید که اگر از آرگومان ACTION در المان FORM یک سند استفاده نشود، فرض بر این است که اطلاعات آن سند توسط برنامه موجود در همان سند مورد بررسی قرار می‌گیرد. جهت ایجاد المان SELECT یا همان لیست، شامل ماههای سال در خط ۱۸ قطعه کد جدیدی به زبان PHP ایجاد شده است که وظیفه آن درج المانهای مختلف OPTION در درون المان SELECT است. برای انجام این کار ابتدا در خط ۱۹ آرایه‌ای با نام $months ایجاد می‌شود. این آرایه شامل اسامی دوازده ماه سال می‌باشد.

پس از آن در هربار گذر از حلقه تکرار به ازای هریک از این ماهها در خط ۲۳ از برنامه یک المان OPTION ایجاد می‌شود. این روش به احتمال قوی یکی از پیچیده‌ترین روشهای موجود برای ایجاد یک المان ساده SELECT است. چنانکه مشاهده می‌کنید، مقدار متغیر $X (که درواقع شمارنده یا کنترل‌کننده حلقه for است) درهر بار گذر از حلقه با مقدار ذخیره شده در متغیر $month در خط ۲۴ مقایسه می‌شود. درصورتی که مقدار این دو متغیر با یکدیگر برابر باشد، دنباله کاراکتری 'SELECTED به ساختار المان OPTION اضافه خواهد شد. این فرآیند اطمینان می‌دهد که هنگام بارگذاری سند وب، ماه صحیحی از سال به‌طور خودکار انتخاب می‌شود. در خطوط ۲۹ تا ۳۷ برنامه از روش مشابهی برای ایجاد لیست شامل سالهای موردنظر استفاده شده است. در انتهای برنامه با اتمام کد PHP در خط ۳۶ کد HTML باقیمانده دکمه‌ای را جهت ارسال اطلاعات در خط ۳۸ برنامه ایجاد می‌کند.

اکنون آنچه که در اختیار داریم، یک فرم HTML است که اطلاعات مربوط به ماه و سال را به‌عنوان ورودی به برنامه‌ای ارسال می‌کند که در همان فرم واقع است. مقادیر پیش‌فرض ماه و سال در این فرم، درصورتی که کاربر برای اولین مرتبه با این فرم کار می‌کند، ماه و سال جاری بوده و در غیر این‌صورت ماه و سالی است که کاربر در بازدید قبلی فرم آنها را انتخاب کرده است. خروجی این برنامه در شکل ۲- ۱۵ قابل مشاهده است.



شکل ۲- ۱۵ فرم ورودی تقویم

ایجاد جدولی برای نمایش تقویم

پس از ایجاد فرم ورودی اکنون نیازمند فرمی هستیم که کلیه روزهای موجود در ماه و سال انتخاب شده توسط کاربر را در قالب یک جدول نمایش دهد. لیست ۶- ۱۵، شامل برنامه کامل مورد نیاز جهت پردازش اطلاعات فرم ورودی و نمایش اطلاعات خروجی در قالب خواسته شده است.

```
<?php
define("ADAY", (60*60*24) );
if ( ! checkdate( $month, 1, $year ) ) {
    $nowArray = getdate();
    $month = $nowArray['mon'];
    $year = $nowArray['year'];
}
$start = mktime ( 12, 0, 0, $month, 1, $year );
$firstDayArray = getdate($start);
?>
<html>
<head>
<title><?php print "Calendar: ".$firstDayArray['month']
        ." ".$firstDayArray['year'] ?></title>
</head>
<body>
<form method="post">
<select name="month">
<?php
$months = Array("January", "February", "March", "April",
                "May", "June", "July", "August", "September",
                "October", "November", "December");
for ( $x=1; $x <= count( $months ); $x++ ) {
    print "\t<option value=\"$x\"";
```

لیست ۶- ۱۵ برنامه کامل تقویم ماهی از سال

```
25:     print ($x == $month)?" SELECTED":"";
26:     print ">".$months[$x-1]."\n";
27: }
28: ?>
29: </select>
30: <select name="year">
31: <?php
32: for ( $x=1980; $x<2010; $x++ ) {
33:     print "\t<option";
34:     print ($x == $year)?" SELECTED":"";
35:     print ">$x\n";
36: }
37: ?>
38: </select>
39: <input type="submit" value="Go!">
40: </form>
41: <p>
42: <?php
43: $days = Array("Sunday", "Monday", "Tuesday", "Wednesday",
44:               "Thursday", "Friday", "Saturday");
45: print "<TABLE BORDER = 1 CELLPADDING=5>\n";
46: foreach ( $days as $day )
47:     print "\t<td><b>$day</b></td>\n";
48: for ( $count=0; $count < (6*7); $count++ ) {
49:     $dayArray = getdate( $start );
50:     if ( (($count) % 7) == 0 ) {
51:         if ( $dayArray['mon'] != $month )
52:             break;
53:         print "</tr><tr>\n";
54:      }
55:     if ( $count < $firstDayArray['wday'] || $dayArray['mon'] != $month )
56:         print "\t<td><br></td>\n";
57:     else {
58:         print "\t<td>".$dayArray['mday']." ". $dayArray['month']."</td>\n";
59:         $start += ADAY;
60:     }
61: }
62: print "</tr></table>";
63: ?>
64: </body>
65: </html>
```

دنباله لیست ۶-۱۵

همان‌گونه که مشاهده می‌کنید به‌دلیل آنکه شاخصهای جدول حاصل روزهای هفته هستند، می‌توانیم یک ساختار تکرار ایجاد کنیم و در هربار گذر از حلقه به ازای عناصر آرایه‌ای از اسامی روزها که پیشتر در خط ۴۳ آن‌را ایجاد کرده‌ایم، با بهره‌گیری از تابع ( ) print در خط ۴۷ مقدار هریک از خانه‌های جدول را مشخص نماییم. تمام قدرت و قابلیت این برنامه در ساختار تکرار for موجود در خط ۴۸ نهفته است که در پاراگرافهای بعدی آن را بررسی می‌کنیم.

به‌منظور کنترل عملیات ساختار تکرار for در خط ۴۸ متغیری با نام $count را معرفی و مقداردهی می‌کنیم. این حلقه پس از تکرار ۴۲ مرتبه عملیاتی که پیش‌بینی شده است، متوقف خواهد

شد. بدین ترتیب می‌توانیم اطمینان حاصل کنیم که در برنامه به تعداد کافی از خانه‌های جدول را جهت درج اطلاعات تقویم ماه در اختیارمان قرار خواهد داد. در داخل حلقه با استفاده از تابع ( ) getdate اجزای تشکیل‌دهنده زمان از برچسب زمان ذخیره شده در متغیر $start در آرایه‌ای با نام $dayArray را ذخیره می‌کنیم. با وجودی که برچسب زمان مشخص شده توسط متغیر $start که در ابتدای هر حلقه مورد استفاده قرار می‌گیرد، همواره حاوی اولین روز از ماه است، در داخل حلقه با اضافه کردن ۱۴ ساعت (یا معادل عددی ۲۴ × ۶۰ × ۶۰) به برچسب زمان در خط ۵۲ به‌راحتی توانسته‌ایم روز بعدی ماه را مورد پردازش قرار دهیم.

در خط ۵۰ برنامه، با بهره‌گیری از عملگر محاسبه باقیمانده صحیح بررسی را در مورد اینکه آیا تعداد یک هفته از روزهای ماه مورد پردازش قرارگرفته است یا خیر انجام داده‌ایم. درصورتی که پاسخ مثبت باشد، یا به‌عبارت دیگر مقدار متغیر $count برابر با صفر یا مضربی از عدد ۷ باشد، عملیات درون این ساختار if به اجرا درخواهد آمد. این روشی است که ما به‌منظور آگاهی از این مطلب که آیا تولید عناصر سطری از جدول به اتمام رسیده و اکنون باید تولید عناصر سطر بعدی را از سر بگیریم، استفاده کرده‌ایم.

پس از تعیین اینکه پردازش مربوط به یک سطر از خانه‌های جدول به انتها رسیده یا خیر، می‌توانیم بررسی دیگری را در خط ۵۱ صورت دهیم. چنانچه مقدار مربوط به کلید دستیابی ' mon ' از آرایه $dayArray برابر با متغیر $month نباشد، لازم است تا به اجرای عملیات حلقه پایان دهیم. چنین عملی را توسط دستورالعمل break در خط ۵۲ انجام می‌دهیم. به‌خاطر داشته باشید که آرایه $dayArray شامل اطلاعات برچسب زمان $start یا به‌طور دقیق‌تر اطلاعات روزهای ماه انتخاب شده توسط کاربر است. هنگامی که به‌واسطه افزایش مقدار برچسب زمان درخط ۵۹ مقدار عنصر ماه از آنچه که کاربر انتخاب کرده بود، متجاوز شود عنصر ['month'] $dayArray از آرایه شامل مقداری متفاوت از مقدار انتخاب شده برای ماه که در متغیر $month ذخیره شده خواهد بود. به گفته دیگر اولین بررسی if در خط ۵۰ از اتمام عملیات پردازش روزهای ماه انتخاب شده توسط کاربر ما را مطلع خواهد کرد.

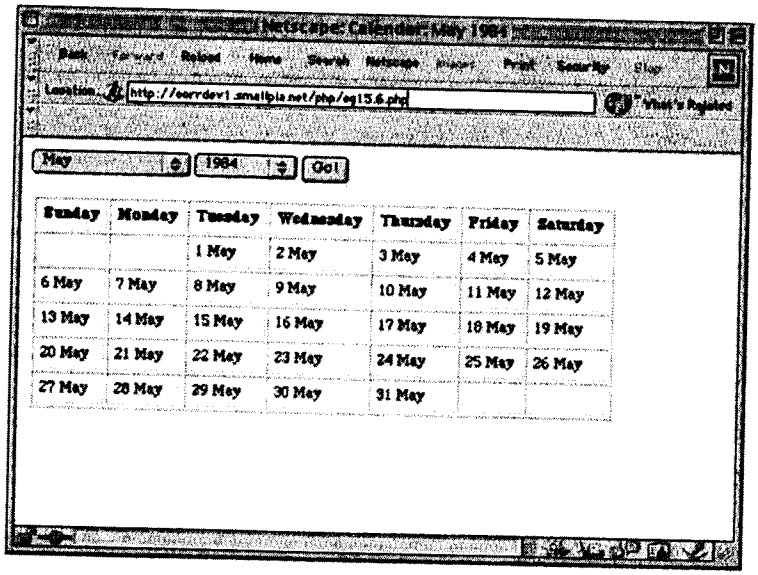
اما با فرض اینکه هنوز پردازش کلیه روزهای ماه به اتمام نرسیده است (در این صورت مقدار عنصر ['month'] $dayArray از آرایه با مقدار متغیر $month برابر خواهد بود)، برنامه به کار خود ادامه خواهد داد.

ساختار تصمیم‌گیری بعدی در خط ۵۵ که یک ساختار if / else می‌باشد، تعیین می‌کند که آیا باید اطلاعات تاریخ در خانه‌ای از سطر جدول درج شود یا خیر. دقت کنید که تمامی ماههای سال از روز یکشنبه آغاز نمی‌شوند. از این‌رو شاید در برخی موارد لازم باشد تا یکی دو خانه از اولین سطر هر ماه را خالی بگذاریم. همچنین دقت کنید که تمامی ماه‌های، سال در روز شنبه به پایان نمی‌رسند و به این علت، باردیگر شاید لازم باشد تا برخی از خانه‌های آخرین سطر هر ماه را خالی بگذاریم. همان‌گونه که

مشاهده می‌کنید، اطلاعات مربوط به اولین روز از هر ماه را در آرایه $firstDayArray ذخیره می‌کنیم. با این روش می‌توانیم با دستیابی به عنصر [' wday '] $First DayArray از این آرایه تعداد روزهای موجود در هفته‌ای از آن ماه را تشخیص دهیم. اگر چنانچه مقدار متغیر $count کوچک‌تر از مقدار این عنصر از آرایه باشد چنین نتیجه می‌گیریم که به خانه مورد نظر جهت درج اطلاعات نرسیده‌ایم. به همین صورت، درصورتی که مقدار متغیر $month برابر با مقدار عنصر ['mon'] $dayArray از آرایه نباشد نتیجه می‌گیریم که به انتهای پردازش روزهای آن ماه رسیده‌ایم و در عین حال در سطری از جدول هستیم که تمامی خانه‌های آن‌را پرنکرده‌ایم. به هرحال در هریک از دو صورت فوق عبارت موجود در خط ۵۶ یک خانه خالی از جدول را بر روی پنجره درج می‌کند یا به بیان بهتر، خانه بعدی جدول را خالی گذاشته و اطلاعاتی را در آن درج نمی‌کند.

بخش else از ساختار آخرین if در خط ۵۷ بخشی است که منتظر آن بودیم. اگر کنترل اجرای برنامه به این بخش رسیده باشد، بدان معنی است که برنامه مشغول پردازش اطلاعات مربوط به ماه انتخاب شده توسط کاربر بوده و ستون مربوط به روز جاری با شماره روزی که در عنصر [' wday '] $firstDayArray از آرایه ذخیره شده است، مطابقت دارد و این نیز خود بدان معناست که وقت آن است تا از آرایه انجمنی $dayArray که پیشتر در خط ۴۹ در درون ساختار تکرار for آن‌را ایجاد کردیم، جهت درج روز و ماه در درون خانه موردنظر از جدول تقویم ماه استفاده نماییم.

در نهایت، در خط ۵۹ از برنامه لازم است تا مقدار متغیر $start را که شامل برچسب زمان است به‌گونه‌ای مناسب افزایش دهیم. برای این‌کار به‌سادگی تعداد ثانیه‌های موجود در یک شبانه‌روز را که معادل با عدد ۲۴ × ۶۰ × ۶۰ است و ما پیشتر آن‌را محاسبه کرده و در ثابتی با نام ADAY ذخیره کرده‌ایم، به این برچسب زمان اضافه می‌کنیم. با این اقدام برنامه آماده است تا پردازش روز بعدی از ماه را در یک حلقه تکرار جدید و با ارزیابی مقدار جدیدی که در متغیر $start ذخیره شده است، آغاز نماید. خروجی حاصل از اجرای این برنامه را البته پس از انتخاب سال و ماه مورد نظر از فرم HTML ورودی، می‌توانید در شکلی مشابه شکل ۳- ۱۵ مشاهده نمایید.



شکل ۳- ۱۵ خروجی برنامه تقویم

کتابخانه‌ای برای محاسبات تقویم

به دلیل جایگاه متداول تاریخ و زمان در برنامه‌های کاربردی وب و اینکه بهره‌گیری از تاریخ معمولاً بخشی از برنامه‌های کاربردی وب به‌ویژه برنامه‌های تجاری و زمان‌بندی با اهداف متنوع است، چنین به‌نظر می‌رسد که وجود یک کتابخانه از کلاسهای مختلفی که بتواند برخی از عملیات مورد نیاز جهت نمایش و ارائه تاریخ را به‌طور خودکار انجام دهد، می‌تواند بسیار مفید و سودمند واقع شود. در این قسمت از درس قصد داریم چنین کتابخانه‌ای را ایجاد نماییم. در خلال این‌کار برخی از تکنیکهایی را که پیش از این بررسی کردیم نیز مورد توجه قرار خواهیم داد.

کتابخانه ساده مورد نظرمان که نام pulldown _ date را برای آن برگزیده‌ایم. به‌منظور بهره‌گیری در یک سایت که وظیفه آن اعطای مسولیتها از طریق پیمان به پیمان‌کاران مختلف است، ایجاد می‌شود. این وب سایت باید امکاناتی را جهت تعیین تاریخ و شروع و پایان پیمان و همچنین مقاطع زمانی را که پیمان‌کاران در دسترس کارفرمایان می‌باشند را در اختیار آنان قرار دهد. تعیین کلیه زمانهای فوق از طریق لیستهایی که به همین منظور در فرم HTML ورودی طراحی شده‌اند، صورت خواهد پذیرفت. تعیین یک زمان بخصوص مستلزم تعیین سه پارامتر زمانی مختلف روز و ماه و سال از طریق المانهای SELECT موجود در فرم HTML می‌باشد.

هنگامی که کاربر پس از انتخاب تاریخهای خواسته شده از جانب کارفرما فرم HTML خود را جهت پردازش ارسال می‌کند، یک برنامه PHP داده‌های ورودی را مورد بررسی وارزیابی قرار می‌دهد.

درصورتی که مشکلی در رابطه با داده‌های ورودی وجود داشته باشد برنامه مذکور فرم ورودی را به‌همراه اطلاعاتی که کاربر وارد کرده است مجدداً در اختیار وی قرار می‌دهد. انجام این فرآیند با بهره‌گیری از فیلدهای متن بسیار ساده‌است اما استفاده از لیست شامل داده‌های قابل انتخاب جذاب‌تر است. صفحاتی از وب که عهده‌دار نمایش اطلاعات استخراج شده از یک بانک اطلاعاتی هستند نیز با همین مشکل دست به گریبانند. در حالی‌که می‌توان اطلاعات ساده را به راحتی در فیلدهای متن فرمهای HTML وارد کرد برای وارد کردن تاریخ لازم است تا آن‌را به بخشهای تشکیل‌دهنده یعنی روز و ماه و سال تقسیم نمود.

کلاس date _ pulldown امکانات قابل توجهی را در رابطه با وارد کردن اطلاعات در اختیار کاربران قرار می‌دهد. این کلاس علاوه بر به‌خاطر سپردن انتخابهای صورت‌گرفته توسط کاربر از صفحه‌ای به صفحه دیگر، فرآیند انتخاب اجزای روز و سال و ماه از زمان را به‌راحتی ممکن می‌سازد.

جهت ایجاد کلاس، همان‌گونه که در درسهای قبلی عنوان کردیم، ابتدا باید آن‌را معرفی کنیم. سپس لازم است تا متد ویژه‌ای با عنوان متد سازنده را جهت نمونه‌گیری از کلاس ایجاد نماییم. در صورت‌تمایل می‌توانیم چند خصوصیت را نیز در درون کلاس معرفی کنیم. لیست ۷- ۱۵ ساختار کلاس date _ pulldown را نشان می‌دهد.

```
class date_pulldown {
   var $name;
   var $timestamp = -1;
   var $months = array("Jan", "Feb", "Mar", "Apr", "May", "Jun",
                       "Jul", "Aug", "Sep", "Oct", "Nov", "Dec");
   var $yearstart = -1;
   var $yearend = -1;

   function date_pulldown( $name ) {
          $this->name = $name;

   }
//...
```

لیست ۷- ۱۵ ساختار کلاس date _ pulldown

چنانکه مشاهده می‌کنید در این لیست ما خصوصیتی با نام $name را معرفی کرده‌ایم. این خصوصیت جهت نام‌گذاری عناصر select از فرم HTML ورودی مورد استفاده قرار خواهد گرفت. خصوصیت $timestampe برای نگهداری مقدار یک برچسب زمان استفاده خواهد شد. خصوصیت $months که یک آرایه است شامل دنباله‌های کاراکتری خواهد بود که در لیست مربوط به ماه از فرم HTML ورودی به نمایش در می‌آیند. همچنین دو خصوصیت $yearstart و $yearend که مقدار عددی 1 - به‌عنوان مقدار اولیه به آنها نسبت داده شده است، به‌ترتیب اولین و آخرین سال از محدوده‌ای از سالها را که در لیست مربوط به سال از فرم HTML ورودی به نمایش در می‌آید، ثبت خواهند کرد.

متد سازنده این کلاس ساختار بسیار ساده‌ای دارد. این متد یک دنباله کاراکتری را به‌عنوان

آرگومان ورودی دریافت می‌کند و آن‌را به خصوصیت $name نسبت می‌دهد. بدین ترتیب متد سازنده فوق خصوصیت $name از کلاس را مقداردهی می‌کند (چنانچه به خاطر دارید مقداردهی خصوصیات یک کلاس از جمله وظایف اصلی هر متد سازنده‌ای محسوب می‌شود).

اکنون که ساختار کلاس مورد نظرمان را تعریف کردیم به مجموعه‌ای از متدها نیاز داریم تا برنامه client به واسطه آنها بتواند با استفاده از مقادیر وارد شده توسط کاربر اقدام به تنظیم تاریخ نماید. لیست ۸- ۱۵ کد لازم جهت پیاده‌سازی این فرآیند را نشان می‌دهد.

```
// ...
      function setDate_global() {
            if( !$this - > setDate_array( $GLOBALS[ $this - >
name ] ) )
                  return $this -> setDate_timestamp( time()
);
            return true;
      }

      function setDate_timestamp( $time ) {
            $this -> timestamp = $time;
            return true;

      function setDateArray( $inputdata ) {
            if( is_array( $inputdate ) &&
                        isset( $inputdate[ 'mon' ] ) &&
                  isset( $inputdate'mday' ] ) &&
                  isset( $inputdate'year' ] ) ) {
                    $this -> timestamp = mktime( 11, 59, 59,
                              $inputdate[ 'mon' ], $inputdate[
'mday' ],
                              $inputdate[ 'year' ] );
                    return true;
            }

            return false;
      }
// ...
```

لیست ۸- ۱۵ کد مربوط به متدهای تنظیم‌کننده زمان

از میان متدهای تعریف شده در این لیست، متد ( ) setDate _ timestamp ساده‌ترین است. این متد یک برچسب زمان را به‌عنوان تنها آرگومان ورودی دریافت کرده و آن‌را به خصوصیت $timestamp از کلاس نسبت می‌دهد.

متد ( ) setDate _ array یک آرایه انجمنی را به‌عنوان آرگومان ورودی دریافت می‌کند. این آرایه انجمنی دست کم باید شامل سه کلید دستیابی با اسامی ' mon ', ' mday ' و ' year ' باشد. قالب

داده‌های ذخیره شده در این فیلدها مشابه قالب آرایه‌ای است که تابع ( ) getdate باز می‌گرداند. این بدان معنی است که متد ( ) setDate _ array آرایه‌ای را به‌عنوان آرگومان ورودی می‌پذیرد که به صورت زیر ایجاد شده باشد:

```
array (' mday ' ⇒ 5, ' mon ' ⇒ 7, ' year ' ⇒ 1999) ;
```

همچنین متد مذکور می‌تواند نتیجه فراخوانی تابع ( ) getdate را نیز به صورت زیر به عنوان آرگومان مورد استفاده قرار دهد (به‌خاطر بیاورید که تابع ( ) getdate یک برچسب زمان را به‌عنوان آرگومان ورودی پذیرفته و در عوض یک آرایه متشکل از اجزای تشکیل دهنده زمان را به‌عنوان خروجی باز می‌گرداند) :

```
SetDate _ array (getdate (931172399) ) ;
```

توجه کنید که ضرورت وجود فیلدهای ماه و روز و سال (یا ' mon ', ' mday ' و ' year ') در آرایه‌ای که به‌عنوان آرگومان متد ( ) setDate _ array آن‌را مورد استفاده قرار می‌دهیم، اتفاقی نبوده‌است. این متد از تابع دیگری با عنوان ( ) mktime که پیشتر آن‌را مورد بررسی قرار دادیم، جهت ایجاد یک برچسب زمان که در نهایت به متغیر $timestamp نسبت داده می‌شود، استفاده می‌کند.

و اما جالب است بدانید که متد ( ) setDate _ global به‌صورت پیش فرض فراخوانی می‌گردد. این متد سعی دارد تا یک متغیر سراسری همنام با خصوصیت $name از شیئ که از کلاس date_pulldown نمونه‌گیری شده است، پیدا کند. اگر چنانچه متد فوق در کار خود موفق باشد و بتواند متغیری سراسری با ساختار مورد نظرش را پیداکند، آن را به‌عنوان آرگومان ورودی متد setDate ( ) array_ مورد استفاده قرار می‌دهد تا متد مذکور قادر به ایجاد برچسب زمان موردنظر باشد. در چنین شرایطی متد ( ) setDate _ global مقدار true را باز می‌گرداند. متد فوق درصورت عدم موفقیت دریافتن ساختار موردنظر مقدار false را به برنامه فراخواننده باز خواهد گرداند.

اگر تا به حال درمورد ساختار و زمان دقت کافی به‌خرج داده باشید، به احتمال زیاد دریافته‌اید که محدوده تغییرات بخشهای روز و ماه از زمان ثابت است حال آنکه چنین مطلبی در مورد بخش سال از زمان صحت ندارد. از آنجا که اطلاع و اطمینان از یک محدوده مشخص می‌تواند بسیار مفید باشد در این قسمت متدهایی را ایجاد می‌کنیم که امکان تعیین محدوده سالی مشخصی را در اختیار برنامه‌ای که نمونه‌ای از کلاس date_ pulldown_ را مورد استفاده قرار می‌دهد، بگذارد (باز هم در این مورد چنانچه محدوده سالی توسط برنامه مشخص نشود از محدوده پیش‌فرض استفاده خواهد شد). کد مربوط به متدهای موردنیاز جهت انجام این فرآیند در لیست ۹- ۱۵ قابل بررسی است.

```
// ...
        function setYearStart( $year ) {
                $this -> yearstart = $year;
        }

        function setYearEnd( $year ) {
                $this -> yearend = $year;
        }

        function getYearStart() {
                if( $this -> yearstart < 0 ) {
                        $nowarray = getdate( time() );
                        $this -> yearstart = $nowarray[ 'year' ] - 5;
                }
                return $this -> yearstart;
        }

        function getYearEnd() {
                if( $this -> yearend < 0 ) {
                        $nowarray= getdate( time() );
                        $this -> yearend = $nowarray[ 'year' ] + 5;
                }
                return $this -> yearend;
        }
// ...
```

لیست ۹- ۱۵ تعریف متدهای موردنیاز جهت کنترل بخش مربوط به سال از یک زمان

همان‌گونه که در لیست مذکور مشاهده می‌کنید، ساختار متدهای ( ) setYearEnd و ( ) setYearStart کاملاً واضح و ساده است. هر دو متد فوق مقدار آرگومان دریافتی خود را به خصوصیتی از شیئ که از کلاس نمونه‌گیری شده است، نسبت می‌دهند. متد ( ) setYearStart خصوصیت $yearstart و متد ( ) setYearEnd نیز خصوصیت $yearend از شئ موردنظر را مقداردهی می‌کند. متد ( ) getYearStart بررسی را در مورد اینکه آیا خصوصیت $yearstart از شئ موردنظر مقداردهی شده است یا خیر صورت می‌دهد. درصورتی که این متغیر مقداردهی نشده باشد، متد مذکور مقدار آن‌را برابر با ۵ سال قبل از سال جاری قرار می‌دهد. متد دیگر یعنی ( ) getYearEnd نیز فرآیند مشابهی را انجام می‌دهد، بدین معنا که یک بررسی در مورد مقدار خصوصیت $yearend انجام می‌گیرد. درصورتی که خصوصیت فوق مقداردهی نشده باشد، مقدار آن‌را برابر با ۵ سال بعد از سال جاری قرار می‌دهد. با در دست داشتن این متدها اکنون آماده‌ایم تا بخش نهایی کلاس date _ pulldown را ارائه دهیم. این بخش از کد را می‌توانید در لیست ۱۰- ۱۵ مشاده کنید.

```
// ...
      function output() {
            if( $this -> timestamp < 0 )
                  $this -> setDate_global();
            $datearray = getdate( $this -> timestamp );
            $out = $this -> day-select( $this -> name,
$datearray );
            $out .= $this -> month_select( $this -> name,
$datearray );
            $out .= $this -> year_select( $this -> name,
$datearray );

            return $out;
      }

      function day_select( $fieldname, $datearray ) {
            $out = "<select name = \"$fieldname" . "[ mday
]\">\n";

            for( $x = 1; $x <=31; $x++ )
                  $out .= "<option value = \"$x\"" . (
$datearray[ 'mday' ] == ($x )
                        ? "SELECTED" : "" ) . "> " . sprintf(
"%02d", $x ) . "\n";
            $out .= "</select>\n";

            return $out;
      }

      function month_select( $fieldname, $datearray ) {
            $out = "<select name = \"$fieldname" . "[ mon
]\">\n";
            for( $x = 1; $x <=12; $x++ )
                  $out .= "<option value = ($x)\"" . (
$datearray[ 'mon' ] == ($x )
                        ? "SELECTED" : "" ) . "> " . $this ->
months[ $x - 1 ] . "\n";
            $out .= "</select>\n";

            return $out;
      }

      function year_select( $fieldname, $datearray) {
            $out = "<select name = \"$fieldname" . "[ year
]\">";
            $start = $this -> getYearStart();
```

لیست ۱۰-۱۵ بخش نهایی از کلاس date _ pulldown

```
            $end = $this -> getYearEnd();
            for( $x = $start; $x < $end; $x++ )
                  $out .= "<otion value = \"$x\"" . (
$datearray[ 'year' ] == ( $x )
                        ? " SELECTED" : "" ) . " $x\n";
            $out .= "</select>\n";
            return $out;
      }
}
```

دنباله لیست ۱۰-۱۵

همان‌گونه که در این لیست مشاهده می‌کنید، متد ( ) output بخش اعظم این کد را با فراخوانی سایر متدها هدایت می‌کند. این متد ابتدا مقدار خصوصیت $timestamp را مورد بررسی قرار می‌دهد. چنانچه برنامه کاربردی یکی از متدهای setDate شئ نمونه‌گیری شده از کلاس date _ pulldown را فراخوانی نکرده باشد، متد ( ) output از مقدار پیش‌فرض خصوصیت $timestamp که عدد 1 - است در عبارت شرطی if جهت ارزیابی استفاده کرده و درصورتی که این بررسی معادل با true ارزیابی شود، متد setDate _ global ( ) را فرا می‌خواند. پس از انجام این کار جهت ایجاد آرایه مورد نیاز سایر متدهایی که به‌زودی آنها را بررسی می‌کنیم، خصوصیت $timestamp به متد getdate ( ) ارسال شده و به ازای هریک از بخشهای ماه و روز سال متد دیگری فراخوانی می‌شود.

متد day _ select ( ) ساختار المان select از فرم HTML را شکل می‌دهد. این ساختار گزینه‌هایی را به‌ازای ۳۱ روز از ماه، جهت انتخاب در اختیار کاربر قرار می‌دهد. تاریخ فعلی که شئ نمونه‌گیری شده از کلاس date _ pulldown مشخص می‌کند، در آرایه $datearray ذخیره شده است. از این آرایه هنگام ایجاد المان select و تعیین گزینه انتخاب شده از این ساختار استفاده می‌شود. توجه کنید که در این متد از یک تابع سیستمی PHP با عنوان sprintf ( ) استفاده کرده‌ایم. این تابع قادر است تا عنصر روز از تاریخ را به‌صورت خاصی قالب‌بندی کند؛ بدین ترتیب که پیش از روزهای یکم تا نهم ماه که به‌ترتیب با اعداد 1 و 9 مشخص می‌شوند، رقم صفر را اضافه می‌کند تا قالب تمامی اعدادی که به روزهای ماه نسبت داده می‌شوند، همگن شود. در درس روز هفدهم با عنوان " بهره‌گیری از دنباله‌های کاراکتری" مطالب بیشتر و مفیدتری درباره تابع sprintf ( ) و چگونگی استفاده از آن فراخواهید گرفت. دو متد دیگر با اسامی month _ select ( ) و year _ select ( ) از روشهای مشابهی جهت ایجاد لیستهای مربوط به ماه و سال از فرم HTML بهره می‌گیرند.

با بررسی لیست اخیر ممکن است این پرسش به ذهنتان برسد که چرا کد مزبور را به‌جای اینکه تنها در یک بلوک ساده بنویسیم، به چهار متد جداگانه تقسیم کرده‌ایم؟ هنگامی که کلاسی را ایجاد می‌کنیم، همواره به خاطر داشته باشید که دو گروه مختلف از کاربران را باید درنظر بگیریم. گروه اول کاربرانی هستند که مایلند تا نمونه‌ای از کلاس date _ pulldown یا به‌عبارت دیگر شیئی از این کلاس

را در برنامه‌های خود استفاده کنند. گروه دوم را دسته‌ای تشکیل می‌دهندکه قصد دارند تا از کلاس موردنظر به‌عنوان سوپرکلاس در طراحی یک کلاس جدید به‌منظور بهبود عملکرد آن یا توسعه قابلیتها استفاده نمایند. برای کاربران گروه اول مایلیم تا رابطهای ساده و مؤثری را جهت بهره‌گیری از قابلیتهای تعبیه شده از کلاس دراختیار قرار دهیم. این‌گونه کاربران به‌راحتی می‌توانند از کلاس date _ pulldown اشیایی را نمونه‌گیری کرده و پس از تنظیم تاریخ موردنظرشان از طریق خصوصیات آن اشیا با فراخوانی متد ( ) output اقدام به نمایش تقویم موردنظر خود نمایند. کاربران این گروه نیازی به اطلاع از ساختار درونی کلاس نداشته و این ساختار از دید آنها پنهان می‌ماند. درمورد کاربران گروه دوم، مایلیم تا تسهیلاتی را در رابطه با تغییر اجزای مختلف کلاس و توسعه قابلیتهای آن در اختیار قرار دهیم. با تقسیم عملکرد کلاس در رابطه با نمایش خروجی به چند متد مختلف کلاسهایی که از کلاس date _ pulldown به‌عنوان سوپرکلاس استفاده می‌کنند، می‌توانند بدون اینکه تاثیری بر کل روند خروجی بگذارند تغییراتی را به‌طور دلخواه به بخشهای مختلف آن اعمال نمایند. برای نمونه، اگر کلاس فرزند مایل باشد تا به‌جای استفاده از لیست امکان انتخاب بخش سال از تقویم را با استفاده از دکمه‌های رادیویی‌دراختیارکاربران‌خودقرار دهد، برنامه‌نویس تنها کافی است تا متد ( ) year _ select از سوپرکلاس را در کلاس فرزند رونویسی نماید. این درصورتی است که اگر عملکرد خروجی کلاس date pulldown _ تنها به یک متد منتهی می‌شد، اعمال چنین تغییری در خروجی یا هر تغییر کوچکی در آن مستلزم تغییرعمده‌ای‌در متد مذکور می‌شد که این به نوبه خود امکان خطا را در عملکرد کلاس گسترش می‌داد.

برنامه موجود در لیست ۱۱- ۱۵ نمونه‌ای از کاربرد این کلاس را نشان می‌دهد.

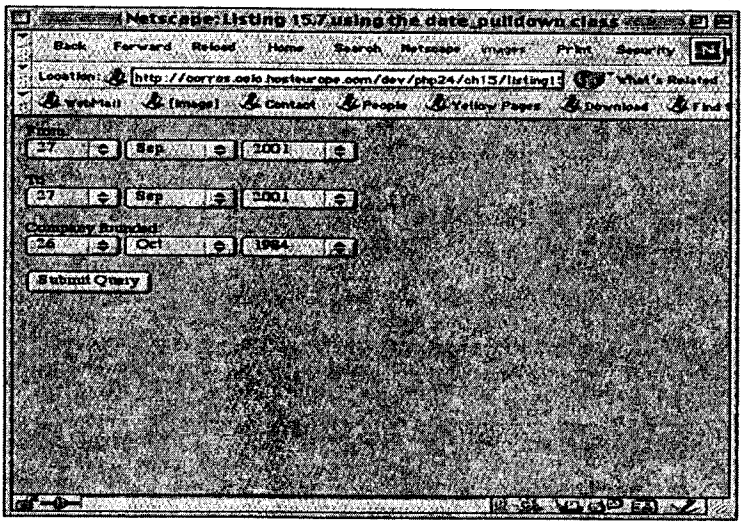
```
1: <html>
2: <head>
3: <title>Listing 15.7 Using the date_pulldown Class</title>
4: </head>
5: <?php
6: include("date_pulldown.class.php");
7: $date1 = new date_pulldown("fromdate");
8: $date2 = new date_pulldown("todate");
9: $date3 = new date_pulldown("foundingdate");
10: $date3->setYearStart(1972);
11: if ( empty( $foundingdate ) )
12:     $date3->setDate_array( array( 'mday'=>26, 'mon'=>4, 'year'=>1984 ) );
13: ?>
14: <body>
15:
16: <form>
17: From:<br>
18: <?php print $date1->output( ); ?><p>
19:
20: To:<br>
```

لیست ۱۱- ۱۵ بهره‌گیری از کلاس date _ pulldown

```
21: <?php print $date2->output( ); ?><p>
22:
23: Company founded:<br>
24: <?php print $date3->output( ); ?><p>
25:
26: <input type="submit" valu="do it">
27: </form>
28:
29: </body>
30: </html>
```

دنباله لیست ۱۱- ۱۵

همان‌گونه که در این لیست مشاهده می‌کنید ما کلاس نهایی date _ pulldown را در یک فایل کتابخانه‌ای با نام date _ pulldown. Class. php ذخیره کرده و با استفاده از تابع ( ) include در خط ۶ مورد دستیابی قرار داده‌ایم. برای تمام نمونه‌های کلاس فوق رفتار و قابلیت پیش‌فرض کلاس را مورد استفاده قرار داده و تنها درمورد شئ ' foundingdate ' این عملکرد پیش‌فرض را رونویسی کرده‌ایم. چنانکه می‌بینید این شئ تاریخ شروع سال را با بهره‌گیری از عبارت موجود در خط ۱۰ سال ۱۹۷۲ میلادی فرض کرده‌است. علاوه بر این، در خط ۱۲ برنامه تاریخ دلخواهی را برای شئ درنظر گرفته‌ایم. این تاریخ یعنی روز دوازدهم از ماه اکتبر سال ۱۹۸۴ تاریخ پیش‌فرضی است که چنانچه کاربر آن‌را تغییر ندهد پس از ارسال فرم به وب سرور دراختیار برنامه قرار خواهد داد. خروجی حاصل از این برنامه را می‌توانید در شکل ۴- ۱۵ مشاهده کنید.



شکل ۴- ۱۵ لیست‌های ایجاد شده با استفاده از کلاس date _ pulldown

جمع‌بندی

در درس این ساعت چگونگی بهره‌برداری از تابع ( ) time را جهت دستیابی به برچسب زمانی مربوط به تاریخ و ساعت جاری مورد بحث قرار داده و توابع مفید دیگری را در مورد تاریخ بررسی کردیم. در این درس نحوه استفاده از تابع ( ) getdate را جهت استخراج اطلاعات مربوط به بخشهای مختلف مربوط به تاریخ و ساعت از برچسب زمان، همچنین استفاده از تابع ( ) date جهت تبدیل برچسب زمان به یک دنباله کاراکتری قالب‌بندی شده را مورد بحث قرار دادیم. طی این درس شما با چگونگی ایجاد برچسب زمان با استفاده از تابع ( ) mktime را فراگرفتید. همچنین با نحوه ارزیابی یک تاریخ مشخص با استفاده از تابع ( ) checkdate آشنا شدید. در درس این ساعت از طریق بررسی یک برنامه نمونه که توابع بحث شده در این ساعت را مورد استفاده قرار می‌داد و همچنین ایجاد یک کلاس عمومی و استفاده از آن در قالب یک کتابخانه توانستیم کاربرد بسیار جالب توجهی از آنچه را که در این درس فراگرفتید در اختیار شما قرار دهیم.

در درس ساعت بعد با ورود به دنیای انواع داده‌های قابل استفاده در برنامه‌های PHP مطالب بسیار مفیدی را که در طراحی خیلی از برنامه‌ها مورد استفاده قرار می‌گیرند، فراخواهید گرفت.

پرسش و پاسخ

پرسش: آیا در زبان PHP تابعی برای تبدیل تقویمها به یکدیگر وجود دارد؟

پاسخ: بله. زبان PHP مجموعه کاملی از توابع مفید را جهت تبدیل انواع تقویمها به یکدیگر ارائه کرده است. در صورت تمایل به مطالعه بیشتر درباره این توابع می‌توانید بخش مربوطه از مستندات PHP در آدرس زیر را مورد دستیابی قرار دهید:

http: // www. php. net / manual / ref. calendar. php

تمرینها

هدف از این بخش آرایه تمرینهایی در قالب آزمون است. پاسخ بخش آزمون بلافاصله بعد از آن آمده است. بخش فعالیتها شامل تمرینهایی است که به منظور افزایش قابلیت برنامه‌نویسی خواننده طراحی شده که البته فاقد پاسخ لازم است.

آزمون

۱- چگونه می‌توان به برچسب زمان مربوط به تاریخ و ساعت فعلی در برنامه‌های PHP دسترسی پیدا کرد؟

۲- کدام تابع در زبان PHP یک برچسب زمان را به‌عنوان آرگومان ورودی دریافت و یک آرایه انجمنی شامل بخشهای زمانی مختلف مربوطه را باز می‌گرداند؟
۳- از کدام تابع جهت قالب‌بندی اطلاعات زمانی استفاده می‌شود؟
۴- از کدام تابع در زبان برنامه‌نویسی PHP می‌توان جهت دستیابی به برچسب زمان معادل یک زمان دلخواه استفاده کرد؟
۵- از کدام تابع در زبان PHP می‌توانیم جهت ارزیابی یک تاریخ مشخص استفاده کنیم؟

پاسخ آزمون

۱- تابع ( ) time تاریخ فعلی را در قالب یک برچسب زمان باز می‌گرداند.
۲- تابع ( ) getdate یک آرایه انجمنی باز می‌گرداند که عناصر آن‌را بخشهای مختلف یک تاریخ مشخص (که از طریق آرگومان ورودی به این تابع ارسال شده است) تشکیل می‌دهند.
۳- از تابع ( ) date می‌توان جهت قالب‌بندی یک تاریخ استفاده نمود.
۴- تابع ( ) mktime با دریافت بخشهای مختلف یک زمان مشخص یعنی ساعت، دقیقه، ثانیه، ماه، روز و سال برچسب زمان متناظر با آن‌را به برنامه فراخواننده باز می‌گرداند.
۵- با استفاده از تابع ( ) checkdate می‌توان تاریخ مشخصی را مورد ارزیابی قرار داد.

فعالیتها

۱- برنامه‌ای ایجاد کنید که از طریق یک فرم ورودی اجزای زمانی مختلف مربوط به یک تاریخ تولد مشخص، شامل ماه و روز و سال را دریافت کند. این برنامه پس از پردازش ورودی‌ها باید بتواند مدت زمان باقیمانده تا آن تاریخ را برحسب روز، ساعت، دقیقه و ثانیه در خروجی نمایش دهد.

ساعت شانزدهم

بهره‌گیری از انواع داده‌ها

در درس این ساعت قصد ما این است که مبحث مربوط به ارزیابی و دستکاری داده‌ها را با تفضیل بیشتری مورد بررسی قرار دهیم. در این ساعت باردیگر نگاهی به انواع داده‌های قابل استفاده در برنامه خواهیم داشت. هرچند که PHP خود مسائل مربوط به کنترل انواع داده‌ها را انجام می‌دهد اما اطلاع از نحوه انجام این فرآیند توسط PHP بسیار مفید خواهد بود به‌ویژه اگر قصد شما از مطالعه این کتاب ایجاد برنامه‌های کاربردی online با قابلیت و کارآیی باشد. همچنین در این درس مبحث آرایه‌ها را مجدداً مورد بررسی قرار داده و برخی از ویژگی‌های پیشرفته‌ای از زبان PHP را که امکانات مناسبی را جهت دستکاری و مرتب‌سازی آنها در اختیار برنامه‌نویس قرار می‌دهد، بررسی خواهیم نمود.
در درس این ساعت مباحث زیر تحت بررسی قرار می‌گیرند:

- **چگونگی تبدیل انواع داده‌ها به یکدیگر**
- **چگونگی تبدیل خودکار داده‌ها توسط PHP در عبارات مختلف**
- **بررسی روشهای مختلف ارزیابی انواع داده‌ها**
- **دلایل اهمیت اطلاع برنامه‌نویس از انواع داده‌های مورد استفاده در برنامه**
- **چگونگی اطلاع از مقداردهی شدن متغیرهای برنامه**
- **بررسی روش دیگری جهت پردازش عناصر یک آرایه**
- **بررسی وجود یک عنصر بخصوص در آرایه**
- **چگونگی تغییر شکل و فیلتر کردن مقادیر ذخیره شده در یک آرایه**
- **چگونگی مرتب‌سازی عناصر یک آرایه**

در ادامه به بررسی هریک از این موارد می‌پردازیم:

نگاهی مجدد به انواع داده‌ها

در درس ساعت چهارم با عنوان " بلوکهای سازنده برنامه‌های PHP " با جزئیات مربوط به انواع داده‌ها در PHP آشنا شدید. با این حال مطالب بیشتری در این رابطه ناگفته مانده است. در این قسمت از درس قصد ما این است تا توابع بیشتری را در مورد تشخیص نوع داده ذخیره شده در یک متغیر از برنامه بررسی کنیم. همچنین در این قسمت به بررسی شرایطی می‌پردازیم که تحت آن PHP خود اقدام به تبدیل انواع داده‌ها به یکدیگر می‌کند.

یادآوری

با مطالبی که تاکنون در درسهای گذشته فراگرفتید، اکنون می‌دانید که متغیرهای مورد استفاده در برنامه‌های PHP می‌توانند مقادیری از نوع عددی (شامل اعداد صحیح و اعشاری)، دنباله‌های کاراکتری، مقادیر boolean (شامل دو مقدار true و false)، اشیا و آرایه‌ها (شامل آرایه‌های عددی و انجمنی) را ذخیره کنند. همچنین در درسهای قبل مطالبی را نیز درمورد دو نوع داده بخصوص یعنی NULL و مراجعی که به منظورهای مختلف از آنها استفاده می‌کنیم، فراگرفتید. به کمک تابع ( ) gettype می‌توانیم نوع داده هر متغیری از برنامه را تشخیص دهیم. این تابع ساختار ساده‌ای دارد به‌گونه‌ای که تنها به یک مقدار (از هرنوع) یا یک متغیر که مقداری از هر نوع را در خود ذخیره کرده است، به‌عنوان آرگومان ورودی نیاز دارد. خروجی تابع ( ) gettype نوع داده یا نوع متغیری است که مقدار موردنظر را در خود ذخیره کرده است. قطعه کد زیر به‌سادگی چگونگی بهره‌گیری از این تابع را جهت تشخیص متغیری از نوع عدد صحیح نشان می‌دهد:

```
$data = 454;
print gettype ($data) ;
// prints "integer"
```

همان‌گونه که در درسهای قبل مشاهده کردید، تبدیل نوع داده‌ها از یک نوع به نوع دیگر با بهره‌گیری از تکنیک ویژه‌ای موسوم به casting یا استفاده از تابع ( ) settype امان‌پذیر است. جهت بهره‌گیری از تکنیک casting کافی است تا قبل از نام متغیر حاوی مقدار موردنظر یا خود آن مقدار نام نوع داده موردنظرتان (نوع داده مقصد) را در درون یک جفت پرانتز ذکر نمایید. توجه داشته باشید که استفاده از این تکنیک به‌هیچ عنوان محتوای متغیر مورد استفاده یا نوع داده آن‌را تغییر نمی‌دهد. اتفاقی که طی این فرآیند رخ می‌دهد این است که PHP یک کپی از متغیر یا مقدار مورد استفاده در فرآیند را با نوع داده خواسته شده (نوع داده‌ای که نام آن در درون جفت پرانتز ذکر می‌شود) به‌دست می‌دهد. قطعه کد زیر با استفاده از این تکنیک نوع داده کپی متغیری با نام $data از نوع اعشاری را به

نوع عدد صحیح تبدیل می‌کند. باردیگر خاطرنشان می‌کنیم که در این فرآیند نوع داده متغیر $data تغییر نمی‌کند:

```
$data = 4.333 ;
print (integer) $data ;
// prints 4
```

پس از این فرآیند متغیر $data کماکان شامل عدد صحیحی است که پیشتر به آن نسبت داده شده بود. دستورالعمل ( ) print در این قطعه کد صرفاً مقدار بازگشتی از عملیات casting (یعنی مقدار ذخیره شده در متغیر $data با نوع داده عدد صحیح) را در خروجی چاپ خواهد کرد.

همان‌گونه که در پاراگراف قبل و نیز در درسهای گذشته عنوان کردیم روش دیگر تبدیل نوع داده ذخیره شده در یک متغیر، استفاده از تابعی با عنوان ( ) settype است. این تابع دو آرگومان را به‌عنوان ورودی دریافت می‌کند. آرگومان اول نام یک متغیر و آرگومان دوم نام نوع داده مقصد است. کاری که این تابع انجام می‌دهد به‌سادگی تبدیل نوع داده آرگومان اول به‌نوعی است که آرگومان دوم مشخص می‌کند. قطعه کد زیر چگونگی استفاده از این تابع را نشان می‌دهد:

```
$data = 4.333 ;
settype ($data, integer) ;
print $data ;
// prints 4
```

همان‌گونه که مشاهده می‌کنید، تابع ( ) settype در قطعه کد بالا نوع داده متغیر $data را که شامل یک عدد اعشاری می‌باشد به نوع عدد صحیح تبدیل می‌کند. توجه داشته باشید که این تابع برخلاف تکنیک casting عملیات تبدیل را بر روی کپی متغیر انجام نداده بلکه خود متغیر تحت تاثیر این تبدیل قرار می‌گیرد. تابع ( ) settype از این نظر یک تابع منحصر به فرد در زبان برنامه‌نویسی PHP محسوب می‌شود.

تبدیل انواع پیچیده‌تر

تا بدین‌جا نحوه تبدیل انواع ساده‌ای از داده‌ها (مانند انواع عددی اعشاری، نوع داده boolean و دنباله‌های کاراکتری) را به‌طور تقریباً مفصل مورد بررسی قرار دادیم. همان‌گونه که در فرآیند مربوط به این تبدیلات مشاهده نمودید، مسأله چندانی درمورد تبدیل این انواع ساده به یکدیگر نداشتیم. حال سؤال این است که آیا می‌توانیم این انواع ساده را به انواع داده پیچیده‌تری همچون اشیا و آرایه‌ها تبدیل کنیم؟ بحث عمده ما در این قسمت از درس بررسی این‌گونه تبدیلات و حل وفصل مسائل جانبی ناشی از این تبدیلات است.

هنگامی که نوع داده ساده‌ای مانند عدد صحیح یا اعشاری را به یک آرایه تبدیل می‌کنیم، حاصل تبدیل آرایه‌ای است که تنها یک عضو داشته و آن عضو همان داده‌ای است که اقدام به تبدیل آن نموده‌ایم (توجه کنید که در تبدیلات انواع پیچیده‌تر از تابع ( ) settype استفاده نکرده و تنها

تکنیک casting را مورد بهره‌برداری قرار می‌دهیم). قطعه کد ساده زیر فرآیندی از این نوع تبدیل را که عبارت از تبدیل یک دنباله کاراکتری به یک آرایه است، نشان می‌دهد:

```
$str = "this is my string" ;
$arr _ str = (array) $str ;
print $arr _ str [0] ;
// print "this is my string "
```

همچنین توجه کنید که هنگام تبدیل یک عدد صحیح یا اعشاری یا یک دنباله کاراکتری به یک شئ (باز هم استفاده از تکنیک casting)، شیئ ایجاد می‌شود که فاقد هرگونه متدی بوده و تنها دارای یک خصوصیت است. نام این خصوصیت همواره scalar بوده و شامل عدد صحیح یا اعشاری یا دنباله کاراکتری است که فرآیند تبدیل بر روی آن انجام شده است (توجه کنید که در زبان PHP به مانند زبان برنامه‌نویسی Perl به انواع ساده داده‌ها که شامل اعداد صحیح و اعشاری، نوع داده boolean و دنباله کاراکتری می‌شود اسکالر گفته می‌شود). قطعه کد ساده زیر فرآیند تبدیل یک دنباله کاراکتری را به یک شئ با نام $arr _ str نشان می‌دهد:

```
$str = "this is my string" ;
$arr _ str = (object) $str ;
Print $arr _ str → scalar ;
// print "this is my string"
```

موضوع فرآیند تبدیل هنگامی جالب‌تر می‌شود که بخواهیم مابین آرایه‌ها و اشیا تبدیل انجام دهیم. هنگامی که آرایه‌ای را به یک شیئ تبدیل می‌کنیم حاصل تبدیل شیئ خواهد بود که فاقد متد بود و اسامی خصوصیات آن‌را کلیدهای دستیابی آرایه تشکیل می‌دهند. یک نمونه از این‌گونه تبدیل را درمورد یک آرایه شاخص‌گذاری شده با استفاده از دنباله‌های کاراکتری (آرایه انجمنی) در قطعه کد زیر مشاهده می‌کنید (دقت کنید که مقادیر خصوصیات شیئ حاصل را مقادیر عناصر ذخیره شده در آرایه مورد استفاده در فرآیند تبدیل تشکیل می‌دهند):

```
$addresses = array (street ⇒ "williams Street" ,
                    town ⇒ "Stockton") ;
$obj _ addresses = (object) $addresses ;
print $obj _ addresses  → street ;
// print "williams Street"
```

به‌طور معکوس فرآیند تبدیل یک شیی (نمونه‌ای از یک کلاس) به یک آرایه موجب ایجاد آرایه‌ای می‌شود که عناصر آن‌را مقادیر خصوصیات شیئ و کلیدهای دستیابی آن‌را نیز اسامی این خصوصیات تشکیل می‌دهند. در این تبدیل از وجود متدهای موجود در شیی موردنظر صرف‌نظر می‌شود. قطعه کد زیر چگونگی تبدیل نمونه‌ای از یک کلاس با نام Point را که شامل دو خصوصیت با اسامی $x و $y و متدی با نام ( ) Point است به آرایه‌ای با نام $array _ point نشان می‌دهد:

```
Class Point {
   var $x ;
   var $y ;
```

```
    function Point ($x, $y) {
        $this → x = $x ;
        $this → y = $y ;
    }
}
$point = new Point (5 , 7) ;
$array _ point = (array) $point ;
print $array _ point ['x'] ;
// print 5
```

تبدیل خودکار انواع داده‌ها به یکدیگر

اگر عبارتی که در یک برنامه استفاده می‌کنید شامل دو عملوند از دو نوع داده مختلف باشد، PHP جهت محاسبه نتیجه عملیات ابتدا یکی از آن دو عملوند را به دیگری تبدیل کرده و پس از این تبدیل عملیات مورد نظر را انجام می‌دهد. در این فرآیند هیچ‌گونه مبدلی به‌کار گرفته نشده و از هیچ روش و تکنیکی نیز جهت تبدیل انواع داده‌ها به یکدیگر استفاده نمی‌شود. آنچه اتفاق می‌افتد این است که PHP یک تبدیل ضمنی بر مبنای قوانینی که طراحان این زبان برنامه‌نویسی از پیش تدوین کرده‌اند، انجام می‌دهد. به‌احتمال قوی تاکنون بدون اینکه متوجه شده باشید بارها و بارها چنین تبدیلی را مورد بهره‌برداری قرار داده‌اید. به‌عنوان مثال متغیرهایی از برنامه که عهده‌دار نگهداری مقادیر وارد شده در فیلدهای متن یک فرم HTML هستند همواره شامل مقادیری از نوع دنباله کاراکتری می‌باشند و ما از این متغیرها در عبارات منطقی جهت تصمیم‌گیری یا محاسبات ریاضی جهت حصول نتیجه موردنظر به‌گونه‌ای استفاده می‌کنیم که گویی درحال بهره‌گیری از متغیرهای عددی هستیم. در ادامه به بررسی چند نمونه از این تبدیلات ضمنی می‌پردازیم.

فرض کنید با ایجاد یک فرم HTML شرایطی را دراختیار کاربر قرار می‌دهیم تا بتواند تعداد ساعاتی را که طی یک هفته از شبکه استفاده کرده است، از طریق یک فیلد متن از یک فرم مشخص نماید. ما این مقدار ورودی را که ماهیت آن یک دنباله کاراکتری است در متغیری با نام $hours ذخیره کرده و آن‌را جهت پردازشهای بعدی حفظ می‌کنیم. قطعه کد زیر شرایط فوق را به‌تصویر کشیده است:

```
$hours = "15" ;
if ( $hours = = 15 )
   print "As a frequent user you may qualify for a discount. " ;
```

در قطعه کد بالا، مقدار دریافتی از فرم ورودی را در قالب یک ساختار if با عدد صحیح 15 مقایسه کرده‌ایم. تعجبی ندارد اگر بدانیم که PHP پیش از انجام یک چنین مقایسه‌ای با بهره‌گرفتن از تکنیک casting اقدامی را جهت تبدیل نوع مقدار ورودی به نوع عدد صحیح انجام می‌دهد. پس از انجام تبدیل مقایسه یک کپی از مقدار ورودی که اکنون به عدد صحیح 15 تبدیلْ شده است با عدد صحیح 15، نتیجه true درپی خواهد بود.

همان‌گونه که حدس می‌زنید، قوانین مربوط به تبدیلات ضمنی تقریباً ساده هستند. هنگام کار با مقادیر عددی (اعداد صحیح و اعشاری) و دنباله‌های کاراکتری همواره این دنباله‌های کاراکتری هستند که به نوع عددی متناظر تبدیل می‌شوند. درصورتی که یک دنباله کاراکتری با یک عدد آغاز شود کل آن دنباله کاراکتری به این عدد تبدیل شده و حاصل تبدیل در محاسبات شرکت خواهد کرد. عبارت زیر را که حاصل‌ضرب عدد صحیح 4 در یک دنباله کاراکتری از این نوع می‌باشد را درنظر بگیرید:

```
4 * "20mb" ;
```

حاصل این عبارت با توجه به تبدیل ضمنی دنباله کاراکتری " 20mb " به عدد صحیح 20 برابر با عدد صحیح 80 خواهد بود. درصورتی که دنباله کاراکتری موردنظر با یک عدد آغاز نشود، ضمن عملیات به عدد صحیح صفر تبدیل خواهد شد. برای نمونه عبارت زیر را که حاصل‌ضرب عدد صحیح 4 و دنباله کاراکتری "about 20mb" را محاسبه می‌کند، درنظر بگیرید:

```
4 * "about 20mb" ;
```

همان‌گونه که در عبارت فوق مشاهده می‌کنید از آنجا که دنباله کاراکتری مورد استفاده در عملیات با یک عدد آغاز نشده است، پیش از عمل ضرب معادل با عدد صفر فرض شده و بنابراین حاصل عملیات نیز برابر با عدد صفر خواهد شد.

درصورتی که دنباله کاراکتری مورد استفاده در عملیات با یک عدد صحیح و به‌دنبال آن یک نقطه‌اعشاری‌آغاز شود، معادل با یک عدد اعشاری فرض خواهد شد. عملیات ضرب زیر که حاصل‌ضرب عدد صحیح 4 و دنباله کاراکتری "1.2" است را درنظر بگیرید. توجه کنید که این دنباله کاراکتری در شرایط ذکر شده صدق می‌کند:

```
4 * "1.2" ;
```

از آنجا که دنباله کاراکتری مورد استفاده در عملیات ضرب از عبارت فوق با یک عدد اعشاری آغاز شده است، پیش از انجام عملیات معادل عدد اعشاری 1.2 فرض شده و بدین ترتیب حاصل عملیات عدد اعشاری 4.8 خواهد بود.

نکته جالب توجهی که باید در این میان ذکر کرد این است که عملگرهای افزایش و کاهش مقادیر عددی (+ + - -) هنگام اعمال به دنباله‌های کاراکتری نیازمند صرف دقت بیشتری از جانب برنامه‌نویس هستند؛ چراکه بر مبنای محتوای دنباله کاراکتری مورد استفاده تأثیرهای مختلفی از خود برجای می‌گذارند.

درصورتی که دنباله کاراکتری که یکی از عملگرهای افزایش یا کاهش به آن اعمال شده است تنها شامل اعداد و ارقام باشد، عملیات موردنظر طبق انتظار انجام می‌شود. به عبارت دیگر اعمال عملگر افزایش به چنین دنباله‌ای از کاراکترها موجب افزایش مقدار عددی متناظر به میزان یک واحد می‌شود. همچنین اعمال عملگر کاهش به این دنباله کاراکتری موجب کاهش مقدار عددی متناظر به میزان یک

واحد می‌شود. توجه کنید که پس از انجام عملیات کاهش یا افزایش درمورد یک چنین دنباله‌ای از کاراکترها ماهیت نتیجه عملیات کماکان دنباله‌ای از کاراکترها خواهد بود. به‌عنوان مثال در این زمینه قطعه کد زیر را که عملگر افزایش را به متغیری که شامل دنباله کاراکتری "4" است اعمال می‌کند، درنظر بگیرید:

```
$str = "4" ;
$str + + ;
print $str ; // prints 5
print gettype ($str); // prints "string"
```

درصورتی که عملگر افزایش به دنباله کاراکتری که شامل حروف باشد اعمال شود، این عملگر تنها به آخرین کاراکتر موجود در دنباله اعمال می‌گردد. قطعه کد زیر نمونه‌ای از این فرآیند را نشان می‌دهد. در این قطعه کد عملگر افزایش به متغیر $str که شامل دنباله کاراکتری "hello " است، اعمال شده است. بدین ترتیب تنها کاراکتر 'o' تحت تاثیر این عملگر قرار گرفته و دنباله کاراکتری "hello " حاصل می‌شود. این بدان دلیل است که با افزایش یک واحدی کد ASCII مربوط به کاراکتر' o '، کاراکتر بعدی یعنی ' p ' حاصل خواهد شد:

```
$str = "hello" ;
$str + + ;
print $str ;   // prints "hellp"
```

خروجی حاصل از قطعه کد فوق را با نتیجه حاصل از اجرای قطعه کد زیر مقایسه کنید. در قطعه کد زیر سعی شده است تا بدون بهره‌گرفتن از عملگر افزایش و تنها با اضافه کردن یک واحد به دنباله کاراکتری "hello " همان نتیجه فوق به دست می‌آید. همان‌گونه که مشاهده می‌کنید، این اقدام با شکست مواجه شده و آنچه به دست می‌آید عدد صحیح 1 می‌باشد:

```
$str = "hello" ;
$str + = 1 ;
print $str ; // prints 1;
print gettype ($str) ;  // prints "integer"
```

احتمالاً متوجه این نکته شده‌اید که مقدار متغیر $str پیش از شرکت در عملیات ریاضی جمع تبدیل به عدد صحیح صفر می‌شود؛ چراکه دنباله کاراکتری مربوطه به عدد آغاز نشده است. از این رو حاصل این عملیات عدد صحیح 1 می‌باشد که در نهایت در متغیر $str ذخیره می‌شود. خروجی تابع ( ) gettype تایید می‌کند که متغیر $str حاوی یک عدد صحیح است.

تبدیل ضمنی مابین دو نوع داده عدد صحیح و عدد اعشاری از این هم ساده‌تر و سرراست‌تر است. چنانچه در عبارتی یکی از عملوندها از نوع اعشاری و دیگری از نوع عدد صحیح باشد، پیش از انجام هر عملی ابتدا عملوند نوع صحیح به نوع اعشاری معادل تبدیل شده و سپس عملیات موردنظر انجام می‌شود. نتیجه حاصل از چنین فرآیندی همان‌گونه که حدس می‌زنید از نوع اعشاری خواهد بود (دقت کنید که این‌گونه تبدیل نیز مشابه آنچه که پیش‌تر شاهد بودید به روش casting صورت

می‌گیرد. به‌عبارت دیگر، نوع متغیر اصلی دست نخورده باقی‌مانده و تنها کپی آن است که پس از تبدیل به نوع اعشاری در محاسبات مورد استفاده قرار می‌گیرد). قطعه کد زیر نمونه‌ای از چنین تبدیل ضمنی را از نوع صحیح به اعشاری نشان می‌دهد:

```
$result = (1 + 20.0) ;
print gettype ($result) ;
// print "double"
```

باردیگر خاطرنشان می‌کنیم که انجام تبدیلات ضمنی توسط PHP به قصد ارزیابی عبارات موجود در برنامه هیچ‌گونه تأثیری بر روی خود آن مقادیری که در عملیات شرکت می‌کنند نداشته و تنها کپی آنها دستخوش تغییر می‌شوند. بدین ترتیب پس از انجام عملیات، اصل مقادیر کماکان در دسترس برنامه‌نویس قرار خواهد داشت (البته به‌شرطی که برنامه به نوعی آنها را رونویسی نکرده باشد).

روش ارزیابی نوع داده‌ها

تاکنون نحوه ارزیابی یا به عبارت دیگر نحوه تشخیص نوع داده متغیرهای برنامه یا مقادیر مورد استفاده در برنامه را با بهره‌گیری از تابع ( ) gettype مشاهده نمودید. تشخیص نوع متغیرها از دیدگاه اشکال‌زدایی برنامه بسیار ارزشمند است چراکه بدین ترتیب برنامه‌نویس از نوع داده‌های ذخیره شده در یک متغیر یا نوع مقداری که در برنامه مورد استفاده قرار می‌گیرد به طور دقیق اطلاع حاصل می‌کند. با این همه شاید در اغلب موارد برنامه‌نویس نیازمند یک چنین دقت بالایی نبوده و تنها اطلاع از این موضوع که آیا متغیر یا مقدار موردنظر از یک نوع خاص می‌باشد یا خیر برای وی کافی باشد. در زبان برنامه‌نویسی PHP توابع مفیدی متناسب با هرنوع داده مورد استفاده در این زبان برای تشخیص مطلب فوق پیش‌بینی شده است. تمامی این توابع متغیر یا مقداری را به‌عنوان آرگومان ورودی دریافت کرده و یکی از دو مقدار true یا false را بسته به اینکه متغیر یا مقدار تحت بررسی از نوع داده موردنظر است یا خیر باز می‌گرداند. جدول ۱- ۱۶ اسامی هریک از این توابع را به‌همراه جزئیات مربوطه نشان می‌دهد.

جدول ۱- ۱۶ توابع مورد استفاده در PHP جهت ارزیابی نوع داده متغیرها و مقادیر مورد استفاده در برنامه

نام تابع	توضیح
is _ array ( )	این تابع درصورتی که آرگومان ورودی یک آرایه باشد، مقدار true را باز می‌گرداند.
is _ bool ( )	این تابع درصورتی که آرگومان ورودی از نوع boolean (یکی از مقادیر true یا false) باشد، مقدار true را باز می‌گرداند.

is _ double ( )	این تابع درصورتی‌که آرگومان ورودی از نوع اعشاری باشد مقدار true را باز می‌گرداند.
is _ int ( )	این تابع درصورتی که آرگومان ورودی از نوع عدد صحیح باشد، مقدار true را باز می‌گرداند.
is _ object ( )	این تابع درصورتی‌که آرگومان ورودی یک شئ (نمونه‌ای از یک کلاس) باشد، مقدار true را باز می‌گرداند.
is _ string ( )	این تابع درصورتی که آرگومان ورودی دنباله‌ای از کاراکترها باشد، مقدار true را باز می‌گرداند.
is _ null ( )	این تابع درصورتی که آرگومان ورودی پوچ باشد(مانند متغیری که هنوز مقداردهی نشده باشد)، مقدار true را باز می‌گرداند.
is _ resource ( )	این تابع درصورتی که آرگومان ورودی یک مرجع باشد، (مانند مرجعی به یک رنگ خاص یا مرجعی به یک فایل یا فهرست موجود بر روی سیستم) مقدار true را باز می‌گرداند.

همان‌گونه که مشاهده می‌کنید این توابع فرآیند تشخیص یا به‌طور دقیق‌تر ارزیابی نوع داده‌ها را به‌راحتی ممکن می‌سازند. به قطعه کد زیر که با استفاده از تابع ( ) gettype اقدام به این کار می‌کند، توجه نمایید:

```
if (gettype ($var) = = "array" )
   print "it's an array" ;
```

تأثیر قطعه کد فوق معادل قطعه کد زیر است که با بهره‌مندی از تابع ( ) is _ array از جدول ۱- ۱۶ بازنویسی شده است.

```
if (is _ array ($var) )
   print "it's an array" ;
```

چنانچه ملاحظه می‌کنید اصول برنامه‌نویسی در روش اخیر به مراتب بهتر از روش اول رعایت شده است. درهر دو روش درصورتی که متغیر $var شامل یک آرایه باشد، پیغامی در خروجی چاپ خواهد شد.

بررسی سایر روشها برای تغییر نوع داده‌ها

تا بدین‌جای درس با دو روش متداول جهت تغییر انواع داده‌ها به یکدیگر آشنا شدید. همان‌گونه که به‌خاطر دارید در روش اول از تکنیک casting و در روش دوم از تابع ( ) settype استفاده کردیم؛ با این توضیح که در روش casting اصل متغیر یا مقدار موردنظر دست‌نخورده باقی‌مانده و تنها کپی آن دچار تغییر نوع می‌شود. در زبان برنامه‌نویسی PHP علاوه بر این روشها توابعی نیز جهت مقادیر از انواع عدد صحیح، عدد اعشاری و دنباله کاراکتری به یکدیگر پیش‌بینی شده است. این توابع آرگومانهایی غیر از دو نوع آرایه‌ای و شیئی را به عنوان ورودی پذیرفته و بسته به تابع مقدار عدد صحیح، عدد اعشاری یا دنباله کاراکتری متناظر با آن‌را باز می‌گردانند. جدول ۲- ۱۶، اسامی هریک از این توابع را به‌همراه توضیحی در مورد هرکدام از آنها نشان می‌دهد. به جهت سادگی از ارائه مثال در این مورد خودداری شده است.

جدول ۲- ۱۶ توابعی جهت تبدیل انواع داده‌ها به یکدیگر

نام تابع	توضیح
double val ( )	این تابع متغیر یا مقداری را به عنوان آرگومان ورودی پذیرفته و عدد اعشاری متناظر با آن‌را به برنامه فراخواننده باز می‌گرداند.
int val ( )	این تابع متغیر یا مقداری را به عنوان آرگومان ورودی پذیرفته و عدد صحیح متناظر با آن‌را به برنامه فراخواننده باز می‌گرداند.
strval ( )	این تابع متغیر یا مقداری را به‌عنوان آرگومان ورودی پذیرفته و دنباله کاراکتری متناظر با آن‌را به برنامه فراخواننده باز می‌گرداند.

اهمیت انواع داده‌ها در برنامه نویسی

هنگامی که برنامه‌نویس متغیری را در برنامه معرفی می‌کند، PHP به‌هیچ وجه وی را ملزم به تعیین نوع داده آن متغیر نمی‌کند. گذشته از این هنگام استفاده از انواع مختلفی از متغیرهای برنامه در یک عبارت PHP درصورت نیاز اقدامات لازم را جهت تبدیل نوع ضمنی آن انجام می‌دهد. حال این پرسش مطرح می‌شود که اگر زبان برنامه‌نویسی PHP تا بدین حد فرآیند نوشتن برنامه‌ها و استفاده از متغیرهایی با انواع داده مختلف را ساده کرده است، پس اطلاع از نوع داده متغیرهای مورد استفاده در برنامه چه اهمیتی برای برنامه‌نویس ممکن است داشته باشد؟

داشتن اطلاع کافی درباره نوع داده مقادیری را که متغیرهای برنامه ذخیره می‌کنند کمک شایانی برای جلوگیری خطا در برنامه می‌باشد. برای روشن شدن اهمیت این مطلب فرض کنید مشغول

نوشتن تابعی هستید که می‌تواند کلیدهای دستیابی و مقادیر متناظر با هریک از آنها در یک آرایه انجمنی را بر روی پنجره مرورگر اینترنت نمایش دهد. از آنجا که PHP در مورد آرگومانهای مورد استفاده در توابع سخت‌گیری بیش از حد از خود نشان نمی‌دهد، بنابراین نیازی نیست تا هنگام تعریف تابع مورد نظرتان متغیری با نوع داده مورد نیاز تابع به‌عنوان آرگومان در دسترس شما باشد. برای روشن شدن مطلب به مثال زیر که قطعه کدی برای تعریف تابعی با نام print Array است، توجه کنید:

```
function printArray ($array) {
   foreach ($array as $key ⇒ $val)
      print "$key : $val < p >" ;
}
```

درصورتی که آرگومان مورد نیاز این تابع که یک آرایه است از جانب برنامه فراخواننده تامین شود تابع مذکور عملیات پیش‌بینی شده را به‌خوبی و بدون مواجه شدن با هرگونه مشکلی انجام خواهد داد. برای نمونه عبارت زیر تابع فوق را با آرایه‌ای که شامل سه عنصر از نوع عدد صحیح است، فراخوانی می‌کند:

```
PrintArray (array (4,4,55) ) ;
```

حال اگر برنامه‌نویس بدون توجه به نوع داده موردنیاز تابع به‌عنوان آرگومان متغیری اسکالر (عددصحیح،عدداعشاری یا دنباله‌ای از کاراکترها) را مورد استفاده قرار دهد، بدون شک تابع printArray ( ) خطایی را آشکار خواهد کرد. عبارت زیر نمونه‌ای از یک فراخوانی نادرست تابع printArray ( ) است که با شکست مواجه شده است چراکه از یک عدد صحیح به جای یک آرایه به‌عنوان آرگومان استفاده شده است:

```
PrintArray (4) ;
// warning : Non array argument supplied for foreach ( ) in
//  / home / matt / htdocs / php _ book / data / test 2. php on line 5
```

به‌واسطه بررسی نوع داده آرگومان ارسالی به تابع، می‌توانیم از بروز مشکلات و خطاهای عدم سازگاری انواع داده‌ها جلوگیری کنیم. برای نمونه در مورد تابع printArray ( ) که تعریف آن‌را در پاراگراف‌های قبل ارائه دادیم می‌توانیم ترتیبی دهیم تا حتی درصورت عدم تامین داده مورد نظر تابع، باز هم تابع مقداری را به برنامه فراخواننده بازگرداند. در قطعه کد زیر تعریف تابع printArray ( ) را به‌گونه‌ای بازنویسی کرده‌ایم که در صورت ارسال یک مقدار اسکالر به تابع به‌عنوان آرگومان ورودی تابع مذکور، مقدار false را به برنامه فراخواننده بازگرداند:

```
Function printArray ($array) {
   if ( ! is _ array ($array) )
      return false ;
   foreach ($array as $key  ⇒ $val)
      print "$key : $val < p >" ;
   return true ;
}
```

همان‌گونه که مشاهده می‌کنید برنامه فراخواننده این تابع می‌تواند به‌راحتی با ارزیابی مقدار بازگشتی از این تابع (true یا false) از موفقیت‌آمیز بودن عملیات انجام شده توسط این تابع اطلاع حاصل نماید.

برنامه‌نویس می‌تواند حتی پا را از این نیز فراتر نهاده و با بهره‌گیری از تکنیک casting در صورت لزوم، اقدام به تبدیل مقادیر اسکالر به آرایه نماید. در قطعه برنامه زیر که بازنویسی دیگری از تابع ( ) printArray است این امکان در اختیار برنامه‌نویس قرار می‌گیرد که حتی با ارسال یک مقدار اسکالر به تابع نیز عملیات مورد درخواست با موفقیت انجام شده و نتیجه true به برنامه فراخواننده بازگردانده شود. این قابلیت انعطاف بالا به‌واسطه این حقیقت جالب به‌دست می‌آید که تابع مورد بحث درصورت دریافت مقدار اسکالر به عنوان آرگومان، آن را به یک آرایه تبدیل کرده و پس از این تبدیل با بهره‌مندی از یک ساختار تکرار عملیات عادی خود را از سر می‌گیرد:

```
function printArray ($array) {
   if ( ! is _ array ($array) )
      $array = (array) $array ;
   foreach ($array as $key ⇒ $val)
      print "$key : $val < p >" ;
   return true ;
}
```

چنانکه ملاحظه می‌کنید تابع ( ) printArray اکنون ساختار بسیار قابل انعطافی دارد، به‌گونه‌ای که می‌تواند با دریافت هرنوع ورودی (حتی از نوع شئ یا نمونه‌هایی از یک کلاس) اقدام به ایجاد آرایه‌ای متناظر با ورودی دریافتی کرده و کلیدهای دستیابی به‌همراه مقادیر مربوطه از این آرایه را در خروجی نمایش دهد.

بررسی و ارزیابی نوع داده متغیرها و مقادیر مورد استفاده در برنامه علاوه بر پیشگیری از خطاهای احتمالی، هنگام ارزیابی مقادیر بازگشتی از توابع نیز مفید واقع می‌شوند. در برخی از زبانهای برنامه‌نویسی مانند C و Java همواره نوع مقدار بازگشتی از یک تابع از پیش مشخص می‌شود (این عمل هنگام تعریف ساختار تابع یا متد موردنظر صورت می‌گیرد). زبان PHP چنین اجباری را به برنامه‌نویس تحمیل نمی‌کند. با این وجود چنین وضعیتی در زبان PHP در برخی موارد ممکن است منجر به گمراهی برنامه‌نویس در رابطه با نوع مقدار بازگشتی شود.

پیشتر در درس ساعت دهم با عنوان " بهره‌گیری از فایل‌ها " با یک چنین وضعیتی هنگام بررسی یک برنامه نمونه مواجه شدیم. چنانکه به‌خاطر دارید، تابع ( ) readdir درصورتی که به انتهای فهرستی که در حال خواندن محتوای درون آن است برسد مقدار false را به برنامه فراخواننده بازمی‌گرداند. تابع مذکور در سایر موارد یک دنباله کاراکتری را باز می‌گرداند که شامل نام فایل یا فهرست خوانده شده است. طبق آنچه که معمول است و ما بارها در درس مربوطه انجام داده‌ایم، می‌توانیم از ساختاری مشابه قطعه کد زیر جهت پردازش محتوای یک فهرست استفاده کنیم:

```
$dh = opendir ("mydir") ;
while ($name = readdir ( $dh) )
   print "$name < br >" ;
closedir ($dh) ;
```

با این وجود در این مورد نکته ظریفی ناگفته مانده که در اینجا مطرح می‌کنیم. درصورتی که نام فهرست مورد بررسی 0 (عدد صفر) باشد، عبارت منطقی موجود در ساختار while به‌صورت false ارزیابی می‌گردد و بدین ترتیب عملیات بدون پردازش محتوای فهرست پایان می‌یابد. با ارزیابی نوع داده مقدار بازگشتی از تابع ( ) readdir، می‌توانیم به شکل جالبی مسأله را به صورت زیر حل و فصل نماییم:

```
$dh = opendir ("mydir") ;
while (is _ string ($name = readdir ($dh) ) )
   print "$name < br >" ;
closedir ($dh) ;
```

بررسی بیشتر در مورد متغیرها

زبان برنامه‌نویسی PHP قابلیت انعطاف فوق‌العاده‌ای را در رابطه با اسامی متغیرها در اختیار برنامه‌نویس قرار می‌دهد. به‌عنوان یک نمونه از این قابلیت می‌توان دنباله کاراکتری مورد استفاده جهت نام‌گذاری یک متغیر را عیناً جهت نام‌گذاری متغیری دیگر مورد بهره‌گیری قرار داد. از این رو با درنظر گرفتن فرآیند نسبت‌دهی مقداری به یک متغیر به صورت زیر:

```
$user = "bob" ;
```

می‌توان دو عبارت مجزا به‌صورت زیر نوشت که با عبارت نسبت‌دهی فوق دقیقاً معادل باشند:

```
$holder = "user" ;
$$holder = "bob" ;
```

همان‌گونه که ملاحظه می‌کنید متغیر $holder شامل دنباله کاراکتری "user" است. از این‌رو می‌توان متغیر $$holder را معادل یک علامت $ دانست که به دنبال آن مقدار متغیر $holder، یعنی همان دنباله کاراکتری واقع شده است. بدین ترتیب عبارت PHP $$holder را معادل با $user فرض خواهد کرد (به عبارت ساده $holder معادل "user" است و اولین عبارت فوق نیز گویای همین واقعیت است).

کاربرد این قابلیت محدود به نام‌گذاری متغیرها نبوده بلکه به‌هنگام دستیابی به مقادیر متغیرها صدق می‌کند.

دو عبارت زیر را در همین رابطه درنظر بگیرید:

```
$user = "bob" ;
print $user ;
```

چنانکه ملاحظه می‌کنید، طبق روش معمول با استفاده از نام متغیر $user توانستیم مقدار ذخیره شده در آن‌را به کمک تابع سیستمی ( ) print نمایش دهیم. نکته جالب توجه این است که به‌جای‌دو عبارت فوق و با بهره‌گرفتن از سه عبارت زیر می‌توانیم نتیجه کاملاً مشابهی را به دست آوریم:

```
$user = "bob" ;
$holder = "user" ;
print $$holder ;
```

با وجود این باز هم PHP قادر است تا بیشتر متعجبتان کند، بدین ترتیب که اگر در چنین شرایطی به جای $$holder دنباله کاراکتری" $$holder" را مورد دستیابی قرار دهیم، احتمالاً نتیجه برای شما دور از ذهن خواهد بود. قطعه کد زیر را در همین رابطه درنظر بگیرید:

```
$user = "bob" ;
$holder = "user" ;
print "$$holder" ;
```

به احتمال قوی آنچه را که شما در خروجی حاصل از اجرای قطعه کد بالا انتظار داشته‌اید، دنباله کاراکتری " bob " یعنی مشابه خروجی قبلی بوده است. اما این مرتبه آنچه که در خروجی ظاهر می‌شود یک علامت $ و به‌دنبال آن دنباله کاراکتری " user " است که با هم تشکیل دنباله کاراکتری "$user" را می‌دهند. بنابراین از مطالب عنوان شده می‌توان این قاعده سرراست را نتیجه گرفت که هرگاه نام متغیری در درون کوتیشن (جفت علامت " ") مورد استفاده قرار بگیرد، PHP آن‌را معادل با مقدار آن متغیر فرض خواهد کرد. در مورد مثال ما PHP به‌سادگی نام متغیر $holder را از آنجا که در درون علامت کوتیشن مورد استفاده قرار گرفته است معادل با دنباله کاراکتری "user" فرض کرده و به‌همین علت حاصل عملیات چیزی جز دنباله کاراکتری "$user" نیست. به‌عبارت دیگر اولین علامت $ بدون هیچ گونه تغییری در جای خود باقی مانده است. اما حتی در این مورد نیز روشی برای قانع کردن PHP وجود دارد تا عبارت موجود در داخل علامت کوتیشن را به‌جای نام متغیر به‌صورت مقدار آن متغیر فرض نماید، یعنی شرایط به‌همان ترتیبی شود که هنگام عدم استفاده از علامت کوتیشن شاهد آن بودیم. ترفند مورد استفاده در این مورد بهره‌گرفتن از جفت علامت { } است، بدین صورت که کافی است تا عبارت بعد از اولین علامت $ را در درون این جفت علامت قرار دهیم. قطعه کد زیر با بهره‌گیری از این ترفند نتیجه مورد نظر را به دست می‌آورد:

```
$user = "bob" ;
$holder = "user" ;
print "$ { $holder}" ;
```

همان‌گونه که مشاهده می‌کنید در آخرین عبارت از قطعه کد فوق به‌دلیل اینکه $holder در درون جفت علامت { } قرار گرفته است، مجموعه {$holder } توسط PHP به‌صورت user فرض خواهد شد. درگام بعدی از آنجا که نام متغیر مورد استفاده در درون علامت کوتیشن معادل با مقدار آن

متغیر فرض می‌شود، بنابراین نتیجه به دست آمده از مرحله قبل یعنی "$user" به‌سادگی به‌صورت دنباله کاراکتری "bob" که درحقیقت مقدار متغیر $user است، فرض خواهد شد.

استفاده از مراجع متغیرها

بنا به پیش‌فرض فرآیند مقداردهی متغیرها با انتساب مقداری به نام آنها صورت می‌گیرد. این مقدار ممکن است مستقیماً و یا از طریق مقدار ذخیره شده در سایر متغیرها به متغیر موردنظر نسبت داده شود. به بیان روشن‌تر هنگامی که متغیری مثلاً با نام $aVariable را به متغیر دیگری مانند $anotherVariable نسبت می‌دهیم، یک کپی از مقداری که در حال حاضر در متغیر $avariable ذخیره شده است، به متغیر$another Variable نسبت داده می‌شود. بعد از این هرگونه تغییری که بر روی مقدار یکی از این متغیرها صورت بگیرد هیچ‌گونه تاثیری بر روی مقدار متغیر دیگر نخواهد داشت. به‌عبارت بهتر این دو متغیر پس از مقداردهی کاملاً مستقل از یکدیگرند.

زبان برنامه‌نویسی PHP4 امکاناتی را در اختیارمان قرار می‌دهد که به‌راحتی بتوانیم این رفتار را تغییر دهیم. در واقع می‌توانیم ترتیبی دهیم تا در مثال قبلی به‌جای نسبت‌دهی یک کپی از مقدار ذخیره شده در متغیر $aVariable به متغیر $anotherVariable مرجعی را که دربردارنده آدرس متغیر $aVariable در حافظه است، به متغیر $anotherVariable نسبت داده شود. برای انجام این کار کافی است تا از علامت & که به علامت آدرس‌دهی شهرت دارد درست پیش از نام متغیر مرجع در فرآیند نسبت‌دهی استفاده نماییم. به قطعه کد زیر که چگونگی نسبت‌دهی متغیرها با استفاده از مرجع را نشان می‌دهد، توجه کنید:

```
$aVariable = 42 ;
$anotherVariable = &$aVariable ;
$aVariable= 325 ;
print $anotherVariable ;   // print 325
```

همان‌گونه‌که در این قطعه کد مشاهده می‌کنید، در عبارت اول متغیر $aVariable با عدد صحیح 42 مقداردهی شده است. عبارت دوم نیز یک فرآیند مقداردهی است، بدین ترتیب که مقدار متغیر $aVariable با بهره‌گیری از علامت & به متغیر $anotherVariable نسبت داده می‌شود. به‌واسطه‌وجودعلامت&دراین‌فرآیند نسبت دهی می‌توان چنین تصور کرد که متغیر $anotherVariable دنباله روی متغیر $aVariable یا دقیق‌تر نام دیگری برای متغیر $aVariable است. بنابراین هرگونه تغییری‌درمقدار متغیر $aVariable موجب تأثیر دقیقاً مشابهی بر روی مقدار متغیر $anotherVariable خواهد شد. به موجب این وابستگی شدید برنامه‌نویس باید دقت زیادی را هنگام استفاده از این‌گونه متغیرها به‌خرج دهد. البته PHP امکانی را نیز برای قطع این وابستگی پیش‌بینی کرده که همان استفاده از تابع ( ) unset است. برای این‌کار کافی است تا یکی از این دو متغیر را به‌عنوان آرگومان به تابع مذکور ارسال کنیم. تابع ( ) unset بلافاصله متغیری را که نام آن‌را به‌عنوان آرگومان دریافت کرده

است از میان برده اما هرگونه مرجعی را که در رابطه با آن متغیر ایجاد شده باشد، دست‌نخورده باقی می‌گذارد.

بررسی وجود و خالی بودن یک متغیر از مقدار

همان‌گونه که تاکنون شاهد بودید بررسی متغیرهای برنامه جهت تشخیص نوع داده مقادیر ذخیره شده در آنها از بسیاری جهات مفید است، اما پیش از انجام هرگونه بررسی در این زمینه مناسب است تا از وجود متغیرها و در گام بعدی از مقداردهی شدن آنها اطمینان حاصل کنیم. واضح است که درصورت عدم وجود متغیر یا عدم مقداردهی آن نمی‌توان نوع داده مربوطه را معین نمود. با استفاده از تابع ویژه‌ای با عنوان ( ) isset در PHP می‌توان از وجود متغیرها در برنامه اطمینان حاصل کرد (منظور از وجود متغیر در برنامه معرفی آن است که ممکن است همراه با مقداردهی نیز باشد). تابع ( ) isset تنها به یک آرگومان ورودی نیاز دارد. این آرگومان نام یک متغیر است که وضعیت مقداردهی آن توسط تابع بررسی می‌شود. درصورتی که متغیر موردنظر شامل یک مقدار باشد، تابع ( ) isset مقدار true و در غیر این‌صورت مقدار false را به برنامه فراخواننده باز می‌گرداند. قطعه کد زیر چگونگی استفاده از این تابع را جهت ارزیابی متغیری با نام $notset نشان می‌دهد:

```
$notset ;
if (isset ($notset) )
   print " \ $notset is set" ;
else
   print " \ $notset is not set" ;
// prints "$notset is not set"
```

همان‌گونه که در قطعه کد بالا مشاهده می‌کنید متغیر $notset بدون اینکه مقداردهی شود، معرفی شده است. از آنجا که این متغیر فاقد مقدار می‌باشد تابع ( ) isset در دومین عبارت مقدار false را باز می‌گرداند بنابراین بخش else از ساختار تصمیم‌گیری if / else از کد فوق اجرا می‌شود. نتیجه اینکه دنباله کاراکتری "$notset is not set" به‌عنوان خروجی بر روی صفحه به‌نمایش در می‌آید. از مطلب فوق این نکته مهم استنباط می‌شود که چنانچه متغیری معرفی شده ولی هنوز مقداری به آن نسبت داده نشده باشد، بررسی تابع ( ) isset در مورد آن مقدار false را در پی خواهد داشت.

حتی با دانستن نکته فوق باز هم احتمال اشتباه وجود دارد. این اشتباه می‌تواند درمورد دو مقدار صفر و دنباله کاراکتری " " به‌راحتی گریبان‌گیر تازه کارها شود. چنانچه متغیری با یکی از این دو مورد مقداردهی شده باشد، بررسی انجام شده توسط ( ) isset در مورد آن متغیر موجب بازگشت مقدار true از تابع فوق خواهد شد. در قطعه کد زیر متغیر $notset این‌بار با استفاده از یک دنباله کاراکتری خالی (" ") مقدار دهی شده است:

```
$notset = " " ;
if (isset ($notset) )
   print " \ $notset is set" ;
else
   print " \ $notset is not set" ;
// prints "$notset is set"
```

همان‌گونه که مشاهده می‌کنید، به‌دلیل مقداردهی متغیر $notset با دنباله کاراکتری خالی (" ") این بار نتیجه متفاوتی به دست آمده و بخش if از ساختار تصمیم‌گیری مورد اجرا قرار می‌گیرد.

متغیرهایی که به‌واسطه ارسال یک فرم HTML به وب سرور مقداردهی می‌شوند در بررسی تابع ( ) isset درمورد آنها همواره منجر به نتیجه true خواهند شد. این وضعیت حتی درصورتی که کاربر هیچ مقداری را در فیلدهای مربوطه از فرم وارد نکرده باشد، بازهم صدق می‌کند. بدین ترتیب لازم است تا پیش از استفاده از این متغیرها در محاسبات مختلف از خالی یا تهی بودن این متغیرها از مقدار اطلاع حاصل کنیم. برای انجام این کار به‌راحتی می‌توانیم از تابع دیگری از PHP با عنوان empty ( ) استفاده کنیم. تابع ( ) empty نام متغیر مورد بررسی را به‌عنوان آرگومان ورودی دریافت می‌کند. درصورتی که این متغیر مقداردهی نشده باشد تابع مذکور مقدار true را باز می‌گرداند. تابع مورد بحث ویژگی جالب توجهی دارد که آن‌را از تابع ( ) isset متمایز می‌کند؛ این تابع درصورتی که متغیر دریافتی به‌عنوان آرگومان حاوی مقدار صفر یا دنباله کاراکتری تهی (" ") باشد، باز هم آن‌را خالی فرض کرده و مقدار true را به برنامه فراخواننده باز می‌گرداند. تابع ( ) empty بسیار دقیق عمل می‌کند چراکه بررسی یک آرایه تهی نیز با استفاده از این تابع منجر به مقدارtrue می‌گردد. قطعه کد زیر از این تابع مفید جهت بررسی متغیر $notset استفاده می‌کند:

```
$notset = " " ;
if (empty ($notset) )
   print " \ $notset is empty" ;
else
   print " \ $notset contains data" ;
// prints "$notset is empty"
```

همان‌گونه که در اولین عبارت از این قطعه کد مشاهده می‌کنید، متغیر $notset با دنباله کاراکتری تهی " " مقداردهی شده است. به این جهت حاصل عبارت منطقی ساختار if که از تابع ( ) empty جهت ارزیابی این متغیر بهره می‌گیرد مقدار true بوده و بنابراین دنباله کاراکتری "$notset is empty" به‌عنوان خروجی بر روی صفحه به نمایش در می‌آید.

بررسی مطالب بیشتر درباره آرایه‌ها

در درس ساعت هفتم با عنوان " آرایه‌ها " به‌طور مفصل درمورد آرایه‌ها و برخی از توابعی که می‌توانیم از آنها جهت تسهیل عملیات مورد نظرمان بر روی آرایه‌ها استفاده کنیم، بحث و بررسی

کردیم. در این قسمت از درس این ساعت مایلیم تا در ادامه این بررسی توابع و تکنیکهای بیشتری را درمورد کار با آرایه‌ها مورد مطالعه قرار دهیم.

روشی دیگر جهت پردازش عناصر یک آرایه

در زبان برنامه‌نویسی php4 چنانکه می‌دانید ساختار تکرار foreach به‌طور خاص جهت پردازش عناصر آرایه طراحی شده است. این ساختار در نوع خود ابزاری کارآمد و پرفایده بوده و همان‌گونه که تاکنون شاهد آن بودید ما نیز جهت پردازش آرایه‌ها در مثالهای مطرح شده در این کتاب این ساختار تکرار را به خدمت گرفتیم. از آنجا که این ساختار مختص PHP4 می‌باشد در مورد برنامه‌های نوشته شده با استفاده از PHP3 به اجبار باید از روش دیگری جهت پردازش آرایه‌ها استفاده کنیم. اگر مجبورید برنامه‌های PHP خود را به‌گونه‌ای بنویسید که با PHP3 سازگار باشد یا علاقه‌مندید تا با مطالعه کد منبع PHP و به‌ویژه آن دسته از کدهایی که پیش از PHP4 منتشر شده‌اند دانش برنامه‌نویسی خود در مورد این زبان کارآمد را افزایش دهید، به‌طور حتم باید با تکنیکی که در این قسمت جهت پردازش آرایه‌ها مطرح می‌کنیم، آشنا باشید.

هنگامی که آرایه‌ای را ایجاد می‌کنید، در درون PHP اتفاقات جالبی رخ می‌دهد. از دیدگاه سیستمی به موازات ایجاد آرایه توسط برنامه‌نویس، PHP لیستی را جهت نگهداری عناصر آرایه سازمان داده و اشاره‌گری را ایجاد می‌کند که به اولین عنصر این لیست اشاره می‌کند. PHP تابعی با عنوان ( ) each در اختیار برنامه‌نویس قرار می‌دهد که با استفاده از آن می‌توان به کلید دستیابی و مقدار متناظر با آن دسترسی داشته باشد. این تابع ساختار نسبتاً ساده‌ای دارد. آرگومان ورودی تابع ( ) each یک آرایه است. همچنین آنچه که این تابع به‌عنوان نتیجه عملیات به برنامه فراخوان باز می‌گرداند، آرایه ساده‌ای است که از چهار عضو تشکیل شده است. نکته بسیار جالب در مورد آرایه بازگشتی از تابع ( ) each این است که آرایه مذکور یک آرایه هیبرید (مرکب) است، بدین صورت که دو عضو اول این آرایه با استفاده از اعداد صحیح صفر و یک و دو عضو بعدی نیز با استفاده از کلیدهای دستیابی (دنباله‌های کاراکتری) " key " و " value " شاخص‌گذاری شده‌اند. به بیان دیگر نیمی از این آرایه‌ها از نوع عددی و نیم دیگر از نوع انجمنی است. پس از هربار فراخوانی تابع ( ) each ، اشاره‌گر لیست به عنصر بعدی موجود در آرایه اشاره خواهد کرد. این وضعیت تا بدانجا ادامه پیدا می‌کند که اشاره‌گر آخرین عضو موجود در لیست را مورد اشاره قرار می‌دهد. در این لحظه چنانچه تابع ( ) each فراخوانی شود به‌جای آرایه هیبرید مذکور مقدار false به عنوان نتیجه عملیات تابع به برنامه فراخواننده باز می‌گردد. اجازه دهید تا جهت روشن‌تر شدن مطلب مثال ساده‌ای را مورد بررسی قرار دهیم. در قطعه کد زیر ابتدا یک آرایه با نام $details ایجاد شده و سپس با بهره‌گیری از تابع ( ) each سعی شده تا عناصر این آرایه مورد پردازش قرار گیرد:

```
$details = array (school ⇒ "Slickly High", course ⇒
        "Selective Indifference") ;
$element = each ($details) ;
print "$element [0] < BR >" ;    // prints "school"
print "$element [1] < p >" ;     // prints "Slickly High"
print "$element [ 'key' ]. "< BR >" ;    // prints "school"
print "$element [ 'value' ]. "< BR >" ;    // prints "Slickly High"
```

همان‌گونه که در این قطعه کد مشاهده می‌کنید ابتدا یک آرایه انجمنی با نام $details که شامل تنها دو عنصر است، ایجاد شده است. در گام بعدی تابع ( ) each فراخوانی شده و نام آرایه فوق به‌عنوان آرگومان ورودی به این تابع ارسال شده‌است. نتیجه بازگشتی حاصل از عملیات تابع که یک آرایه چهارعضوی است در متغیری با نام $element ذخیره می‌شود. طبق آنچه که پیشتر در مورد آن بحث کردیم، اکنون می‌توانیم چنین ادعا کنیم که پس از اجرای این دو فرآیند کلید دستیابی و مقدار اولین عضو موجود در آرایه $details در آرایه چهارعضوی $element ذخیره شده است.

چنانچه ملاحظه می‌کنید نسبت دادن آرایه بازگشتی از تابع ( ) each به یک متغیر و استفاده از آن در سایر عملیات اندکی پرزحمت و شاید هم کمی گیج کننده است. خوشبختانه زبان PHP برای چنین مواقعی تابعی را پیش‌بینی کرده است تا اندازه زیادی زحمت دستیابی به عناصر این‌گونه آرایه‌ها را کاهش می‌دهد. نام این تابع ( ) list است. تابع ( ) list نام هر تعدادی از متغیرها را به‌عنوان آرگومانهای ورودی دریافت کرده و هریک از آنها را با مقدار متناظر از یک آرایه که در سمت راست تساوی مشخص می‌شود، مقداردهی می‌کند. برای روشن‌تر شدن مطلب فوق قطعه کد زیر را که نحوه استفاده از تابع ( ) list را نشان می‌دهد، درنظر بگیرید. همان‌گونه که مشاهده می‌کنید در این کد از آرایه عددی برای نمایش قابلیت ( ) list بهره گرفته‌ایم:

```
$array = array (44, 55, 66, 77) ;
list ($first, $second) = $array ;
print "$first" ;        // prints 44
print "< BR >" ;
print "$second" ;    // prints 55
```

توجه کنید که با بهره‌گرفتن از تابع ( ) list چگونه توانسته‌ایم دو عنصر اول از آرایه $array را جهت پردازش به‌ترتیب در متغیرهای $first و $second ذخیره کنیم (همان‌گونه که متوجه شده‌اید در مورد ( ) list از واژه تابع استفاده کرده‌ایم. حقیقت این است که ( ) list را نمی‌توان به مانند توابع عادی موجود در زبان PHP مانند تابع ( ) empty یا ( ) isset یک تابع فرض کرد. در واقع ( ) list به‌واسطه جفت پرانتزی که بعد از آن قرار گرفته است به تابع می‌ماند، اما در عمل جزء ابزاری برای مقداردهی توأم چندین متغیر از روی مقادیر یک آرایه بیش نیست).

بدین ترتیب ( ) list کاربردی‌های مفیدی را جهت استفاده پیش روی برنامه‌نویس قرار می‌دهد. یکی از کاربردهای مفید این ابزار به احتمال قوی کاربردی است که در رابطه با تابع ( ) each مطرح

می‌شود. با استفاده از ابزار ( ) list می‌توانیم با هربار فراخوانی تابع ( ) each دو متغیر مختلف به‌طور توأم مقداردهی نماییم. پس از این عمل قادر خواهیم بود تا به مانند آنچه که در رابطه با ساختار تکرار foreach شاهد بودیم از این متغیرها جهت پردازش استفاده کنیم. قطعه کد زیر با بهره‌گیری از ( ) list اقدام لازم جهت ذخیره مقادیر ذخیره شده در آرایه $details را در متغیرهای $key و $value صورت می‌دهد. فرآیند تا جایی تکرار می‌شود که عناصری جهت پردازش در این آرایه وجود داشته باشند:

```
$details = array ( school ⇒ "Slickly High" ,
                   course ⇒ "Selective Indifference" ) ;
while ( list ($key, $value) = each ( $details ) )
   print "$key: $value < BR >" ;
```

با وجودی که قطعه کد بالا در عمل کار می‌کند اما باید اعتراف کنیم که مورد خاصی را در این میان فراموش کرده‌ایم. این نکته ظریف عبارت از آن است که چنانچه آرایه $details پیش‌تر به همین شیوه مورد پردازش قرار گرفته باشد اشاره‌گر درونی PHP که در ابتدای کار به اولین عنصر آرایه اشاره می‌کند در انتهای پردازش آرایه به آن‌سوی آخرین عضو موجود در آرایه اشاره خواهد کرد. به‌عبارت دیگر برای آنکه این اشاره‌گر وظیفه خود را به خوبی انجام دادهٔ و همواره به موقعیت مناسب اشاره کند لازم است تا به روشی به ابتدای آرایه تحت بررسی اشاره نماید (به جایگاه اولیه خود بازگردد). خوشبختانه بازهم در زبان PHP چنین فرآیندی با بهره‌گیری از یک تابع قابل پیاده‌سازی است. تابعی که بدین منظور در PHP طراحی شده است ( ) reset نام دارد. تابع مذکور ساختاری بی‌اندازه ساده داشته و تنها به نام آرایه موردنظر به‌عنوان آرگومان نیاز دارد.

به‌عنوان جمع‌بندی ساختار آشنای ( ) foreach را به صورت زیر جهت پردازش عناصر آرایه مورد استفاده قرار می‌دهیم:

```
Foreach ( $details as $key ⇒ $value )
   Print "$key : $value < BR >" ;
```

با بهره‌گیری از ( ) list و ( ) reset می‌توانیم به شکل زیر بنویسیم:

```
reset ( $details ) ;
while ( list  ( $key, $value ) = each ( $details ) )
   print "$key : $value < BR >" ;
```

بررسی وجود یک متغیر خاص در آرایه

پیش از انتشار PHP4 برای اطلاع از اینکه آیا مقدار خاصی در یک آرایه موجود می‌باشد یا خیر، برنامه‌نویسان مجبورند تا به روشی تمامی عناصر آرایه را با آن مقدار موردنظر مقایسه کنند. این فرآیند تا هنگامی ادامه پیدا می‌کرد که یا مقدار موردنظر در درون آرایه یافت شود و یا اینکه بدون حصول نتیجه‌ای فرآیند پردازش عناصر موجود در آرایه به انتها رسد. خوشبختانه در این مورد نیز PHP4 ابزار مورد نیاز برنامه‌نویس را جهت اطلاع از مطلب فوق در اختیار وی قرار می‌دهد. در این مورد ابزار مورد

نیاز تابعی با نام in _ array ( ) است. تابع in _ array ( ) جهت انجام عملیات خود به دو آرگومان ورودی نیاز دارد. اولین آرگومان این تابع مقدار مورد جستجو و دومین آرگومان آن نیز نام آرایه‌ای است که جستجو برای یافتن مقدار موردنظر در درون آن انجام می‌شود. تابع مورد بحث درصورتی که مقدار مورد جستجو دردرون آرایه یافت شود مقدار true و در غیر این‌صورت مقدار false را به برنامه فراخواننده باز می‌گرداند. قطعه کد زیر نمونه‌ای از چگونگی بهره‌گیری از این تابع مفید را نشان می‌دهد. در این قطعه کد جستجویی در آرایه $details برای یافتن دنباله کاراکتری "Selective Indifference" ترتیب داده شده است:

```
$details = array ( school ⇒ "Slickly High" ,
                   course ⇒ "Selective Indifference" ) ;
if ( in _ array ( "Selective Indifference" , $details) )
   print "This course has been suspending
      pending investigation < p > \ n" ;
```

چنانکه در این قطعه کد مشاهده می‌کنید، به‌دلیل وجود دنباله کاراکتری مورد جستجو در آرایه $details تابع in _ array ( ) مقدار true را بازمی‌گرداند که به نوبه خود منجر به چاپ پیغامی در خروجی خواهد شد.

از تابع in _ array ( ) معمولاً در مواقعی استفاده می‌کنیم که خواسته باشیم مجدداً از وجود مقدار خاصی در آرایه اطمینان حاصل کنیم. با این همه درصورتی که بخواهیم از موقعیت مقدار موردنظر در آرایه (البته درصورت وجود) اطلاع حاصل کنیم، تابع in _ array ( ) هیچ کمکی به ما نخواهد کرد. درعوض PHP دیگری را تدارک دیده‌است که چنین قابلیتی را در اختیار برنامه‌نویس قرار می‌دهد. این تابع، array _ search ( ) نام دارد. تابع array _ search ( ) نیز ساختاری ساده داشته به‌گونه‌ای که تنها از دو آرگومان ورودی جهت انجام جستجوی مقدار موردنظر در آرایه استفاده می‌کند. آرگومان اول این تابع مقدار موردنظر و آرگومان دوم آن تابع موردجستجو را مشخص می‌کند. درصورتی که تابع array _ search ( ) در کار خود موفق شود یا به‌عبارت دیگر بتواند مقدار مورد جستجو را در آرایه پیدا کند، کلید دستیابی آن مقدار را به برنامه فراخواننده بازمی‌گرداند. اما چنانچه این تابع موفق به یافتن آن مقدار در آرایه موردنظر نشود تنها مقدار false را به برنامه فراخواننده باز خواهدگرداند. قطعه کد زیر در قالب مثالی ساده عملکرد این تابع را نشان داده‌است:

```
$searchme = array (33, 44, 77, "22") ;
print array _ search (22, $searchme) ;
// prints 3
```

همان‌گونه که مشاهده می‌کنید در قطعه کد بالا مقدار ' 22 ' را در آرایه‌ای با نام $searchme مورد جستجو قرار داده‌ایم. نکته‌ای که ذکر آن در این میان جالب توجه است این نکته است که تابع arrray _ search ( ) هنگام تلاش برای یافتن مقدار موردنظر در آرایه از تکنیک casting جهت تبدیل و سپس مقایسه آن مقدار با عناصر آرایه استفاده می‌کند. در قطعه کد مورد بررسی مقدار مورد جستجو

عدد صحیح 22 است حال آنکه چنین عددی در درون آرایه $searchme وجود ندارد اما یک دنباله کاراکتری با عنوان " 22 " در این آرایه موجود است. با این حال مشاهده می‌کنیم که تابع array_search ( ) در فرآیند جستجو موفق بوده و موقعیت این دنباله کاراکتری را به‌عنوان نتیجه عملیات تابع باز می‌گرداند. ممکن است در مواردی این وضعیت خوشایند برنامه‌نویس باشد اما گاهی نیز لازم است تا فرآیند جستجو علاوه بر مقدار، نوع داده آن‌را نیز مورد توجه قرار دهد. تابع array_search ( ) البته چنین قابلیتی را نیز دراختیار ما قرار داده است. سومین آرگومان این تابع یک آرگومان اختیاری است که می‌تواند یکی از دو مقدار true یا false باشد. درصورتی که این آرگومان با مقدار true تغذیه شود تابع array _ search ( ) متوجه این نکته خواهد شد که باید عامل نوع داده را نیز در فرآیند جستجو دخیل نماید. استفاده از مقدار false عملکرد این تابع را تغییر نمی‌دهد. عبارت زیر را که شامل فراخوانی تابع array_search ( ) دررابطه با همان قطعه کد است، درنظر بگیرید. در اینجا همان فرآیند جستجو جهت یافتن مقدار ' 22 ' تکرار شده است، با این تفاوت که این بار عامل نوع داده مقدار مورد جستجو نیز در این فرآیند دخیل است:

```
Print array _ search (22, $searchme, true) ;
```

به‌دلیل اهمیت نوع داده در فرآیند جستجوی مقدار ' 22 '، این بار تابع array_search ( ) موفق به یافتن مقدار موردنظر نشده و بنابراین مقدار false را باز می‌گرداند. تابع فوق آرگومان اختیاری دیگری را نیز به‌عنوان چهارمین آرگومان می‌پذیرد که شرایط سخت‌تری را جهت جستجو تحمیل می‌کند.

> تابع array_search ( ) برای اولین‌بار در PHP4.05 معرفی شده است.

حذف عنصر خاصی از یک آرایه

با بهره‌گیری از تابع unset ( ) می‌توانیم به‌راحتی عنصر دلخواهی از یک آرایه را حذف کنیم. تابع unset ( ) جهت انجام عملیات پیش‌بینی شده تنها به یک آرگومان ورودی نیاز دارد. این تابع نام یک متغیر یا عنصری از یک آرایه را به‌عنوان آرگومان دریافت کرده و آن‌را از آرایه موردنظر حذف می‌کند. به دو نمونه از نحوه استفاده از این تابع جهت حذف عنصر موردنظر از آرایه توجه کنید. این دو عبارت نشان می‌دهند که تابع unset ( ) هم با عناصر آرایه‌های معمولی و هم با عناصر آرایه‌های انجمنی به‌خوبی کار می‌کند:

```
unset ($test [ 'address' ] ) ;
unset ($unmbers [1] ) ;
```

یکی از پیامدهای نامطلوب استفاده از تابع unset ( ) در این حقیقت نهفته است که شاخصهای آرایه پس از فرآیند حذف هیچ تغییری نمی‌کنند. این بدان معنی است که پس از حذف عنصر موردنظر

موقعیت سایر عناصر به‌منظور حفظ ترتیب شاخصها تغییر نخواهد کرد. بدین ترتیب با حذف عنصر $numbers [1] در قطعه کد فوق احتمالاً ترتیب سایر عناصر آرایه $numbers به صورت زیر خواهد بود:

```
$numbers [0]
$numbers [2]
$numbers [3]
```

باوجود این کماکان می‌توانیم با بهره‌گیری از ساختار تکرار foreach عناصر باقی‌مانده در آرایه را مورد پردازش قرار دهیم. جالب است بدانید که با استفاده از تابع ویژه‌ای با نام array _ values ( ) همواره می‌توانیم از روی یک آرایه موجود آرایه جدیدی ایجاد کنیم. بدین ترتیب این تابع ترفندی را به ذهن می‌آورد که با کمک آن می‌توانیم آرایه‌ای را که عنصر یا عناصری را از آن حذف کرده‌ایم، مجدداً بازسازی کنیم به‌گونه‌ای که ترتیب شاخصها اصلاح گردد. درمورد مثال فوق، عبارت زیر می‌تواند این عمل را برای ما انجام دهد:

```
$numbers = array _ values ($numbers) ;
```

اعمال یک تابع به تمامی عناصر موجود در یک آرایه

همان‌گونه که می‌دانید مقدار یک متغیر اسکالر (اعداد صحیح و اعشاری، مقادیر boolean و دنباله‌های کاراکتری) را می‌توانیم به‌راحتی دریک عبارت تغییر دهیم. تغییر یک آرایه از طرف دیگر به‌دلیل تعداد عناصر موجود در آن به‌این راحتی ممکن نیست؛ چراکه تغییر موردنظر باید به تمامی عناصر آن آرایه اعمال گردد (البته در برخی از کاربردها تنها پردازش برخی از عناصر آرایه مدنظر است). برای نمونه اگر قرار باشد تا عدد صحیحی را به تمامی عناصر موجود در یک آرایه اضافه کنیم، همان‌گونه که تابه‌حال دیدیم یکی از روشها این است که با بهره‌گیری از یک حلقه تکرار تمامی عناصر آرایه موردنظر را دستخوش این تغییر نماییم. خوشبختانه PHP در این مورد نیز تسهیلاتی پیش‌بینی کرده است که این فرآیند را به روش دیگری که بسیار ساده‌تر از به‌کارگیری ساختارهای تکرار است، انجام می‌دهد، ضمن اینکه برنامه‌نویس از خطاهای احتمالی ناشی از حلقه‌های تکرار نیز مصون می‌ماند.

پاسخ PHP در این مورد تابعی با نام array _ walk ( ) است. این تابع کلیدهای دستیابی و مقادیر متناظر با آنها در آرایه موردنظر را جهت پردازش به تابعی که برنامه‌نویس آن‌را تعریف می‌کند، ارسال می‌نماید. این تابع از دو آرگومان ضروری استفاده می‌کند. اولین آرگومان نام آرایه موردنظر است که هدف پردازش عناصر آن می‌باشد. دومین آرگومان تابع فوق، نام تابعی است که عناصر آرایه موردنظر راپردازش‌می‌کند.این نام باید در قالب یک دنباله کاراکتری به تابع مورد بحث ارسال شود. آرگومان سوم تابع( ) array _ walk یک آرگومان اختیاری است که خود به عنوان آرگومان تابع تعریف شده توسط کاربر (یعنی تابعی که نام آن به‌عنوان دومین آرگومان تابع ( ) array _ walk مورد استفاده قرار گرفته است) ارسال می‌گردد. درصورتی که تابع پردازش کننده عناصر آرایه به این آرگومان نیاز نداشته باشد، می‌توان از ارسال آن به تابع ( ) array _ walk چشم‌پوشی کرد.

اجازه دهید تا برای روشن شدن مطلب مثالی را با یکدیگر مورد بررسی قرار دهیم. فرض کنید آرایه‌ای متشکل از بهای یکسری کالا در دست داریم که از یک بانک اطلاعاتی استخراج کرده‌ایم. اما پیش از بهره‌برداری از این آرایه ابتدا لازم است تا مقداری را به‌عنوان مالیات به هریک از عناصر این آرایه اضافه کنیم. در قدم اول می‌توانیم تابعی ایجاد کنیم که وظیفه آن دریافت کلید دستیابی و مقدار متناظر با آن از یک آرایه و افزودن مالیات موردنظر به مقدار آن است. تابع ساده add _ tax ( ) که تعریف آن‌را در زیر مشاهده می‌کنید، به‌راحتی چنین فرآیندی را انجام می‌دهد:

```
function add _ tax (&$val, $key, $tax _ pc) {
   $val + = (( $tax _ pc / 100) * $val) ;
}
```

نکته مهمی که در این رابطه هنگام طراحی تابع مورد نظرتان باید همواره به‌خاطر داشته باشید این است که هر تابعی که نام آن به‌عنوان دومین آرگومان تابع array _ walk ( ) مورد استفاده قرار می‌گیرد‌بایددست کم دو آرگومان ضروری دریافت کند. اولین آرگومانها نماینده کلید دستیابی و دومین آنها مقدار متناظر با آن کلید دستیابی از آرایه موردنظر است. ضمناً تابع مذکور می‌تواند آرگومانهای دیگری را نیز به‌عنوان آرگومانهای اختیاری دریافت نماید.

همان‌گونه که پیشتر در درس مربوط به توابع نیز بارها عنوان کردیم، اگر مایل باشیم تا مقدار متغیری را که به‌عنوان آرگومان به یک تابع ارسال می‌شود در درون تابع تغییر دهیم، باید پیش از نام آن آرگومان در تعریف تابع از علامت & استفاده کنیم. این عمل موجب می‌شود تا تابع موردنظر به خود آن متغیر دسترسی داشته باشد (به این روش ارسال آرگومان به تابع ارسال از طریق مرجع گفته می‌شود. روش دیگر ارسال متغیر به تابع ارسال کپی آن است که طبعاً تابع موردنظر به اصل متغیر دسترسی نخواهد داشت). به همین دلیل هرگونه تغییری که در درون تابع بر روی متغیر ارسال شده به آن اعمال شود، موجب تغییر مقدار اصلی آن متغیر خواهد شد. چنانکه در تعریف تابع add _ tax ( ) مشاهده می‌کنید اولین آرگومان که بیانگر مقدار عنصری از آرایه می‌باشد از طریق ارسال به روش مرجع به این تابع ارسال شده است. از این‌رو تابع مذکور به‌سادگی می‌تواند مقدار اصلی این متغیر را تغییر دهد. به‌همین دلیل نیز تابع add _ tax ( ) هیچ مقداری را به برنامه فراخواننده باز نمی‌گرداند.

اکنون که تابع مورد نظرمان را جهت تغییر عناصر آرایه تعریف کردیم، آماده‌ایم تا از طریق تابع array _ walk ( ) آن‌را فراخوانی کنیم. قطعه برنامه موجود در لیست ۱- ۱۶ تعریف تابع add _ tax ( ) و چگونگی استفاده از آن در تابع array _ walk ( ) و همچنین خروجی حاصل از اجرای تابع اخیر را نشان می‌دهد.

```
function add_tax( &$val, $key, $tax_pc ) {
    $val += ( ($tax_pc/100) * $val );
}
$prices = array( 10, 17.25, 14.30 );
array_walk( $prices, "add_tax", 10 );
foreach ( $prices as $val )
    print "$val<BR>";
// Output:
// 11
// 18.975
// 15.73
```

لیست ۱- ۱۶ تعریف تابع ( ) add _ tax و بهره‌گیری از تابع ( ) array _ walk به‌همراه خروجی مربوطه

همان‌گونه که در این لیست مشاهده می‌کنید آرایه $prices به همراه تابع ( ) add _ tax به عنوان دو آرگومان ضروری به تابع ( ) array _ walk ارسال شده‌اند. تابع ( ) add _ tax همان‌گونه که از تعریف آن بر می‌آید جهت انجام عملیات پیش‌بینی شده نیازمند اطلاع از نرخ مالیات فروش جاری است. سومین آرگومان تابع ( ) array _ walk که یک آرگومان اختیاری است به همین منظور پیش‌بینی شده است. بنابراین جهت ارسال نرخ مالیات مذکور به تابع ( ) add _ tax ، کافی است تا آن‌را به‌عنوان سومین آرگومان به تابع ( ) array _ walk ارسال کنیم. از تکنیک فوق علاوه بر این می‌توانید در فراخوانی متدی از یک شئ به‌ازای هریک از عناصر موجود در یک آرایه نیز استفاده کنید. در این کاربرد به‌جای ارسال نام تابع به تابع ( ) array _ walk (آرگومان دوم)،لازم است تا یک آرایه دو عضوی را که عضو اول آن نام شئ موردنظر است به تابع نامبرده ارسال کنید. عنصر دوم این آرایه یک دنباله کاراکتری است که نام متد موردنظر از شئ مذکور را جهت فراخوانی مشخص می‌کند. قطعه برنامه موجود در لیست ۲- ۱۶ شامل یک کلاس با نام TaxAdder است که نمونه‌ای از آن با نام $taxer در قالب اولین عنصر از آرایه موردنیاز تابع ( ) array _ walk به این تابع ارسال شده است.

```
class TaxAdder {
   var $taxrate = 10;
   function add_tax( &$val, $key ) {
       $val += ( ($this->taxrate/100) * $val );
   }
}

$prices = array( 10, 17.25, 14.30 );
$taxer = new TaxAdder();

array_walk( $prices, array( $taxer, "add_tax") );
foreach ( $prices as $val )
    print "$val<BR>";
// Output:
// 11
// 18.975
// 15.73
```

لیست ۲- ۱۶ بهره‌گیری از تابع ( ) array _ walk جهت فراخوانی متدی از یک شیئ به‌ازای عناصر یک آرایه

با انتشار PHP 4.0.6 ، ابزار جدیدی نیز جهت اعمال توابع تعریف شده توسط کاربر به عناصر یک آرایه معرفی شد. این ابزارِ جدید چیزی نیست جز تابعی با نام ( ) array _ map که جهت اجرا به نام یک تابع و دست کم نام یک آرایه نیاز دارد. پس از ارسال آرایه موردنظر به تابع ( ) array _ map ، این تابع آرایه جدیدی را به‌عنوان حاصل عملیات خود به برنامه فراخواننده باز می‌گرداند. این آرایه جدید شامل مقادیر بازگشتی حاصل از عملیات تابعی است که آرایه اول را مورد پردازش قرار می‌دهد. فرآیند مذکور به‌طور خلاصه، نگاشتی یک به یک از عناصر آرایه ورودی به آرایه بازگشتی می‌باشد. درصورتی که تعداد آرایه‌های ارسالی به تابع ( ) array _ map بیش از یکی باشد تابع مذکور بلافاصله پس از دریافت آرایه تمامی عناصر آنها را به‌طور موازی تحت پردازش تابع تعیین شده به‌عنوان اولین آرگومان قرار می‌دهد، بدین ترتیب که ابتدا از تمامی آرایه‌های دریافتی را به تابع مذکور ارسال می‌کند. پس از اتمام پردازش این عناصر شاخص شماره یک از تمامی عناصر آرایه‌های دریافتی شاخص شماره صفر را به تابع مذکور ارسال می‌کند و فرآیند تا به آخر به همین صورت انجام می‌شود. چنانکه مطلب اندکی گنگ است، با کمک یک مثال می‌توانیم نکات مبهم در این فرآیند را روشن کنیم.

فرض کنید لیستی شامل اسامی چند صفحه وب را در اختیار داریم و اکنون مایلیم تا این اسامی را که در قالب دنباله‌های کاراکتری در اختیار ما گذاشته شده‌اند به پیوندهای فعالی بر روی صفحه مرورگر اینترنت تبدیل کنیم که کاربر با کلیک بر روی آنها بتواند صفحه مقصد را بر روی صفحه مرورگر خود مشاهده نماید. اکنون بهترین زمان جهت بهره‌گیری از توانمندی‌های تابع( ) array_map است. لیست ۳- ۱۶ قطعه برنامه‌ای را جهت حل این مسأله نشان می‌دهد.

```
$urls = array( "about.html", "index.html", "contact.html", "service.html" );
function make_link( $a ) {
   return "<a href=\"$a\">a link</a><br>\n";
}
$new_urls = array_map( "make_link", $urls );
foreach ( $new_urls as $val )
   print $val;

// prints:
// <a href="about.html">a link</a><br>
// <a href="index.html">a link</a><br>
// <a href="contact.html">a link</a><br>
// <a href="service.html">a link</a><br>
```

لیست ۳- ۱۶ استفاده از تابع ( ) array _ map

همان‌گونه که در این لیست مشاهده می‌کنید، تابع ( ) array _ map به‌طور مکرر تابع ( ) make_link را فراخوانی کرده و عناصر آرایه $urls را به‌عنوان آرگومان ورودی به آن ارسال می‌کند. تابع ( ) make_link با دریافت هریک از عناصر آرایه مذکور آنها را به پیوندی فعال تبدیل می‌کند. آنچه که تابع ( ) array _ map در این میان باز می‌گرداند آرایه جدیدی است که شامل پیوندهای فعال می‌باشد. بین عناصر دو آرایه ورودی به تابع فوق یعنی $urls و آرایه بازگشتی از این تابع یک تناظر یک به یک برقرار است.

اما همان‌گونه که پیشتر نیز اشاره کردیم، قابلیت اصلی تابع ( ) array _ map در توانایی آن در کار بر روی چندین آرایه مختلف به‌طور موازی می‌باشد. تصور کنید که علاوه بر آرایه $urls ، آرایه دیگری داریم که شامل عناوین صفحاتی است که کاربر با کلیک بر روی عناصر آرایه اول ($urls) آنها را مشاهده خواهد کرد. آنچه در اینجا مایلیم انجام دهیم این است که اسامی هریک از این صفحات را به‌عنوان یک پیوند فعال بر روی یک صفحه وب نمایش دهیم. به‌عبارت دیگر قصد داریم تا رابطه‌ای مابین عناصر این دو آرایه ایجاد کنیم. لیست ۴- ۱۶ با بهره‌گیری از تابع ( ) array _ map عملیات موردنظر را پیاده‌سازی کرده است.

```
$urls = array( "about.html", "index.html", "contact.html",
"services.html" );
$names = array( "about us", "home", "contact us", "our
services" );

function make_link( $a, $b ) {
      return "<a href = \"$a\">$b</a><br>\n";
}

$new_urls = array_map( "make_link", $urls, $names );
foreach( $new_urls as $val )
      print $val;
```

```
// prints:
// <a href = "about.html">about us</a><br>
// <a href = "index.html">home</a><br>
// <a href = "contact.html">contact us</a><br>
// <a href = "services.html">our services</a><br>
```

لیست ۴- ۱۶ استفاده از تابع ( ) array _ map جهت پردازش همزمان دو آرایه مختلف

چنانکه در این لیست مشاهده می‌کنید، این‌بار علاوه بر آرایه شامل اسامی پیوندها ($urls) آرایه دیگری را نیز با نام $names که شامل عناوین صفحات متناظر با عناصر آرایه اول است، به تابع array _ ( ) map ارسال کرده‌ایم. همچنین تغییراتی در ساختار تابع ( ) make _ link داده‌ایم تا بتواند دو آرگومان مختلف را از ورودی دریافت کند. تابع ( ) make _ link ، عناصر هر دو آرایه $urls و $names را به‌عنوان دو آرگومان دریافت کرده و پس از انجام عملیات موردنظر بر روی آنها مقداری را که بیانگر یک پیوند فعال HTML است در قالب دنباله‌ای از کاراکترها به برنامه فراخواننده بازمی‌گرداند. تابع ( ) map _ array، با دریافت مقادیر بازگشتی از تابع ( ) make _ link ، اقدام به ایجاد یک آرایه جدید کرده و آن را در متغیری با نام $new _ urls ذخیره می‌کند.

پالایش آرایه‌ها با استفاده از تابع ( ) array _ filter

آرایه‌ها به‌طور ذاتی ساختارهای بسیار منعطفی هستند و این موضوع نقطه قوت بزرگی برای آنها محسوب می‌شود. علاوه بر این محدودیتهای بسیار اندکی در رابطه با نوع داده‌ها و همچنین محدوده‌ای از داده‌ها که می‌توانند ذخیره کنند، وجود دارد. با این همه در برخی موارد لازم است تا دقیقاً عناصری از آرایه را که در محدوده خاصی قرار داشته یا به‌عبارت دیگر در شرایط خاصی صدق می‌کنند، بازیابی کنیم. برای مثال ممکن است تنها به عناصری از یک آرایه که ماهیت آنها دنباله کاراکتری است، علاقه‌مند بوده یا بخواهیم اعداد صحیحی را که از یک حدی بزرگ‌تر باشند، مورد بازیابی قرار دهیم. در زبان PHP تابعی پیش‌بینی شده است که انجام این‌گونه پالایش‌ها را در مورد عناصر یک آرایه ممکن می‌سازد. نام این تابع ( ) array _ filter است. تابع ( ) array _ filter جهت اجرای عملیات موردنظر به دو آرگومان نیاز دارد. آرگومان اول این تابع نام یک آرایه و آرگومان دوم آن مرجعی به یک تابع است که توسط تابع ( ) array _ filter فراخوانی می‌شود (این مرجع ممکن است نام یک تابع دیگر یا یک تابع بدون نام را مشخص نماید). مقدار بازگشتی از تابع ( ) array _ filter آرایه‌ای است که حاصل پالایش آرایه ورودی به تابع فوق می‌باشد. تابع ( ) array _ filter به‌ازای هریک از عناصر آرایه دریافتی تابعی را که نام آن به‌عنوان دومین آرگومان دریافت شده است، فراخوانی می‌کند. نتیجه بازگشتی از تابع فراخوانی شده توسط ( ) array _ filter یک مقدار boolean است. درصورتی که این تابع مقدار true را به تابع ( ) array _ filter بازگرداند عنصر تحت بررسی از آرایه ورودی در آرایه

بازگشتی از تابع ( ) array _ filter اسامی تمامی کلیدی دستیابی را در یک آرایه انجمنی حفظ می‌کند. برای روشن شدن مطلب نمونه‌ای از استفاده از این تابع را در قطعه برنامه موجود در لیست ۵- ۱۶ آورده‌ایم. در این قطعه برنامه از تابع ( ) array _ filter جهت ایجاد آرایه‌ای که مقادیر آن کوچک‌تر از عدد صحیح 120 است، استفاده شده است.

```
function less_than( $a ) {
      return ( $a < 120 );
}

$my_array = array( "a" => 200, "b" => 80, "c" => 90, "d" =>
150, "e" => 130, "f" => 110 );
$filtered_array = array_filter( $my_array, less_than );
foreach( $filtered_array as $key => $val )
      print "$key: $val<br>";

// prints:
// b: 10
// c: 90
// f: 110
```

لیست ۵- ۱۶ استفاده از تابع ( ) array _ filter جهت پالایش یک آرایه

در قطعه برنامه فوق به شیوه‌ای که تابع ( ) less _ than مقدار بازگشتی از نوع boolean را ایجاد می‌کند، توجه کنید. این تابع همان‌گونه که مشاهده می‌کنید نتیجه حاصل از یک عبارت منطقی را به برنامه‌فراخواننده(در اینجا تابع ( ) array _ filter) باز می‌گرداند. این روش معمولاً در میان برنامه‌نویسان مرسوم است، اما درصورتی‌که قابلیت خوانایی بیشتری را از برنامه انتظار دارید، می‌توانید به‌راحتی عبارت فوق را به دو عبارت مجزا به‌صورت زیر تقسیم کنید:

```
if ($a < 120)
   return true ;
return false ;
```

علاوه براین توجه کنید که کلیدهای دستیابی آرایه حاصل از فراخوانی تابع ( ) array _ filter به‌همراه مقادیر متناظر با آنها در آرایه ( ) array _ filtered$ ذخیره می‌شود.

مرتب سازی عناصر آرایه به شیوه دلخواه

در درسهای گذشته با چگونگی مرتب‌سازی عناصر آرایه برمبنای کلید دستیابی یا مقادیر موجود در آن آشنا شدید. اما مرتب‌سازی آرایه به روش فوق همواره آن‌چیزی نیست که موردنظر ما باشد. گاهی ممکن است لازم باشد تا این عناصر را بر مبنای مقدار خاصی از آرایه مرتب کنیم یا

به‌عنوان یک نمونه دیگر بخواهیم عناصر آرایه را به شیوه‌ای مرتب کنیم که به کلی با ترتیب استاندارد حروف الفبا یا اعداد ریاضی متفاوت باشد.

در زبان برنامه‌نویسی PHP این امکان وجود دارد که برنامه‌نویس برطبق شیوه‌ای ازمقایسه مقادیر که مدنظر دارد توابعی را ایجاد کرده و عناصر آرایه را با بهره‌گیری از این توابع که درواقع حکم عملگرهای مقایسه‌ای جدید را دارند، مرتب نماید. در این شیوه از مرتب‌سازی درمورد آرایه‌هایی که با استفاده از اعداد صحیح شاخص‌گذاری شده‌اند از تابعی با عنوان ( ) usort استفاده می‌کنیم. این تابع جهت عملیات موردنظر خود به دو آرگومان نیاز دارد. آرگومان اول آرایه‌ای است که قصد مرتب‌سازی عناصرآن‌را داریم. آرگومان دوم این تابع خود نام تابعی است که مقایسات موردنیاز را به صورتی که برنامه‌نویس تعیین کرده است، انجام می‌دهد.

تابعی که برنامه‌نویس جهت مقایسه عناصر آرایه تعریف می‌کند باید شامل دو آرگومان ورودی باشد. این دو آرگومان طبق منطقی که در درون تابع پیاده‌سازی می‌شود، مورد مقایسه قرار می‌گیرند. درصورتی که این دو آرگومان با یکدیگر برابر باشند تابع باید مقدار صفر را به برنامه فراخواننده بازگرداند (منظور از برابری در اینجا معادل بودن دو آرگومان برطبق معیارهای تعریف شده در درون تابع است). درصورتی‌که آرگومان اول باید به آرگومان دوم تقدم داشته باشد، تابع باید مقدار عددی 1 - و در صورتی که آرگومان دوم باید نسبت به آرگومان اول تقدم داشته باشد لازم است تا تابع تعریف شده توسط برنامه‌نویس مقدار عددی 1 را به برنامه فراخواننده بازگرداند.

برای روشن شدن مطلب فوق و نحوه اطلاع دقیق از چگونگی کارکرد تابع ( ) usort و همچنین نحوه تعریف تابع مقایسه‌کننده اجازه دهید تا در اینجا برنامه‌ای را مورد بررسی قرار دهیم. برنامه موجود در لیست ۶- ۱۶ با بهره‌گیری از تابع ( ) usert و تعریف تابعی با نام priceCmp عناصر یک آرایه چندبعدی را مطابق منطق تعریف شده در تابع اخیر مرتب می‌کند.

```
<?php
$products = array(
        array( name=>"HAL 2000",           price=>4500.5  ),
        array( name=>"Tricorder",          price=>55.5    ),
        array( name=>"ORAC AI",            price=>2200.5  ),
        array( name=>"Sonic Screwdriver", price=>22.5     )
        );
function priceCmp( $a, $b ) {
    if ( $a['price'] == $b['price'] )
        return 0;
    if ( $a['price'] < $b['price'] )
        return -1;
    return 1;
}
usort( $products, priceCmp );
foreach ( $products as $val )
```

```
    print $val['name'].": ".$val['price']."<br>\n";
?>
```

لیست ۶- ۱۶ استفاده از تابع ( ) usort جهت مرتب‌سازی عناصر یک آرایه چندبعدی

همان‌گونه که در این لیست مشاهده می‌کنید ابتدا آرایه‌ای را با عنوان $products درخط ۲ از برنامه تعریف کرده‌ایم. آنچه که برنامه درنهایت انجام می‌دهد مرتب‌سازی عناصر این آرایه چندبعدی بر مبنای فیلد price هریک از آرایه‌های یک بعدی موجود در درون این آرایه است.

تابعی که دربرگیرنده منطق مقایسه است در خط ۸ از برنامه با عنوان ( ) priceCmp تعریف شده است.

این تابع چنانکه ملاحظه می‌کنید دو آرگومان ورودی با نامهای $a و $b را از برنامه فراخواننده این تابع دریافت می‌کند. این دو آرگومان در درون تابع ( ) priceCmp مشخص‌کننده دو آرایه‌ای هستند که فیلدهای price از هریک در درون تابع مذکور با یکدیگر مقایسه خواهند شد. با مقایسه مقادیر این فیلدها در درون تابع( ) priceCmp درصورتی‌که این مقادیر از نظر ریاضی برابر باشند تابع مذکور با استفاده از دستورالعمل return در خط ۱۰ مقدار صفر را به برنامه فراخواننده (که در اینجا تابع ( ) usort است) باز می‌گرداند. چنانچه فیلد اول کوچک‌تر از فیلد دوم باشد، دستورالعمل return خط ۱۲ مقدار 1 - را به برنامه فراخواننده باز می‌گرداند. و بالاخره درصورتی که فیلد اول کوچک‌تر از فیلد دوم باشد دستورالعمل return خط ۱۳ مقدار 1 را به برنامه فراخواننده باز خواهد گرداند (دقت کنید که برای مورد آخر هیچ‌گونه مقایسه‌ای صورت نگرفته و تنها به صورت ضمنی نتیجه‌گیری شده است).

با در اختیار داشتن تعریف تابع ( ) priceCmp (تابع مقایسه کننده) و آرایه چندبعدی $products ، اکنون می‌توانیم تابع ( ) usort را جهت مرتب‌سازی عناصر آرایه در خط ۱۵ از برنامه فراخوانی کنیم. چنانکه می‌بینید آرایه ( ) $products و نام تابع ( ) priceCmp به ترتیب به عنوان آرگومانهای اول و دوم تابع ( ) usort مورد استفاده این تابع قرار گرفته‌اند. تابع مذکور پس از دریافت آرگومانهای نامبرده به طور مکرر تابع ( ) priceCmp را فراخوانی کرده و عناصر آرایه $products را به آن ارسال می‌کند. در نهایت تابع ( ) usort ترتیب عناصر فوق را با توجه به مقدار بازگشتی از تابع ( ) priceCmp درهربار فراخوانی مشخص می‌کند.

برنامه فوق در خط ۱۶ آرایه مرتب شده را با بهره‌گیری از یک ساختار تکرار نمایش می‌دهد.

توجه کنید که در مورد آرایه‌های شاخص‌گذاری شده با اعداد صحیح (که ما آنها را آرایه‌های معمولی نیز می‌نامیم) از تابع ( ) usort استفاده می‌کنیم. اگر لازم باشد تا آرایه‌های انجمنی را نیز به این شیوه مرتب کنیم باید تابع دیگری را که عملکرد مشابهی را در اختیارمان قرار می‌دهد، استفاده نماییم. نام این تابع ( ) uasort است. تابع ( ) uasort ضمن مرتب‌سازی آرایه‌های انجمنی، جامعیت عناصر این آرایه‌ها را نیز حفظ می‌کند. یعنی رابطه بین کلید دستیابی و مقدار متناظر با آن درآرایه

پس از فرآیند مرتب‌سازی کماکان به قوت خود باقی می‌ماند. برنامه موجود در لیست ۷- ۱۶ نمونه‌ای از چگونگی به‌کارگیری تابع ( ) uasort را جهت مرتب‌سازی عناصر آرایه‌ای با نام $products که در این لیست یک آرایه انجمنی است، نشان می‌دهد. همان‌گونه که مشاهده می‌کنید این تابع در خط ۱۵ از این لیست فراخوانی شده است.

```
<?php
$products = array(
        "HAL 2000" => array( color =>"red",    price=>4500.5
),
        "Tricorder" => array( color =>"blue",  price=>55.5
),
        "ORAC AI" => array( color =>"green", price=>2200.5
),
        "Sonic Screwdriver" => array( color =>"red",
price=>22.5     )
        );
function priceCmp( $a, $b ) {
    if ( $a['price'] == $b['price'] )
        return 0;
    if ( $a['price'] < $b['price'] )
        return -1;
    return 1;
}
uasort( $products, priceCmp );
foreach ( $products as $key => $val )
    print "$key: ".$val['price']."<br>\n";
?>
```

لیست ۷- ۱۶ استفاده از تابع ( ) uasort جهت مرتب‌سازی عناصر یک آرایه انجمنی

ازآنجا که عملکرد تابع ( ) uasort کاملاً مشابه تابع ( ) usort است از توضیح بیشتر درباره برنامه موجود در لیست فوق خودداری می‌کنیم.

درصورت تمایل می‌توانیم عناصر موجود در آرایه‌های انجمنی را براساس کلیدهای دستیابی طبق یک منطق مشخص مرتب نماییم. تابعی که در PHP چنین قابلیتی را در اختیارمان قرار می‌دهد، ( ) uksort نام دارد. این تابع دقیقاً عملکردی مشابه دو تابع دیگر یعنی ( ) usort و ( ) uasort دارد، با این تفاوت ظریف و قابل توجه که تابع موردبحث فرآیند مقایسه را به‌جای اینکه درمورد مقادیر آرایه انجام دهد این کار را درمورد کلیدهای دستیابی یک آرایه انجمنی انجام می‌دهد. برای روشن شدن مطلب اجازه دهید تا برنامه‌ای را که در همین زمینه تهیه شده است، با هم بررسی کنیم. کد مربوط به این برنامه در لیست ۸- ۱۶ قابل مشاهده است.

```
<?php
$exes = array(
        'xxxx' => 4,
```

```
        'xxx' => 5,
        'xx' => 7,
        'xxxxx' => 2,
        'x' => 8
        );
function priceCmp( $a, $b ) {
    if ( strlen( $a ) == strlen( $b ) )
        return 0;
    if ( strlen( $a ) < strlen( $b ) )
        return -1;
    return 1;
}
uksort( $exes, priceCmp );
foreach ( $exes as $key => $val )
    print "$key: $val<br>\n";

// output:
// x: 8
// xx: 7
// xxx: 5
// xxxx: 4
// xxxxx: 2
?>
```

لیست ۸- ۱۶ بهره‌گیری از تابع( ) uksort جهت مرتب‌سازی عناصر یک آرایه انجمنی بر مبنای کلیدهای دستیابی

همان‌گونه که در این لیست مشاهده می‌کنید، در خط ۱۶ از برنامه تابع ( ) uksort به‌منظور مرتب‌سازی یک آرایه انجمنی با نام $exes مورد استفاده قرار گرفته است. تابع مقایسه کننده که باز هم ( ) priceCmp نام گرفته است مقایسه را بر مبنای تعداد کاراکترهای موجود در کلیدهای دستیابی این آرایه انجمنی انجام می‌دهد. این تابع برای تشخیص طول هر کاراکتری (کلید دستیابی) نیازمند بهره‌گیری از تابع ( ) strlen است. در درس ساعت آینده این تابع را به‌طور کامل مورد بررسی قرار خواهیم داد. چنانکه می‌بینید از این تابع در خطوط ۱۰ و ۱۲ از برنامه جهت تعیین تعداد کاراکترهای هریک از کلیدهای دستیابی آرایه انجمنی $exes استفاده کرده‌ایم. تابع ( ) strlen ساختار بسیار ساده‌ای دارد به‌گونه‌ای که یک دنباله کاراکتری ساده را مستقیماً یا در قالب یک متغیر برنامه دریافت کرده و عدد صحیحی را که نماینده تعداد کاراکترهای آن است به‌عنوان نتیجه عملیات به برنامه فراخواننده (که در اینجا تابع ( ) priceCmp است) بازمی‌گرداند.

جمع‌بندی

درس این ساعت همان‌گونه که تصدیق می‌کنید به بررسی برخی از مباحث پیشرفته زبان برنامه‌نویسی PHP در رابطه با بهره‌گیری از آرایه‌ها و انواع داده‌ها اختصاص داشت. در درس این ساعت شما با فرآیندی که هنگام تبدیل انواع داده‌های پیچیده به مقادیر اسکالر (اعداد صحیح و اعشاری، نوع داده boolean و دنباله‌های کاراکتری) رخ می‌دهد، همچنین فرآیند معکوس آشنا شدید. مشاهده کردید که چگونه PHP از انواع مختلف داده‌ها در یک عبارت استفاده کرده و درصورت لزوم با بهره‌گرفتن از تکنیک مفید casting اقدام به تبدیل آنها به یکدیگر می‌کند. همچنین در این درس توابعی چون ( ) is_array را که جهت ارزیابی یک نوع داده خاص به‌کار می‌روند و نیز توابع دیگری مانند تابع ( )inval را که به‌منظور تبدیل انواع داده‌های مختلف به یکدیگر مورد استفاده قرار می‌گیرند، مورد بحث و بررسی دقیق قرار دادیم. مطلب دیگری که در درس این ساعت مورد بررسی قرار دادیم، روشی بود که در نسخه‌های پیشین PHP جهت پردازش عناصر آرایه مورد استفاده قرار می‌گرفت. در این روش از دو تابع ( ) each و ( ) list استفاده می‌شد. مطلب دیگر بررسی وجود یک مقدار خاص در یک آرایه بود که با بهره‌گیری از تابعی با نام ( ) in _ array انجام می‌شد. موضوع مهم دیگر چگونگی حذف یک مقدار از یک آرایه بود. همان‌گونه که ملاحظه کردید برای انجام این کار PHP تابعی با نام ( ) usert را پیش‌بینی کرده است. پالایش عناصر یک آرایه موضوع مهم دیگری بود که در درس این ساعت مورد بررسی قرار گرفت. سه تابع ( ) array _ walk، ( ) array _ map و ( ) array _ filter توابعی هستند که PHP برای انجام این کار در اختیار برنامه‌نویس قرار داده است. آخرین مطلبی که در این درس مورد بررسی قرار گرفت چگونگی مرتب‌سازی عناصر یک آرایه بر مبنای قوانین و منطق مرتب‌سازی تدوین شده و توسط برنامه‌نویس در قالب یک تابع بود که ما آن‌را تابع مرتب‌سازی نامیدیم. در این رابطه نیز PHP توابع مفیدی با عنوان ( ) usort، ( ) uasort و ( ) uksort را جهت استفاده برنامه‌نویس تدارک دیده است. دو تابع آخر برای مرتب‌سازی عناصر آرایه‌های انجمنی مورد استفاده قرار می‌گیرند.

در درس ساعت آینده به چگونگی کار با دنباله‌های کاراکتری اختصاص دارد. ابزارهای معرفی شده توسط PHP جهت بررسی، قالب‌بندی و اعمال تغییرات بر روی دنباله کاراکتری موضوع بحث ساعت آینده است.

پرسش و پاسخ

پرسش: آیا در این درس و درس مربوط به آرایه‌ها کلیه توابعی که PHP4 جهت کار با آرایه‌ها معرفی کرده است مورد بحث و بررسی قرار گرفتند؟

پاسخ: خیر، توابعی که در PHP4 به‌منظور کار با آرایه‌ها در اختیار برنامه‌نویس قرار داده شده است، بسیار متنوع است؛ اما به‌دلیل محدودیت کتاب تنها تعدادی از آنها را که متداول‌تراز بقیه تشخیص دادیم، بررسی نمودیم. جهت اطلاع از لیست کامل این توابع و جزئیات مربوطه به آدرس URL زیر مراجعه کنید:

http: // www. php. net / manual / ref. array. Php

تمرینها

هدف از این بخش ارائه تمرینهایی در قالب آزمون است. پاسخ بخش آزمون بلافاصله بعد از آن آمده است. بخش فعالیتها شامل تمرینهایی است که به منظور افزایش مهارت و قابلیتهای برنامه‌نویسی خواننده طراحی شده است که البته فاقد پاسخ است.

آزمون

۱- از کدام تابع PHP می‌توان جهت تبدیل انواع داده‌های موجود در این زبان برنامه‌نویسی به یکدیگر استفاده کرد؟

۲- آیا روشی هست که طی آن بدون بهره‌گیری تابع فرآیند خواسته شده در تمرین قبل را بتوان انجام داد؟

۳- عبارت زیر چه‌چیزی را بر روی صفحه نمایش می‌دهد؟

```
print "four" * 200 ;
```

۴- چگونه می‌توان درمورد اینکه متغیری شامل یک آرایه می‌باشد یا خیر اطلاع حاصل کرد؟

۵- از کدام تابع PHP می‌توان جهت دسترسی به معادل عدد صحیح آرگومان ورودی تابع استفاده کرد؟

۶- چگونه می‌توان در مورد اینکه آیا متغیری مقداردهی شده است یا خیر اطلاع حاصل نمود؟

۷- چگونه می‌توان درمورد اینکه آیا متغیری فاقد مقدار تهی است یا خیر اطلاع حاصل کرد؟ (منظور از مقدار تهی عدد صفر یا دنباله کاراکتری خالی است)

۸- از کدام تابع PHP می‌توان جهت حذف عنصری از یک آرایه استفاده نمود؟

۹- از کدام تابع PHP می‌توان جهت مرتب‌سازی عناصر آرایه‌ای که با استفاده از اعداد صحیح شاخص‌گذاری شده‌اند بر مبنای یک ترتیب دلخواه و قابل تعریف توسط برنامه‌نویس استفاده کرد؟

پاسخ آزمون

۱- با بهره‌گیری از تابع ( ) settype در زبان PHP می‌توان مقادیری از هرنوع را به نوع داده موردنظر تبدیل کرد.

۲- با بهره‌گیری از روشی که به تکنیک casting معروف است بدون استفاده از توابع می‌توان فرآیند تبدیل نوع داده‌ها را به یکدیگر انجام داد. برای این‌کار کافی است تا نام نوع داده موردنظر را در درون یک جفت پرانتز پیش از مقداری که قصد تبدیل آن‌را داریم یا پیش از نام متغیری که شامل این مقدار است، قرار دهیم.

۳- از آنجا که دنباله‌های کاراکتری که محتوای آنها با یک رقم شروع نمی‌شود در فرآیند تبدیل معادل عدد صفر فرض می‌شوند، لذا عبارت موردنظر معادل صفر می‌باشد.

۴- جهت اطلاع درمورد اینکه آیا متغیری از برنامه شامل یک آرایه می‌باشد یا خیر می‌توانیم از تابعی با نام ( ) is _ array که به همین منظور در زبان برنامه‌نویسی PHP پیش‌بینی شده است، استفاده نماییم. روش دیگر این است که متغیر موردنظر را با استفاده از تابع ( ) gettype (مورد ارزیابی قرار دهیم. از مقدار بازگشتی حاصل از اجرای این تابع می‌توانیم به نتیجه موردنظر دست پیدا کنیم. تنها کافی است تا نام متغیر موردنظر را به‌عنوان آرگومان به این تابع ارسال نماییم.

۵- بابهره‌گیری از تابع ( ) intval می‌توانیم معادل عدد صحیح یک مقدار خاص را به دست آوریم. کافی است تا مقدار موردنظر را به عنوان آرگومان به این تابع ارسال کنیم. مقدار بازگشتی از این تابع معادل عددصحیح آن مقدار خواهد بود.

۶- با استفاده از تابع ( ) isset در زبان برنامه‌نویسی PHP می‌توانیم از این موضوع که آیا متغیری از برنامه حاوی مقدار می‌باشد یا خیر اطلاع حاصل کنیم.

۷- با بهره‌گیری از تابع ( ) empty در PHP می‌توان از خالی بودن یک متغیر در برنامه اطلاع حاصل کرد. چنانچه این تابع که متغیر مورد بررسی را به‌عنوان آرگومان ورودی دریافت می‌کند مقدار true را به برنامه فراخواننده باز گرداند می‌توان چنین نتیجه گرفت که آن متغیر مقداردهی نشده است یا اینکه مقدار آن صفر یا یک دنباله کاراکتری تهی است.

۸- به کمک تابع ( ) unset در زبان PHP می‌توان عنصری از یک آرایه را حذف نمود. برای بهره‌گیری از این تابع کافی است تا عنصر موردنظر را به‌عنوان آرگومان به این تابع ارسال کنیم.

۹- بابهره‌گیری‌ازتابع ( ) usort در زبان برنامه‌نویسی PHP می‌توان آرایه‌ای را که دارای شاخصهای عددی می‌باشد بر مبنای یک منطق دلخواه مرتب نمود.

فعالیتها

۱- باردیگر برنامه‌های ارائه شده در کلیه درسهای این کتاب را که از ساختار تکرار foreach استفاده کرده‌اند، مورد بررسی قرار داده و آنها را به‌گونه‌ای بازنویسی کنید که با PHP3 سازگار باشند. توجه کنید که ساختار تکرار foreach برای اولین بار در PHP4 معرفی شده است.

۲- آرایه دلخواهی را که شامل انواع مختلفی از داده‌ها باشد، ایجاد کنید و سپس عناصر موجود در آن‌را بر مبنای یک الگوی دلخواه مرتب نمایید.

ساعت هفدهم

بهره‌گیری از دنباله‌های کاراکتری

وب جهان‌گستر (World Wide Web) به شدت به متون ساده وابسته است. مهم نیست که تا چه اندازه از محتوای گرافیکی یا چندرسانه‌ای استفاده می‌کنید، آنچه که زیربنای هر محتوایی را در مورد اسناد وب تشکیل می‌دهد کد HTML است. از این‌رو تعجبی ندارد اگر بدانیم توابع بسیار متنوعی جهت قالب‌بندی، بررسی و ارزیابی و تغییر و دستکاری دنباله‌های کاراکتری در زبان برنامه‌نویسی PHP4 پیش‌بینی شده است.

رئوس مطالب مورد بحث در این ساعت به قرار زیر است:

- **قالب‌بندی دنباله‌های کاراکتری**
- **نحوه اطلاع از طول (تعداد کاراکترهای) یک دنباله کاراکتری**
- **چگونگی پیداکردن یک دنباله کاراکتری کوچک‌تر در یک دنباله کاراکتری بزرگ**
- **چگونگی تقسیم یک دنباله کاراکتری به بخشهای مختلف**
- **چگونگی حذف فضای خالی ابتدا و انتهای یک دنباله کاراکتری**
- **چگونگی جایگزین کردن بخشهایی از یک دنباله کاراکتری با دنباله‌های کاراکتری دیگر**
- **چگونگی تغییر حروف بزرگ به کوچک و بالعکس در دنباله‌ای از کاراکترها**

در ادامه به بررسی هریک از این موارد می‌پردازیم.

قالب‌بندی دنباله‌های کاراکتری

تاکنون ما به‌سادگی هرآنچه را که قصد نمایش آن‌را بر روی صفحه مرورگر اینترنت داشتیم با بهره‌گیری از تابع ساده‌ای که در زبان PHP به‌همین منظور تهیه شده است یعنی تابع ( ) print چاپ می‌کردیم. علاوه بر این تابع PHP دو تابع مفید دیگر را که امکان قالب‌بندی متن موردنظر را پیش از چاپ آن بر روی صفحه در اختیار قرار می‌دهد، عرضه کرده است. این قالب‌بندی بسیار متنوع بوده و شامل مواردی چون گردکردن اعداد اعشاری با تعداد مشخص اعشار، تعیین شیوه ترازبندی متن در درون یک فیلد با نمایش داده‌ها در سیستمهای عددی مختلف می‌شود. در این قسمت از درس این ساعت قصد داریم تا به امکانات مختلفی دررابطه با قالب‌بندی متون که دو تابع مطرح در این زمینه در اختیار برنامه‌نویسان قرار می‌دهند، بپردازیم. این توابع عبارتند از دو تابع ( ) printf و ( ) sprintf.

استفاده از تابع ( ) printF

اگر پیش از این با استفاده از زبان برنامه‌نویسی C اقدام به نوشتن برنامه‌های هرچند ساده کرده باشید، به‌طور حتم با تابع ( ) printf آشنایی دارید. مشخصه‌ها و عملکرد این تابع در زبان PHP مشابه معادل خود در زبان برنامه‌نویسی C است اما این تشابه به میزان صددرصد نیست. تابع ( ) printf جهت انجام عملیات خود که قالب‌بندی و نمایش متون است نیازمند آرگومانی از نوع دنباله کاراکتری است که معمولاً در میان برنامه‌نویسان با نام "دنباله کاراکتری کنترل کننده شیوه قالب‌بندی" یا format control string شهرت دارد. این تابع همچنین آرگومان‌های دیگری از انواع مختلفی از داده‌ها را نیز به‌عنوان ورودی دریافت می‌کند. دنباله کنترل کننده قالب‌بندی شامل دستورالعملهایی است که چگونگی نمایش خروجی نهایی را که قالب خام (قالب‌بندی نشده) آن‌را به‌عنوان آرگومانی دیگر دریافت می‌کند، مشخص می‌نماید. برای روشن شدن مطلب در اینجا مثالی را با یکدیگر مرور می‌کنیم. عبارت زیر که شامل فراخوانی تابع ( ) printf است یک عدد صحیح را در قالب دهدهی جهت نمایش قالب‌بندی کرده وبه خروجی می‌فرستد:

```
printf ("This is my number : %d" , 55) ;
// prints "This is my number : 55"
```

چنانکه‌دراین‌مثال مشاهده می‌کنید در داخل دنباله کاراکتری کنترل‌کننده قالب‌بندی (آرگومان اول تابع ( ) printf) کد ویژه‌ای تعبیه شده است که مشخص کننده شیوه قالب‌بندی متن توسط این تابع است. این کد معمولاً با نام "مشخصات تبدیل" یا Conversion Specification شناخته می‌شود.

واژه جدید مشخصات تبدیل هر دنباله کنترل‌کننده قالب‌بندی با یک علامت درصد (%) آغاز شده و چگونگی فرآیند قالب‌بندی متن موردنظر (بخش متناظر از متن با این مشخصات تبدیل) را

مشخص می‌کند. استفاده از هر تعداد مشخصات تبدیل در درون دنباله کنترل کننده قالب‌بندی مجاز است به شرطی‌که متناظر با هریک از این مشخصات آرگومانی را جهت قالب‌بندی به این تابع ارسال نماییم.

به مثالی که در این رابطه تهیه شده است، توجه کنید. عبارت زیر با بهره‌گیری از تابع printf ( ) دو عدد صحیح مختلف را که به‌عنوان دو آرگومان مختلف به این تابع ارسال شده‌اند پس از قالب‌بندی به خروجی می‌فرستد:

```
printf ("First number : %d < br > \ n Second number : %d < br > \ n" , 55 , 66) ;
// Output :
// First number : 55
// Scond number : 66
```

در عبارت printf ( ) فوق، اولین مشخصه تبدیل به اولین آرگومان این تابع (با صرف‌نظر از آرگومان اول که دنباله کنترل‌کننده قالب‌بندی است)، یعنی 55 مربوط می‌شود. همچنین دومین مشخصه تبدیل به دومین آرگومان تابع فوق (باز هم به‌شرط صرف‌نظر از دنباله کنترل‌کننده قالب‌بندی)، یعنی 66 مربوط است. کاراکتر d که بعد از علامت درصد در دنباله کنترل‌کننده قالب‌بندی در این عبارت مشاهده می‌شود، موجب می‌گردد که تابع printf ( ) مقدار آرگومان متناظر با این مشخصه تبدیل را به‌عنوان عددی صحیح فرض نماید. کاراکتر d که بخشی از مشخصه تبدیل است، معمولاً با عنوان "مشخصه نوع داده" یا type specifier شناخته می‌شود.

بررسی مشخصه نوع داده‌ها در تابع printf ( )

همان‌گونه که در قسمت قبل مشاهده کردید مشخصه نوع داده d موجب می‌شود تا تابع printf ( ) آرگومان متناظر با مشخصه تبدیل مربوطه را یک عدد صحیح دهدهی (در مبنای ۱۰) فرض کرده و آن‌را با بهره‌گیری از ارقام صفر تا ۹ نمایش دهد. جدول ۱- ۱۷ لیست مربوط به سایر مشخصه‌های نوع داده‌ها به‌همراه جزئیات مربوطه را نشان می‌دهد.

جدول ۱-۱۷ مشخصات نوع داده‌ها

مشخصه	توصیف
d	آرگومان ورودی را به‌عنوان عدد صحیح دهدهی یا دسیمال نمایش می‌دهد.
b	آرگومان ورودی را به‌عنوان عدد صحیح باینری یا دودویی نمایش می‌دهد.
c	آرگومان ورودی را به‌عنوان کد ASCII معادل عدد صحیح نمایش می‌دهد.
f	آرگومان ورودی را به عنوان یک عدد اعشاری (ممیز شناور) نمایش می‌دهد

o	آرگومان ورودی را به‌عنوان عددی در مبنای هشت یا اکتال نمایش می‌دهد.
s	آرگومان ورودی را به‌عنوان یک دنباله کاراکتری نمایش می‌دهد.
x	آرگومان ورودی را به‌عنوان یک عدد در مبنای شانزده (هگزادسیمال) با بهره‌گیری از حروف a تا f به‌عنوان ارقام این مبنا نشان می‌دهد.
X	آرگومان ورودی را به‌عنوان عددی در مبنای شانزده (هگزادسیمال) با بهره‌گیری از حروف A تا F به‌عنوان ارقام این مبنا نشان می‌دهد.

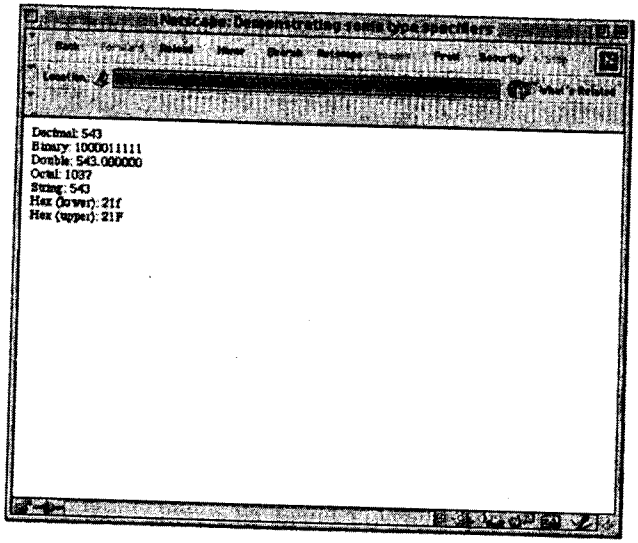
برنامه PHP موجود در لیست ۱- ۱۷ با بهره‌گیری از تابع ( ) printf و استفاده از مشخصه‌های نوع جدول ۱- ۱۷، اقدام به نمایش عدد صحیح 543 در قالبهای صحیح دهدهی، صحیح دودویی، اعشاری، اکتال، دنباله کاراکتری و هگزادسیمال نموده است.

همان‌گونه که در این لیست مشاهده می‌کنید مشخصه‌های تبدیل تنها چیزهایی نیستند که ما به دنباله‌های کنترل‌کننده قالب‌بندی اضافه کرده‌ایم. تابع ( ) printf هرآنچه را که به‌عنوان متن به آن داده می‌شود، در خروجی نمایش خواهد داد.

```
1: <html>
2: <head>
3: <title>Demonstrating some type specifiers</title>
4: </head>
5: <body>
6: <?php
7: $number = 543;
8: printf("Decimal: %d<br>", $number );
9: printf("Binary: %b<br>", $number );
10: printf("Double: %f<br>", $number );
11: printf("Octal: %o<br>", $number );
12: printf("String: %s<br>", $number );
13: printf("Hex (lower): %x<br>", $number );
14: printf("Hex (upper): %X<br>", $number );
15: ?>
16: </body>
17: </html>
```

لیست ۱- ۱۷ بهره‌گیری از مشخصه‌های نوع داده‌ها در تابع ( ) printf

خروجی حاصل از اجرای این برنامه در قالب یک سند وب در شکل ۱- ۱۷ قابل مشاهده است. چنانکه در این شکل ملاحظه می‌کنید تابع ( ) printf روش سریع و مطمئنی جهت تبدیل داده‌های عددی از یک مبنای عددنویسی به مبنای عددنویسی دیگر و نمایش خروجی حاصل است.



شکل ۱- ۱۷ تاثیر استفاده از مشخصه نوع داده‌ها در فراخوانی‌های مختلف تابع ( ) printf

هنگامی که یک رنگ بخصوص را جهت رنگ آمیزی ناحیه‌ای از سند HTML موردنظرتان انتخاب می‌کنید درحقیقت درحال ترکیب سه عدد در مبنای شانزده (هگزادسیمال) هستید. این عدد در محدوده 00 تا FF هستند به ترتیب نماینده طیفی از سه رنگ اصلی قرمز، سبز و آبی می‌باشند (در اصطلاح به این ترکیب RGB گفته می‌شود).

اکنون می‌دانید که با بهره‌گیری از تابع ( ) printf می‌توانیم این اعداد هگزادسیمال را به سه عدد در مبنای دهدهی تبدیل کنیم. یادآوری می‌کنیم که محدوده فوق در مبنای عددی هگزادسیمال معال محدوده صفر تا 255 در مبنای دهدهی است. قطعه کد زیر با استفاده از تابع ( ) printf سه طیف از رنگهای اصلی را به معادل هگزادسیمال تبدیل می‌کند:

```
$red = 204 ;
$green = 204 ;
$blue = 204 ;
printf (" # % X % X % X" , $red, $green, $blue) ;
// prints "# CCCCCC"
```

با وجودی که می‌توان از مشخصه‌های نوع داده جهت تبدیل اعداد دهدهی به هگزادسیمال (و بالعکس) استفاده کرد اما به کمک آن نمی‌توان تعداد کاراکترهایی را که طی هر تبدیل ایجاد می‌شوند، تعیین نمود. در مورد رنگهای موجود در بخشهای مختلف از یک سند HTML و عموماً کدهایی که جهت‌تعیین منحصر به فرد رنگهای موجود مورد استفاده قرار می‌گیرند، هر عدد در مبنای هگزادسیمال باید با دو کاراکتر مشخص شود. کاملاً واضح است که این وضعیت در مواردی منجر می‌شود تا قطعه کد بالا دچار مشکل گردد. برای مثال فرض کنید مقادیر هر سه طیف از سه رنگ اصلی را در قطعه برنامه فوق به‌جای 204 با عدد 1 جایگزین کنیم. در این صورت آنچه که در خروجی مشاهده می‌کنیم، چیزی

جز دنباله کاراکتری 111 # نخواهد بود. با بهره‌گیری از مشخصه خاصی در زبان PHP با عنوان مشخصه padding که در قسمت بعد آن‌را شرح خواهیم داد، می‌توانیم ترتیبی دهیم تا خروجی فوق به‌صورتی که یک رقم صفر پیش از هر یک از ارقام 1 قرار بگیرد اصلاح شود. بدین ترتیب خروجی اصلاح شده به صورت دنباله کاراکتری "010101 #" خواهد بود. در قسمت بعد این خروجی را نیز به دست خواهیم آورد.

اصلاح خروجی حاصل از تبدیل مبناها با بهره‌گیری از مشخصه padding

همواره می‌توانید با بهره‌گرفتن از مشخصه ویژه‌ای در PHP با عنوان مشخصه padding ترتیبی دهید تا در خروجی به‌منظور اصلاح نتیجه عملیات کاراکترهایی اضافی پیش از کاراکترهای موردنظر درج شود. مشخصه padding باید بلافاصله بعد از علامت درصد (٪) که بیانگر یک مشخصه تبدیل است، ظاهر شود. جهت اصلاح خروجی به‌گونه‌ای که یک رقم صفر بی‌ارزش پیش از هریک از کاراکترهای خروجی واقع شود، مشخصه padding باید شامل رقم صفر و به‌دنبال آن تعداد کاراکترهای تشکیل دهنده خروجی باشد. درصورتی که خروجی حاصل از عملیات تعداد کمتری از کاراکترها را نسبت به این تعداد تولید نماید، حد اختلاف آنها با رقم صفر پر خواهد شد. مثال ساده زیر یک فراخوانی از تابع ( ) printf را نشان می‌دهد که خروجی را به‌صورت دنباله‌ای شامل چهار کاراکتر قالب‌بندی می‌کند. به دو صفر بی‌ارزش پیش از عدد 36 در خروجی حاصل از این فراخوانی توجه فرمایید.

```
printf (" %04d ", 36);
// prints " 0036 "
```

مشخصه padding تنها جهت کار با مقادیر عددی طراحی نشده است. درصورتی که بخواهیم خروجی را به‌گونه‌ای اصلاح کنیم که پیش از کاراکترهای قابل نمایش در خروجی از جای خالی استفاده شود، می‌توانیم مشخصه padding را به‌گونه‌ای بسازیم که از یک فضای خالی و به‌دنبال آن تعداد کاراکترهایی که باید در خروجی ظاهر شوند، ذکر گردد. قطعه کد زیر یک فراخوانی ساده تابع ( ) printf را نشان می‌دهد که جهت قالب‌بندی خروجی به‌گونه‌ای که از چهارکاراکتر تشکیل شده و درصورت کمبود کاراکترهای خروجی از فضای خالی به‌جای آنها استفاده شود، طراحی شده است:

```
printf (" % 4d ", 36) ;
// prints " 36 "
```

درحالت معمولی مرورگرهای اینترنت از نمایش فضاهای خالی موجود در اسناد HTML خودداری به‌عمل می‌آورند. درصورتی‌که مایل به نمایش این فضاهای خالی و همچنین تأثیر بهره‌گیری از علامت خط جدید در اسناد HTML هستید، لازم است تا بخش مربوطه از سند HTML خود را در درون علایم آغازین و پایانی <pre> و </pre> قرار دهید. به نمونه زیر در این رابطه توجه نمایید:

```
< pre >
< ? php >
   print " The spaces will be visible " ;
? >
< / pre >
```

درصورتی‌که بخواهید کل یک سند HTML را به‌عنوان یک سند متن قالب کنید، می‌توانید به‌جای محصورکردن کل سند وب در درون علایم آغازین و پایانی فوق از تابع ( ) header جهت تغییر هدر Content _ Type سند موردنظرتان بهره ببرید. به فراخوانی نمونه زیر در همین رابطه توجه نمایید:

```
header ( "Content - Type : Text / plain" ) ;
```

به‌خاطر داشته باشید که جهت حصول نتیجه موردنظر از فراخوانی تابع ( ) header برنامه اسکریپت نباید هیچ‌گونه خروجی را به مرورگر اینترنت ارسال نماید.

مشخصه padding حتی به استفاده از فضای خالی یا رقم صفر نیز محدود نمی‌باشد بدین معنی که هر کاراکتر دیگری را می‌توانید جهت اصلاح خروجی موردنظرتان مورد استفاده قرار دهید به‌شرطی که پیش از آن کاراکتر و درست پس از علامت درصد از یک علامت کوتیشن ساده (') استفاده نمایید. فراخوانی زیر چگونگی عملیات را نشان می‌دهد:

```
printf ( "% ' X 4d", 36) ;
// prints "XX 36"
```

با دانستن مطالب فوق درباره اصلاح خروجی اکنون می‌توانیم کد HTML مربوط به تبدیل و نمایش رنگها را که در قسمت قبل ارایه کردیم، اصلاح نماییم. آنچه که در کد قبلی انجام دادیم تبدیل سه عدد صحیح دهدهی به اعداد هگزادسیمال متناظر بدون اصلاح خروجی حاصل از تبدیل بود. در قطعه برنامه زیر کد مذکور را بازنویسی کرده‌ایم:

```
$red = 1 ;
$green = 1 ;
$blue = 1 ;
printf ( "# %02 X %02X %02X" , $red, $green , $blue) ;
// prints "# 010101"
```

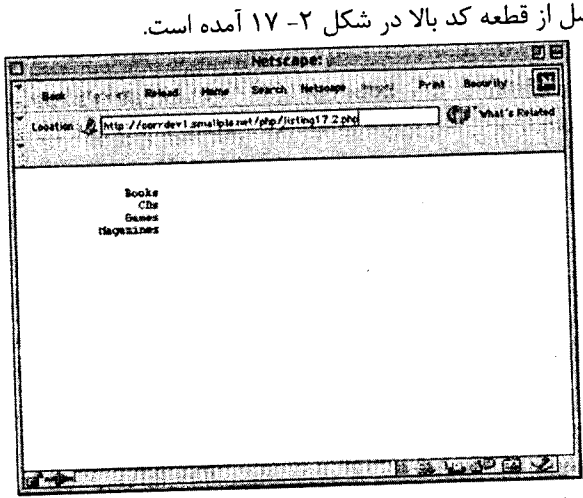
در قطعه برنامه فوق هرعدد هگزادسیمال با استفاده از دو کاراکتر در خروجی نمایش داده می‌شود. چنانچه خروجی حاصل از تبدیل کمتر از دو کاراکتر تولید نماید، فقدان وجود کاراکتر با رقم صفر جبران خواهد شد.

تعیین پهنای یک فیلد

با بهره‌گیری از مشخصه‌ای که در این قسمت مورد بحث قرار خواهیم داد می‌توانید تعداد کاراکترهای قابل درج در خروجی حاصل از اجرای تابع ( ) printf را تعیین کنید. مشخصه پهنا یا width specifier یک عدد صحیح دهدهی است که موقعیت آن در داخل بخش مشخصات تبدیل درست بعد از علامت درصد است (با این پیش‌فرض که هیچ مشخصه دیگری از نوع padding در این بخش مورد استفاده‌قرارنگرفته‌باشد).حاصل‌اجرای قطعه زیر لیستی شامل چهار رقم قلم مختلف از کالاهای گوناگون است که اسامی تمامی آنها در درون حوزه‌ای به پهنای بیست کاراکتر جای گرفته‌اند. همان‌گونه که پیشتر نیز در قالب یک نکته به آن اشاره کردیم جهت نمایش فضاهای خالی موجود در سند HTML دربرگیرنده این خروجی لازم است تا بخش مربوط به خروجی را در درون علایم آغازین و پایانی < per > و < / per > قرار دهیم:

```
Print "< pre >" ;
Printf ( "% 20s \ n", "Books" ) ;
Printf ( "% 20s \ n", "CDs" ) ;
Printf ( "% 20s \ n", "Games" ) ;
Printf ( "% 20s \ n", "Magazines" ) ;
Print "< / pre >" ;
```

خروجی حاصل از قطعه کد بالا در شکل ۲- ۱۷ آمده است.



شکل ۲- ۱۷ تأثیر استفاده از مشخصه پهنا در خروجی حاصل از تابع ( ) printf

چنان‌که در این شکل نیز مشاهده می‌کنید بنا به پیش‌فرض مقادیر خروجی در فیلدها از سمت

راست‌تنظیم می‌شوند. به‌عبارت دیگر، آخرین کاراکتر هرخروجی در آخرین موقعیت از فضای پیش‌بینی شده در فیلد (در اینجا بیستامین موقعیت) واقع می‌شود. درصورتی‌که خواسته باشیم خروجی را از سمت چپ تراز کنیم تنها کافی است تا پیش از مشخصه از یک علامت منفی استفاده نماییم. عبارت زیر چگونگی انجام این کار را نشان می‌دهد:

```
Printf ( "% -20s \ n" , "Left aligned" ) ;
```

توجه کنید که شیوه ترازبندی که در مشخصات تبدیل انتخاب می‌کنید به بخش صحیح هریک از اعداد خروجی اعمال می‌شود. به بیان دیگر، تنها بخشی از اعداد اعشاری که پیش از نقطه اعشار واقع است، متأثر از این شیوه ترازبندی خواهد شد. از این جهت هنگامی که ترازبندی از سمت راست را درمورد یک عدد اعشاری مورد استفاده قرار می‌دهیم تنها بخش صحیح آن یعنی بخشی که قبل از نقطه اعشار واقع شده است، مشمول این قاعده خواهد شد.

تعیین دقت خروجی

هنگام‌ارائه‌خروجی‌حاصل از یک عملیات در قالب اعداد اعشاری با بهره‌گیری از بخش مشخصات تبدیل از دنباله کاراکتری قالب‌بندی می‌توانیم دقت موردنظر خود جهت نمایش این نوع داده‌ها را به‌طور دقیق مشخص نماییم. این امکان به ویژه هنگام کار با مقادیری که نشانگر پول رایج می‌باشند، مفید است (مانند 4. 95 $ که نماینده چهار دلار و نود وپنج سنت است). دقت اعداد اعشاری در خروجی با بهره گیری از مشخصه ویژه‌ای با عنوان مشخصه دقت یا Precision Specifier تعیین می‌شود. مشخصه دقت باید دقیقاً پیش از مشخصه نوع داده در مشخصات تبدیل واقع شود. این مشخصه شامل یک علامت نقطه (به نشانه نقطه اعشار) و به دنبال آن عددی است که بیانگر دقت بخش اعشار از عدد اعشاری می باشد (منظور دقتی است که عدد اعشاری برحسب آن گرد می‌شود). این نوع مشخصه تنها هنگامی بر روی خروجی تأثیر دارد که به‌همراه مشخصه نوع داده f (به نشانه عدد اعشاری) مورد استفاده قرار بگیرد. نمونه‌ای از چگونگی استفاده از این مشخصه جهت قالب‌بندی عدد اعشاری 5.333333 با بهره‌گیری از تابع ( ) printf نشان داده شده است:

```
Printf (" % . 2f ", 5. 333333) ;
// prints "5. 33"
```

درزبان برنامه‌نویسی C این امکان وجود دارد که با بهره‌گیری از تابع ( ) printf و مشخصه، دقت اندازه خروجی یک عددصحیح را نیز تعیین کنیم. در زبان برنامه‌نویسی PHP4 دقت کنید که مشخصه دقت هیچ‌گونه تأثیری بر روی اعداد صحیحی که با استفاده از تابع ( ) printf قالب‌بندی می‌شوند، ندارد. از این‌رو برخلاف تابع ( ) printf در زبان C در اینجا جهت درج فضای خالی یا رقم بی‌ارزش صفر پیش از یک عدد صحیح در خروجی حاصل از این تابع، مانند گذشته لازم است تا از مشخصه padding استفاده نماییم.

مروری دقیق‌تر بر مشخصات تبدیل

جدول ۲- ۱۷ شامل لیستی از مشخصه‌هایی است که می‌توان از آنها جهت ساخت مشخصات تبدیل بهره گرفت. دقت کنید که سطرهای این جدول منطبق بر ترتیبی است که این مشخصه‌ها در مشخصات تبدیل واقع می‌شوند. همان‌گونه که احتمالاً متوجه شده‌اید بهره‌گیری توأم از دو مشخصه پهنای فیلد و مشخصه padding اندکی مشکل است. از این‌رو همواره بهتر است تنها یکی از این دو مشخصه را جهت قالب‌بندی مورد استفاده قرار دهید.

جدول ۲- ۱۷ اجزای تشکیل‌دهنده مشخصات تبدیل

نام مشخصه	توصیف عملکرد	مثال
مشخصه padding	این مشخصه تعداد کاراکترهایی را تعیین می‌کند که باید با استفاده از تابع ( ) printf در خروجی نمایش داد. همچنین مشخصه فوق کاراکتری را نیز تعیین می‌کند که درصورت کمبود کاراکترهای خروجی به خروجی حاصل اضافه می‌شود.	'4'
مشخصه پهنای فیلد	این مشخصه فضایی را تعیین می‌کند که خروجی حاصل از تابع ( ) printf باید در آن فضا قالب بندی شده و به نمایش گذاشته شود.	'20'
مشخصه دقت	این مشخصه دقت موردنظر جهت گردکردن اعداد اعشاری حاصل از خروجی تابع ( ) printf (تعداد ارقام اعشار) را نشان می‌دهد.	'.4'
مشخصه نوع داده	این مشخصه نوع داده خروجی حاصل از تابع ( ) printf را تعیین می‌کند.	'd'

برنامه موجود در لیست ۲- ۱۷ به‌منظور نمایش لیستی از محصولات به‌همراه قیمت هرکدام از آنها از تابع ( ) printf به‌همراه مشخصه‌های جدول ۲- ۱۷ جهت قالب‌بندی خروجی استفاده می‌کند.

```
1: <html>
2: <head>
3: <title>Using printf() to format a list of product prices</title>
4: </head>
5: <body>
6: <?php
7: $products = Array(    "Green armchair"=>"222.4",
8:                       "Candlestick"=>"4",
9:                       "Coffee table"=>80.6
10:                      );
```

لیست ۲- ۱۷ بهره‌گیری از تابع ( ) printf جهت نمایش لیست محصولات و قیمت هریک از آنها

```
11: print "<pre>";
12: printf("%-20s%23s\n", "Name", "Price");
13: printf("%'-43s\n", "");
14: foreach ( $products as $key=>$val )
15:     printf( "%-20s%20.2f\n", $key, $val );
16: print("</pre>");
17: ?>
18: </body>
19: </html>
```

دنباله لیست ۲-۱۷

همان‌گونه که مشاهده می‌کنید، ابتدا در خط ۷ از این برنامه یک آرایه انجمنی شامل اسامی کالاها و بهای مربوط به هریک از آنها ایجاد کرده‌ایم. نکته قابل ذکر در اینجا بهره‌گیری از نشانه <per> است. این نشانه چنانکه قبلاً نیز اشاره کردیم، موجب خواهد شد تا مرورگر اینترنت مورد استفاده کاربر فضاهای خالی و علایم خط جدیدی را که توسط برنامه به قالب‌بندی خروجی اضافه می‌شود همان‌گونه که هست، نشان دهد. تأثیر این فرآیند از نشانه آغازین < per > شروع شده و با نشانه < / per > خاتمه پیدا می‌کند. اولین فراخوانی تابع ( ) printf را در خط ۱۲ از برنامه انجام داده‌ایم. اکنون با دانشی که در زمینه قالب‌بندی خروجی به‌دست آورده‌اید، می‌توانید دنباله کاراکتری کنترل کننده قالب‌بندی این فراخوانی را که به صورت زیر است، تفسیر نمایید:

" % -20s % 23s \ n "

اولین مشخصه تبدیل در دنباله کاراکتری فوق یعنی " 20s- % " شامل مشخصه پهنای فیلدی است که اندازه‌ای برابر با بیست کاراکتر را مشخص می‌کند. علامت منفی در این مشخصه بیانگر این است که خروجی حاصل از این قالب‌بندی باید از سمت چپ تراز شود. همچنین مشخصه نوع داده مورد استفاده در این تبدیل از نوع دنباله کاراکتری است که با بهره‌گیری از کاراکتر ' s ' مشخص شده است. دومین مشخصه تبدیل از دنباله کاراکتری کنترل کننده قالب‌بندی به‌صورت " 23s % " نوشته شده است که پهنای فیلدی را به اندازه بیست و سه کاراکتر تعیین می‌کند. عدم وجود علامت منفی در این مشخصه بدین معنی است که ترازبندی خروجی به‌طور پیش‌فرض (یعنی به‌طور ترازبندی از سمت راست) انجام می‌شود. تابع ( ) printf در خط ۱۲ با این دنباله کاراکتری قالب‌بندی جهت نمایش مورد فراخوانی قرار می‌گیرد.

دومین فراخوانی تابع( ) printf که در خط ۱۳ انجام شده است با قرار دادن متوالی کاراکتر – پشت سر هم به تعداد ۴۳ مرتبه اقدام به ترسیم خطی درست در زیر عناوین فیلدهای Name و Price می‌کند. این قابلیت چنانکه مشاهده می‌کنید با بهره‌گرفتن از مشخصه padding به دست آمده است.

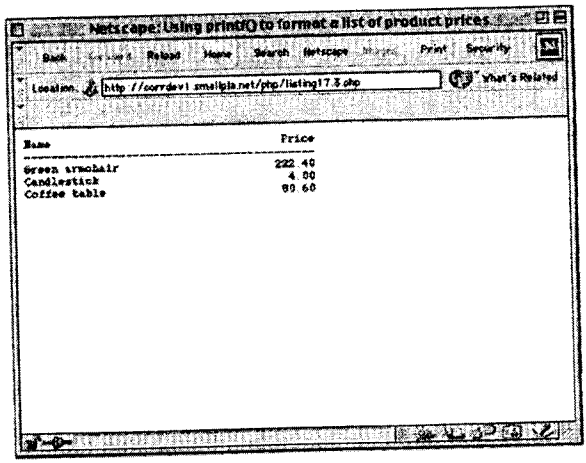
آخرین فراخوانی تابع ( ) printf که در خط ۱۵ از برنامه انجام شده است، بخشی از یک ساختار تکرار foreach است که با هربار گذر از حلقه به ازای عناصر آرایه‌ای که اسامی محصولات را نگه می‌دارد مورد فراخوانی قرار می‌گیرد. در فراخوانی این تابع ( ) printf مانند اولین فراخوانی از دو مشخصه تبدیل

استفاده کرده‌ایم. اولین مشخصه تبدیل که به صورت " 20s- % " نوشته شده‌است، موجب می‌شود تا نام محصول به‌عنوان یک دنباله کاراکتری در خروجی نمایش داده شود. وجود علامت منفی در این مشخصه نشانگر ترازبندی خروجی از سمت چپ است. پهنای فیلد چنانکه ملاحظه می‌کنید برابر با بیست کاراکتر انتخاب شده است. دومین مشخصه تبدیل که به شکل " 20. 2 f % " نوشته شده است، موجب می‌شود تا بهای محصول موردنظر در فیلدی با پهنای بیست کاراکتر به نمایش درآید. عدم وجود علامت منفی بدین معنی است که ترازبندی خروجی قالب‌بندی شده توسط این مشخصه بر مبنای ترازبندی پیش‌فرض یعنی ترازبندی از سمت راست انجام شود. همچنین مشخصه دقت که به صورت ". 2" تعیین شده است، باعث خواهد شد تا بهای کالا در قالب عددی اعشاری با دقتی برابر با دو رقم اعشاری در خروجی به نمایش درآید.

خروجی حاصل از این برنامه در شکل ۳- ۱۷ قابل مشاهده است.

جابه‌جایی آرگومان ها

با انتشار نسخه PHP 4.0.6 این امکان در اختیار برنامه‌نویس قرارگرفت که درصورت لزوم با بهره‌گرفتن از دنباله کاراکتری کنترل‌کننده قالب‌بندی خروجی ترتیب آرگومان‌های ارسالی به تابع را جهت نمایش خروجی تغییر دهد.



شکل ۳- ۱۷ خروجی برنامه لیست ۲- ۱۷ که با بهره‌گیری از قابلیت قالب‌بندی تابع ( ) printf ایجاد شده است

برای مثال فرض کنید که مایلیم تاریخهای مشخصی را بر روی صفحه مرورگر اینترنت نمایش دهیم. برای این‌کار تاریخهای موردنظرمان را در یک آرایه چندبعدی ذخیره کرده و سپس با بهره‌گیری از تابع ( ) printf آنها را در جهت نمایش در خروجی با روشهایی که تا بدین جای درس فراگرفتیم

قالب‌بندی می‌کنیم. لیست۳-۱۷ برنامه PHP مورد نیاز جهت تعریف آرایه‌ای شامل تاریخها و دستورالعملهای خروجی را نشان می‌دهد.

```
<?
$dates = array (
            array ('mon' => 12, 'mday' =>25, 'year' =>2001 ),
            array ('mon' => 5, 'mday' =>23, 'year' =>2000 ),
            array ('mon' => 10, 'mday' =>29, 'year' =>2001 ),
            ) ;
$format = include ("local_format.php");
foreach ($dates as $date ) {
    printf ( "$format" , $date['mon'], $date['mday'], $date['year'] );
}
?>
```

چنانکه در این لیست مشاهده می‌کنید، به طور مستقیم دنباله کاراکتری کنترل‌کننده قالب‌بندی را در درون تابع ( ) printf مورد استفاده قرار نداده و درعوض از دنباله کاراکتری کنترل‌کننده واقع در فایلی با عنوان local _ format . php جهت قالب‌بندی تاریخهای موردنظر در خروجی استفاده کرده ایم. این فایل با بهره‌گرفتن از تابع ( ) include در برنامه ما شامل شده است.

با فرض اینکه فایل مذکور شامل قطعه برنامه کوچک زیر باشد:

```
< ? php
return "%02d /% 02d /% d < br >" ;
? >
```

خروجی حاصل از برنامه لیست ۳- ۱۷ در قالب mm / dd / yy به صورت زیر نمایش خواهد یافت:

```
12 / 25 / 2001
05 / 23 / 2000
10 / 29 / 2001
```

اکنون چنین تصور کنید که قصد داریم تا از برنامه موجود در لیست ۳- ۱۷ در یک سایتی که موقعیت فیزیکی آن در بریتانیاست، استفاده کنیم. همان‌گونه که اطلاع دارید نحوه نمایش تاریخ در ممالک بریتانیایی به‌گونه‌ای است که عدد روز پیش از عدد ماه قرار می‌گیرد. به‌عبارت دیگر شکل عمومی یک تاریخ در این قالب نمایش به‌صورت dd / mm / yy است. واضح است که بخش اصلی برنامه جهت انجام این تغییر نباید دستخوش تغییر شود. آنچه که در این میان باید تغییر کند قواعد قالب‌بندی موجود در فایل local _ format . php است. خوشبختانه بواسطه قابلیتهایی که PHP4.0.6 در اختیارمان قرار می‌دهد می‌توانیم ترتیبی را که مقادیر آرگومان‌ها در خروجی حاصل از قالب‌بندی نمایش داده می‌شوند با تغییراتی که در ترتیب عناصر دنباله کاراکتری کنترل‌کننده قالب‌بندی خروجی می‌دهیم به‌دلخواه خود تنظیم نماییم. در این موردخاص به‌منظور دستیابی به نوع قالب‌بندی تاریخ بریتانیایی کافی است تا فایل مذکور را به شکل زیر تغییر دهیم:

```
< ? php
return " % 2 \ $02d / % 1 \ $02d / % 3 \ $d < br >;
? >
```

همان‌گونه که مشاهده می‌کنید به راحتی می‌توانیم شماره (ترتیب) آرگومان موردنظرمان را پس از علامت درصد (که علامت شروع مشخصه تبدیل است) ذکر کرده و به دنبال آن از علامت $ که با بهره‌گیری از یک علامت \ پیش از آن تحت پوشش قرار گرفته است، استفاده نماییم. از این‌رو آنچه که از کد موجود در فایل local _ format . php برمی‌آید، این است که مایلیم تا دومین آرگومان پیش از همه در خروجی به نمایش درآمده و به‌دنبال آن اولین آرگومان و درنهایت سومین آرگومان در خروجی به‌نمایش درآید. به‌عبارت دیگر این کد به‌سادگی باعث می‌شود تا جای آرگومان‌های اول و دوم در خروجی برنامه با یکدیگر تعویض شود. نتیجه حاصل از این تغییر، خروجی است که در سه خط زیر مشاهده می‌کنید:

```
25 / 12 / 2001
23 / 05 / 2000
29 / 10 / 2001
```

ثبت و ذخیره یک دنباله کاراکتری قالب‌بندی شده

تابع ( ) printf به‌سادگی خروجی خود را بر روی مرورگر اینترنت می‌فرستد. این بدان معنی است که نتایج محاسبه شده در اختیار برنامه PHP قرار نمی‌گیرند. با این وجود می‌توان با بهره‌گیری از تابع ( ) sprintf بر این محدودیت تابع ( ) printf فایق آمد. تابع ( ) sprintf از بسیاری جهات عملکردی مشابه با تابع ( ) printf دارد. تفاوت بسیار مهمی که با این حال مابین این دو تابع وجود دارد، این است که دنباله کاراکتری بازگشتی از تابع ( ) sprintf را به‌جای ارسال به خروجی و نمایش در صفحه مرورگراینترنت می‌توان در یک متغیر به‌منظور استفاده‌های بعدی در برنامه ذخیره نمود. قطعه کد زیر که از این تابع جهت قالب‌بندی استفاده می‌کند، قادر است تا یک عدد اعشاری را ابتدا تا دو رقم اعشار گرد نموده و سپس نتیجه حاصل از این عمل را به‌جای ارسال به خروجی در متغیری با نام $dosh ذخیره کند:

```
$dosh = sprintf ( " % . 2f " , 2. 334454) ;
print "You have $dosh dollars to spend" ;
```

یکی از استفاده‌های مهمی که برنامه‌نویسان از تابع ( ) sprintf به‌عمل می‌آورند، این است که سعی می‌کنند تا داده‌های قالب‌بندی شده را در یک فایل ذخیره نمایند. بدین ترتیب که آنها تابع ( ) sprintf را فراخوانی کرده و مقدار بازگشتی حاصل از آن‌را به یک متغیر نسبت داده و درنهایت با بهره‌گیری از تابع ( ) fputs مقدار متغیر مذکور را در یک فایل چاپ می‌کنند.

بررسی دقیق درباره دنباله‌های کاراکتری

یکی‌اززوایای‌تاریک‌برنامه‌نویسی‌این است که همواره نمی‌توان هرگونه اطلاعاتی درمورد داده‌هایی که در برنامه مشغول استفاده از آنها هستیم، به دست آورد. دنباله‌های کاراکتری از آن دسته از داده‌هایی هستند که از هر مرجعی ممکن است، تولید شده باشند. این منابع ممکن است فیلدهایی از یک فرم HTML باشند که کاربران اطلاعاتی را در درون آنها وارد می‌کنند، یا داده‌هایی باشند که در قالب رکوردهایی از جداول موجود در بانکهای اطلاعاتی به دست آمده‌اند یا مقادیری باشند که به‌سادگی در درون فایل‌های متن ذخیره شده‌اند و یا از آن‌هم ساده‌تر ممکن است از طریق بخشهای مختلف یک سند وب تامین شده باشند. پیش از اینکه کار خود را با داده‌هایی که از این منابع خارجی به دست آمده‌اند، آغاز نماییم اغلب لازم است تا اطلاعات بیشتری درمورد آنها به دست آوریم. خوشبختانه php4 توابع بسیار کارآمد و مفیدی را در اختیارمان قرار داده که با بهره‌گیری از آنها می‌توانیم به اطلاعات بسیار ارزنده‌ای درباره دنباله‌های کاراکتری موردنظرمان دست پیدا کنیم.

نکته‌ای در باب شاخص‌گذاری دنباله‌های کاراکتری

در قسمتهای باقیمانده از درس این ساعت ما به‌طور مکرر از واژه شاخص (index) دررابطه با دنباله‌های کاراکتری بهره خواهیم گرفت. چنانچه خاطرتان باشد تا بدین جای کتاب از این واژه به کرات درمورد آرایه‌ها و بحثهای مربوطه استفاده نمودیم. درواقع شاید دانستن این نکته برای شما بسیار جالب باشد که آن‌گونه که ممکن است تصور کرده باشید آرایه‌ها و دنباله‌های کاراکتری مقوله‌های بسیار مجزایی از یکدیگر نیستند. همواره می‌توانیم دنباله‌های کاراکتری را به منزله آرایه‌هایی درنظر بگیریم که محتوای هریک از عناصر آنها تنها یک کاراکتر است. به‌واسطه این رویکرد جدید می‌توانیم همان‌گونه که عناصر آرایه‌ها را مورد دستیابی قرار می‌دهیم به همان ترتیب نیز کاراکترهای تشکیل دهنده یک دنباله کاراکتری را نیز از طریق شماره‌هایی که به‌عنوان شاخص آن کاراکترها درنظر می‌گیریم، مورد دستیابی قرار دهیم. قطعه کد زیر این رویکرد را در عمل نشان می‌دهد. در قطعه کد ابتدا دنباله کاراکتری $test با محتوای " scallywag " ایجاد شده و سپس کاراکترهای اول و سوم آن که به‌ترتیب با شاخصهای صفر و ۲ مشخص شده‌اند، مورد دستیابی قرار گرفته و در خروجی چاپ شده‌اند:

```
$test = "scallywag" ;
print $test [0] ; // prints "s"
print $test [2] ; // prints "a"
```

به‌خاطر سپردن این نکته درمورد دنباله‌های کاراکتری حائز اهمیت زیادی است که هنگام صحبت درمورد موقعیت یا محل قرارگیری یک کاراکتر در دنباله‌ای از کاراکترها (یا همان شاخص

کاراکتر در یک دنباله کاراکتری)، مانند آنچه که در مورد آرایه‌ها شاهد بودیم موقعیت اولین کاراکتر را با شاخص صفرام و بقیه را نسبت به آن می‌سنجیم.

تعیین طول دنباله‌ای از کاراکترها با بهره‌گیری از تابع ( ) strlen

با بهره‌گیری از تابع( ) strlen می‌توانیم طول دنباله‌های کاراکتری را تعیین کنیم. دقت کنید که منظور از طول دنباله کاراکتری تعداد کاراکترهای تشکیل‌دهنده آن است نه شماره شاخص آخرین کاراکتر موجود در آن (بدین ترتیب طول دنباله‌های کاراکتری همواره یکی بیشتر از شماره شاخص آخرین کاراکتر موجود در دنباله است چراکه همان‌گونه که قبلاً اشاره کردیم شاخص‌گذاری کاراکترهای موجود در یک دنباله کاراکتری مانند شاخص‌گذاری عناصر آرایه‌ها از عدد صفر آغاز می‌شود). ساختار این تابع بسیار ساده بوده و جهت اجرای عملیات پیش‌بینی شده تنها نیازمند نام متغیری است که دنباله کاراکتری موردنظر را ذخیره می‌کند (می‌توان دنباله کاراکترها را نیز مستقیماً به‌عنوان آرگومان به این تابع ارسال نمود). آنچه که این تابع به‌عنوان نتیجه عملیات باز می‌گرداند عدد صحیحی است که نمایانگر تعداد کاراکترهای دنباله کاراکتری ارسال شده به این تابع می‌باشد. از تابع ( ) strlen درمواردی چون محاسبه طول دنباله‌های کاراکتری که کاربر در فیلد خاصی از یک فرم ورودی HTML وارد کرده است استفاده می‌شود. در قطعه کد زیر از تابع ( ) strlen جهت اطمینان از اینکه کد عضویت ذخیره شده در متغیر $membership شامل تنها چهار رقم (نه کمتر و نه بیشتر) است، استفاده شده است. به چگونگی استفاده از آن جهت تشکیل یک عبارت تصمیم‌گیری در ساختار if توجه کنید:

```
if (strlen ($membership) = = 4 )
   print "Thank you ! " ;
else
   print "Your membership number must have 4 digits < p >" ;
```

چنانکه در قطعه کد فوق مشاهده می‌کنید اولین تابع ( ) print تنها هنگامی فراخوانی خواهد شد که تعداد کاراکترهای دنباله کاراکتری ذخیره شده در متغیر $membership دقیقاً برابر با چهار عدد باشد. چنانچه این تعداد کمتر یا بیشتر از چهار کاراکتر باشد دومین تابع ( ) print در بخش else از ساختار if / else فوق جهت نمایش یک پیغام خطا مورد فراخوانی قرار خواهد گرفت.

یافتن‌یک‌دنباله‌کاراکتری‌کوچک در یک دنباله کاراکتری بزرگ‌تر با بهره‌گیری از تابع ( )strstr

همان گونه که پاراگراف‌ها از جملات و جملات از کلمات مختلف تشکیل شده‌اند، می‌توان دنباله‌های کاراکتری را نیز طبق یک رویکرد جدید متشکل از اجزایی فرض کرد که خود از نوع دنباله کاراکتری هستند. به‌عبارت دیگر می‌توان یک دنباله کاراکتری را به‌عنوان دنباله‌ای از چندین دنباله

کاراکتری کوچک‌تر فرض کرد. بدین ترتیب می‌توان به‌منظورهای مختلفی دنباله‌های کاراکتری گوناگونی را در درون‌یکدیگرموردجستجوقرار داد. خوشبختانه در این مورد نیز PHP تابعی را جهت استفاده برنامه‌نویس پیش‌بینی کرده است. با بهره‌گیری از تابع( ) strstr می‌توان یک دنباله کاراکتری کوچک را دردرون‌یک دنباله کاراکتری بزرگ‌تر مورد جستجو قرار داد. ساختار این تابع نسبتاً ساده بوده به‌گونه‌ای که به دو آرگومان ورودی جهت انجام عملیات موردنظر نیاز دارد. اولین آرگومان یک دنباله کاراکتری را مشخص می‌کند که فرآیند جستجو در درون آن صورت می‌گیرد و دومین آرگومان نیز دنباله کاراکتری مورد جستجو را مشخص می‌نماید. درصورتی که دنباله کاراکتری مورد جستجو (دومین آرگومان) در درون دنباله کاراکتری اصلی (اولین آرگومان) یافت نشود، تابع ( ) strstr مقدار false را به نشانه عدم موفقیت به برنامه فراخواننده باز می‌گرداند. اما چنانچه دنباله کاراکتری مورد جستجو در درون دنباله کاراکتری اصلی یافت شود تابع مورد بحث بخشی از دنباله کاراکتری اصلی را که با دنباله کاراکتری مورد جستجو آغاز شده است به برنامه فراخواننده باز خواهد گرداند. برای روشن شدن مطلب اجازه دهید تا مثالی را در این زمینه با هم مرور کنیم. فرض کنید که در آخرین مثال قسمت قبل آن دسته از کاربرانی که کد عضویت آنها شامل دنباله کاراکتری "AB" باشد برای ما اهمیت خاص داشته باشند. از این‌رو لازم است تا قطعه برنامه‌ای بنویسیم که چنین کاربرانی را با تعیین کد عضویت مربوطه از سایرین متمایز نماید. برای این منظور کافی است تا با بهره‌گیری از تابع strstr ( ) در کد عضویت کاربران موجود به‌دنبال دنباله کاراکتری "AB" بگردیم. قطعه برنامه زیر چگونگی انجام این فرآیند را نشان می‌دهد:

```
$membership = "PAB 7" ;
if (strstr ($membership, "AB") )
   print "Thank you. Don't forget that your membership expires soon ! " ;
else
   print "Thank you ! " ;
```

چنانکه در قطعه کد فوق مشاهده می‌کنید، متغیر $membership شامل کد عضویت "PAB7" است که البته دنباله کاراکتری "AB" را در بر می‌گیرد. از این جهت تابع ( ) strstr دنباله کاراکتری "AB 7" را به برنامه فراخواننده باز می‌گرداند، یعنی بخش ویژه‌ای از دنباله کاراکتری اصلی که با دنباله کاراکتری مورد جستجو ("AB") آغاز می‌شود. به دلیل آنکه دنباله کاراکتری نتیجه توسط PHP به‌صورت true ارزیابی می‌شود اولین فراخوانی تابع ( ) print موجب چاپ آن بر روی صفحه نمایش خواهد شد. در اینجا این پرسش منطقی ممکن است به ذهن برسد که اگر کاربر دنباله کاراکتری "pad 7" را به‌عنوان کد عضویت خود وارد کرده باشد، آیا باز هم نتیجه فوق حاصل خواهد شد یا خیر؟ تابع ( ) strstr نسبت به بزرگی و کوچکی حروف مورد جستجو حساس است از این‌رو این تابع در یافتن دنباله کاراکتری "AB" در کد عضویت کاربر ناموفق بوده و مقدار false را به برنامه فراخواننده باز می‌گرداند. به‌عبارت دیگر، در قطعه کد فوق عبارت شرطی‌بخش if از ساختار تصمیم‌گیری به‌صورت

false ارزیابی شده و بنابراین دومین تابع ( ) print اقدام به نمایش پیغام "!Thank you" بر روی صفحه می‌کند. درصورتی که بخواهیم چنین حساسیتی مابین حروف بزرگ و کوچک در یافتن دنباله‌های کاراکتری در میان نباشد یا به‌عبارت بهتر هر دو دنباله کاراکتری "AB" و "ab" (یا حتی ترکیبی از آنها مانند "aB" یا "Ab") به شکل یکسانی در فرآیند جستجوی دنباله کاراکتری موردنظر ارزیابی شوند، به‌جای بهره‌گیری از تابع ( ) strstr می‌توانیم از تابع دیگری با نام ( ) stristr استفاده کنیم. عملکرد هر دو تابع دقیقاً مشابه یکدیگر بوده و از نظر ساختار کاملاً یکسان هستند. تنها تفاوت در این است که تابع ( ) stristr برخلاف تابع ( ) strstr نسبت به بزرگی و کوچکی حروف موجود در دنباله کاراکتری مورد جستجو حساس است.

یافتن موقعیت یک دنباله کاراکتری کوچک در درون یک دنباله کاراکتری بزرگ‌تر با بهره‌گیری از تابع ( ) strpos

به‌کمک تابع ( ) strpos می‌توان دریافت که آیا یک دنباله کاراکتری در درون یک دنباله کاراکتری دیگر موجود می‌باشد یا خیر و درصورتی که پاسخ مثبت باشد موقعیت دنباله کاراکتری مورد جستجو را در درون دنباله کاراکتری اصلی مشخص می‌نماید. این تابع جهت اجرای عملیات پیش‌بینی شده احتیاج به دو آرگومان ورودی دارد که از این نظر بسیار شبیه به دو تابع قبل، یعنی توابع ( ) strstr و ( ) stristr می‌باشد. آرگومان اول تابع ( ) strpos بیانگر دنباله کاراکتری اصلی است، یعنی دنباله کاراکتری که فرآیند جستجو در درون آن انجام می‌شود. دومین آرگومان این تابع، مطابق آنچه انتظار داریم دنباله کاراکتری مورد جستجو است. این تابع برخلاف دو تابع قبل آرگومان سومی را نیز به‌عنوان یک آرگومان اختیاری می‌پذیرد. این آرگومان اختیاری که یک عدد صحیح است موقعیتی را در دنباله کاراکتری اصلی مشخص می‌کند که فرآیند جستجو برای یافتن دنباله کاراکتری کوچک‌تر از آن موقعیت آغاز می‌شود. چنانچه تابع ( ) strpos در فرآیند یافتن دنباله کاراکتری مورد جستجو موفق نشود، مقدار false را به‌عنوان نتیجه عملیات به برنامه فراخواننده باز می‌گرداند. اما درصورتی‌که این تابع بتواند رشته کاراکتری موردنظر را در درون دنباله کاراکتری اصلی پیدا کند موقعیت شروع آن، یعنی شاخص مربوط به اولین کاراکتر از دنباله کاراکتری مورد جستجو در درون دنباله کاراکتری اصلی را به برنامه فراخواننده باز خواهد گرداند. اجازه دهید تا برای روشن شدن موضوع مثالی را در اینجا ارائه دهیم. در قطعه برنامه زیر سعی کرده‌ایم تا با بهره‌گرفتن از تابع ( ) strpos از وجود دنباله کاراکتری که با "mz" در درون یک دنباله کاراکتری بزرگ‌تر آغاز می‌شود اطلاع حاصل کنیم. متغیر $membership به مانند قبل حاوی دنباله کاراکتری اصلی است:

```
$membership = "mzxyz" ;
if (strpos ($membership, "mz" ) = = = 0 )
   print "hello mz" ;
```

به‌ترفندی که در این قطعه برنامه جهت تحصیل نتیجه موردنظر استفاده کرده‌ایم توجه کنید. آنچه‌مسلم‌است‌تابع( ) strposموفق به یافتن دنباله کاراکتری "mz" در دنباله کاراکتری "MZOOXYZ" که در متغیر $membership ذخیره شده است، خواهد شد اما نکته مهم اینجاست که تابع فوق دنباله کاراکتری "mz" را درست در ابتدای دنباله کاراکتری اصلی یعنی در شاخص شماره صفر پیدا می‌کند. این بدان معنی است که تابع ( ) strpos مقدار عددی صفر را به نشانه موفقیت‌آمیز بودن عملیات به برنامه فراخواننده باز می‌گرداند. این وضعیت اکنون باید زنگهای خطر را درگوش شما به‌صدا درآورده باشد؛ چراکه در دنیای برنامه‌نویسی و به‌طور خاص PHP عدد صفر معمولاً به منزله false ارزیابی می‌شود.

چنانکه مشاهده می‌کنید، جهت غلبه بر این مشکل خاص از عملگر جدیدی که در PHP معرفی شده و با عنوان عملگر بررسی معادل بودن (equivalence) شناخته می‌شود، استفاده کرده‌ایم. این عملگر که ظاهر آن به شکل سه علامت تساوی متوالی است (= = =) درصورتی که عملوندهای سمت چپ و راست معادل (برابر یا مساوی) یکدیگر بوده و ضمناً هر دو از یک نوع داده واحد باشند مقدار true و در غیر این‌صورت مقدار false را به برنامه فراخواننده باز می‌گرداند.

استخراج بخشی از یک دنباله کاراکتری به‌عنوان یک دنباله کاراکتری کوچک‌تر با استفاده از تابع ( ) substr

گاهی اوقات لازم است تا بخشی از یک دنباله کاراکتری جهت انجام یک بررسی خاص از یک دنباله کاراکتری بزرگ‌تر استخراج شود. با بهره‌گیری از تابعی با نام ( ) substr در زبان PHP می‌توانیم قسمتی از یک دنباله کاراکتری را برمبنای شاخص شروع موردنظر و طول بخشی از دنباله کاراکتری که به آن علاقه‌مندیم استخراج کنیم. ساختار تابع ( ) substr ساده بوده و جهت انجام عملیات موردنظر تنها به دو آرگومان ورودی نیاز دارد. اولین آرگومان این تابع یک دنباله کاراکتری است که مایلیم بخشی از آن‌را استخراج نماییم. دومین آرگومان تابع فوق یک عدد صحیح است که موقعیت شروع (شاخص کاراکتری از دنباله اصلی که عملیات جداسازی از آنجا آغاز می‌شود) را تعیین می‌کند. تابع substr ( ) تمام کاراکترهایی از دنباله موردنظر را که از موقعیت شروع (دومین آرگومان) آغاز شده و تا انتهای دنباله کاراکتری مذکور ادامه دارد به‌عنوان نتیجه عملیات و در قالب یک دنباله کاراکتری جدید به برنامه فراخواننده باز می‌گرداند. تابع ( ) substr به‌عنوان یک آرگومان اختیاری آرگومان سومی را نیز دریافت می‌کند. این آرگومان اختیاری که یک عدد صحیح است طول (تعداد کاراکترهای) موردنظر از دنباله کاراکتری اصلی را جهت استخراج مشخص می‌کند. درصورتی که از این آرگومان اختیاری در فراخوانی تابع ( ) substr استفاده شود این تابع تنها تعدادی از کاراکترهایی را که به برنامه فراخواننده باز می‌گرداند که از نقطه شروع آغاز شده و به اندازه مشخص شده در این آرگومان ادامه پیدا کند. قطعه

برنامه نمونه زیر چگونگی بهره‌گیری ازاین تابع را جهت استخراج یک دنباله کاراکتری از درون دنباله کاراکتری "scallywag" نشان می‌دهد. اولین فراخوانی تابع ( ) substr عملیات استخراج را از موقعیت کاراکتر ششم (هفتمین کاراکتر) آغاز کرده و تا انتهای دنباله کاراکتری ادامه می‌دهد. حال آنکه دومین فراخوانی از همان موقعیت به اندازه دو کاراکتر را به‌عنوان نتیجه باز می‌گرداند:

```
$test = "scallywag" ;
print substr ($test , 6) ;        // prints "wag"
print substr ($test , 6 , 2) ;    // prints "wa"
```

درصورتی که هنگام فراخوانی تابع ( ) substr از یک عدد صحیح منفی به‌عنوان دومین آرگومان ورودی (موقعیت شروع فرآیند استخراج) استفاده کنیم تابع مذکور به‌صورت معکوس عمل می‌کند به‌گونه‌ای که به‌جای شمارش از موقعیت شروع به انتهای دنباله کاراکتری شمارش درجهت عکس از نقطه شروع به سمت ابتدای دنباله کاراکتری انجام می‌شود. ارایه یک مثال در این زمینه می‌تواند نکات مبهم‌راآشکارکند.درقطعه‌برنامه‌زیر سعی شده‌است تا به آن‌دسته از کاربرانی که آدرس پست الکترونیکی آنها با پسوند uk . خاتمه می‌یابد پیغامی ارسال شود:

```
$test = "matt @ corrosive. Co. uk " ;
if ($test = substr ($test, -3) = = ". uk " )
   print "Don't forget our special offers for British customers" ;
else
   print "Welcome to our shop ! " ;
```

تجزیه دنباله‌های کاراکتری به اجزای سازنده با استفاده از تابع ( ) strtok

تجزیه دنباله‌های کاراکتری به عناصر تشکیل دهنده آن از بسیاری جهات می‌تواند مفید واقع گردد. با بهره‌مندی از تابع ویژه‌ای که در زبان PHP ارائه شده است، می‌توان یک دنباله کاراکتری را بر مبنای کلمات تشکیل‌دهنده آن به بخشهای مجزایی تقسیم نمود. نام این تابع مفید ( ) strtok است. این تابع جهت انجام عملیات خود تنها به دو آرگومان ورودی نیاز دارد. اولین آرگومان این تابع دنباله کاراکتری است که قصد تجزیه آن‌را داریم. دومین آرگومان تابع ( ) strtok خود یک دنباله کاراکتری است که در حقیقت مبنای فرآیند تجزیه کاراکتری اصلی را تعیین می‌کند. این دنباله کاراکتری درصورت نیاز می‌تواند شامل تعداد دلخواهی از کاراکترها باشد. تابع ( ) strtok به‌عنوان نتیجه عملیات اولین بخش از بخشهای تشکیل‌دهنده دنباله کاراکتری اصلی را به برنامه فراخواننده باز می‌گرداند (منظور از اولین بخش آن بخش از دنباله کاراکتری اصلی است که از ابتدای آن آغاز شده و تا محل وقوع دنباله کاراکتری تعیین شده توسط دومین آرگومان ادامه دارد). پس از اولین فراخوانی تابع ( )strtok دنباله کاراکتری در محلی از حافظه ثبت خواهد شد. بدین ترتیب در فراخوانی‌های بعدی این تابع، نیازی به ارسال این دنباله کاراکتری به‌عنوان اولین آرگومان به این تابع نبوده و تنها کافی است تا دنباله کاراکتری که مبنای فرآیند تجزیه را مشخص می‌کند به‌عنوان تنها آرگومان به این تابع ارسال

گردد. به‌عبارت دیگر اولین فراخوانی تابع ( ) strtok نیازمند دو آرگومان و فراخوانی‌های بعدی آن نیازمند تنها یک آرگومان است. هربار که این تابع فراخوانی می‌شود بخش دیگری از دنباله کاراکتری اصلی که توسط دنباله کاراکتری تعیین شده به‌عنوان آرگومان از سایر بخشها تفکیک شده است را به برنامه فراخواننده باز می‌گرداند. این رویه تا بدان جا ادامه می‌یابد که کلیه اجزای تشکیل دهنده دنباله کاراکتری تفکیک شده باشند. در این وضعیت چنانچه تابع ( ) strtok باردیگر فراخوانی شود مقدار false را به برنامه فراخواننده باز خواهد گرداند؛ چرا که در این‌صورت هیچ بخشی از دنباله کاراکتری اصلی جهت تفکیک باقی‌نمانده است. معمولاً تابع ( ) strtok در قالب یک ساختار تکرار، بارها و بارها مورد فراخوانی برنامه اصلی (برنامه‌ای که آن‌را مورد استفاده قرار می‌دهد) واقع می‌گردد. این بدان جهت است که در هربار فراخوانی این تابع بخشی از دنباله کاراکتری اصلی را به برنامه فراخواننده باز می‌گرداند. برای روشن شدن نکات مبهم در این مورد اجازه دهید تا موضوع را با بررسی یک مثال در این زمینه پیگیری نماییم. برنامه موجود در لیست ۳- ۱۷ از تابع ( ) strtok جهت تجزیه یک آدرس URL و تفکیک نام میزبان و مسیر موردنظر از دنباله پرس و جو (بخشی از آدرس که به دنبال علامت ? واقع می‌شود) استفاده می‌کند. این برنامه همچنین بخشهای نام و مقدار موجود در دنباله پرس و جو را نیز از یکدیگر تفکیک می‌نماید.

```
 1: <html>
 2: <head>
 3: <title>Listing 17.3 Dividing a string into
 4:         tokens with strtok()</title>
 5: </head>
 6: <body>
 7: <?php
 8: $test  = "http://www.deja.com/qs.xp?";
 9: $test .= "OP=dnquery.xp&ST=MS&DBS=2&QRY=developer+php";
10: $delims = "?&";
11: $word = strtok( $test, $delims );
12: while ( is_string( $word ) ) {
13:    if ( $word )
14:        print "$word<br>";
15:    $word = strtok( $delims);
16: }
17: ?>
18: </body>
19: </html>
```

لیست ۳- ۱۷ تجزیه یک دنباله کاراکتری با بهره‌گیری از تابع ( ) strtok

دقت کنید که تابع ( ) strtok یک ابزار کارآمد است و استفاده از آن درصورتی‌که برنامه‌نویس اندکی هوش و ذکاوت از خود به‌خرج دهد، نتایج بسیار جالب توجهی را به‌بار خواهد آورد. همان‌گونه که در برنامه موجود در این لیست مشاهده می‌کنید ابتدا دنباله کاراکتری را که فرآیند تجزیه دنباله کاراکتری اصلی بر مبنای آن صورت خواهد گرفت در خط ۱۰ از برنامه در متغیری با عنوان $delims

ذخیره کرده‌ایم. در خط ۱۱ ضمن فراخوانی تابع ( ) strtok آدرس URL موردنظر جهت تجزیه به اجزای تشکیل دهنده و نیز دنباله کاراکتری ذخیره شده در متغیر $delims را به‌ترتیب به‌عنوان آرگومان‌های اول و دوم به این تابع ارسال نموده‌ایم. چنانکه در خط ۱۱ ملاحظه می‌کنید اولین بخش تفکیک شده از آدرس URL در متغیری با نام $word ذخیره شده است. در خط ۱۲ از برنامه با بهره‌گیری از تابع ( ) is _ string که پیشتر به معرفی آن پرداختیم از وجود یک دنباله کاراکتری در درون این متغیر اطمینان حاصل نموده و نتیجه این بررسی را در قالب یک ساختار تکرار از نوع while مورد استفاده قرار داده‌ایم. درصورتی‌که متغیر $word حاوی یک دنباله کاراکتری نباشد، می‌توانیم به این نتیجه برسیم که فرآیند تفکیک آدرس URL به اجزای تشکیل‌دهنده به پایان رسیده و هیچ بخش دیگری جهت پردازش باقی نمانده است.

با اندکی دقت متوجه خواهید شد که ما فرآیند بررسی مقدار بازگشتی حاصل از تابع ( ) strtok را از آن‌جهت انجام می‌دهیم که اگر آدرس URL (یا هر دنباله کاراکتری مورد تجزیه دیگر) شامل دو دنباله کاراکتری جداکننده (دنباله کاراکتری که در اولین فراخوانی تابع ( ) strtok به‌عنوان دومین آرگومان و در سایر فراخوانی‌های این تابع به‌عنوان تنها آرگومان مورد استفاده قرار می‌گیرد) متوالی باشد، تابع مورد بحث هنگام مواجه شدن با اولین آنها یک دنباله کاراکتری تهی را به برنامه فراخواننده باز خواهد گرداند. از این‌رو بهره‌گیری از ساختار while به‌صورت ساده زیر است:

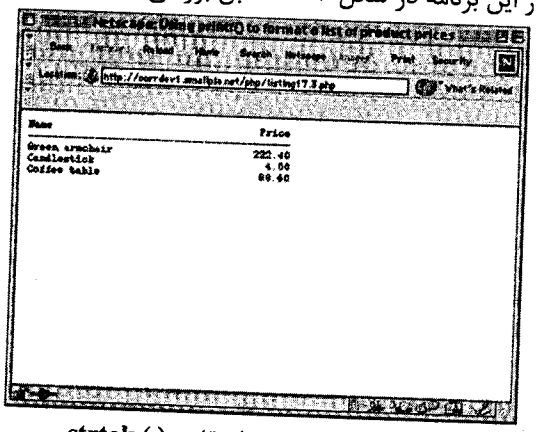
```
While ($word) {
   $word = strtok ($delims) ;
}
```

درصورتی‌که متغیر مورد ارزیابی یعنی $word حاوی یک دنباله کاراکتری تهی باشد، موجب می‌شود تا حاصل عبارت شرطی این ساختار معادل با مقدار false فرض شده و بنابراین در داخل بدنه ساختار تکرار متغیر مذکور مجدداً مقداردهی نشود و این وضعیت حتی درصورتی‌که فرآیند تفکیک و تجزیه دنباله کاراکتری اصلی به انتهای خود نرسیده باشد، بازهم مشاهده خواهد شد.

از طرف دیگر، با بررسی اینکه متغیر $word حاوی دنباله کاراکتری است می‌توان کار را ادامه داد. درصورتی‌که این متغیر حاوی یک دنباله کاراکتری تهی نباشد، دستورالعمل ( ) print در خط ۱۴ از برنامه موجب می‌شود تا محتوای متغیر فوق بر روی صفحه مرورگر اینترنت به نمایش درآید. همان‌گونه که مشاهده می‌کنید تابع ( ) strtok در خط ۱۵ مجددا باعث مقداردهی متغیر فوق به‌ازای گذر بعدی حلقه while می‌شود. توجه کنید که در فراخوانی تابع ( ) strtok برای دومین‌بار از ارسال دنباله کاراکتری مورد نظر به آن به عنوان آرگومان خودداری شده است. درصورتی‌که برنامه‌نویس به اشتباه مرتکب چنین خطایی شود، باردیگر مقدار بازگشتی حاصل از اولین فراخوانی تابع ( ) strtok به دست می‌آید و این بدان معنی است که چنین اتفاقی در سایر فراخوانی‌های تابع فوق نیز روی داده و هدف از به‌کارگیری این تابع که تفکیک اجزای تشکیل‌دهنده یک دنباله کاراکتری است، تأمین نخواهد

شد. به‌بیان ساده اجرای برنامه در یک حلقه بی‌پایانی سرگردان خواهد ماند بدون اینکه برنامه‌نویس نتیجه موردنظر خود را از برنامه دریافت کرده باشد. اطلاع از چگونگی بهره‌گرفتن از توابع اهمیت زیادی دارد که باید به آن توجه کرد.

خروجی حاصل از این برنامه در شکل ۳- ۱۷ قابل بررسی است.



شکل ۳- ۱۷ استفاده از تابع ( ) strtok

دستکاری دنباله‌های کاراکتری

زبان برنامه‌نویسی PHP4 توابع متعددی را در اختیار برنامه‌نویس قرار داده که با استفاده از آنها می‌توان تغییراتی اندک و یا در حد اساسی بر روی دنباله‌های کاراکتری اعمال کرد. در ادامه به بحث درمورد این‌گونه توابع خواهیم پرداخت.

مرتب‌سازی و سامان‌دهی دنباله‌های کاراکتری با بهره‌گیری از توابع Ltrim ( ), trim ( ) و strip_ tags ( )

هنگامی که دنباله کاراکتری موجود در برنامه را از طریق منابعی چون فرمهای ورودی HTML یا فایل‌های متن موجود در سیستم فایل به دست می‌آورید، ممکن است متوجه شوید که در برخی از موارد قدری فضای خالی به انتها یا ابتدای این دنباله‌های کاراکتری ضمیمه شده است. برای مثال ممکن است کاربر پیش از واردکردن نام و نام خانوادگی خود یا هرگونه اطلاعات دیگری در فیلدهای مربوطه از یک فرم ورودی HTML ابتدا (به‌طور خواسته یا نا خواسته) با زدن کلید space بر روی صفحه کلید اقدام به درج قدری فضای خالی به ابتدا یا انتهای اطلاعات وارد شده نماید. این‌گونه فضاهای خالی در بسیاری از موارد از جمله پردازش دنباله‌های کاراکتری نامطلوب بوده و لازم است تا با به‌کاربردن شیوه‌ای آنها را از ابتدا یا انتهای دنباله‌های کاراکتری حذف کنیم. خوشبختانه توابع ارائه

شده در PHP4 چنین امکاناتی را در رابطه با دستکاری دنباله‌های کاراکتری در اختیار کاربران قرار داده است.

با بهره‌گیری از تابعی با نام trim( )، می‌توانیم هرگونه فضای خالی ایجاد شده در اثر زدن کلیدهایی چون space, Tab و یا Enter را از ابتدا و انتهای دنباله‌های کاراکتری موردنظر حذف کنیم. این تابع ساختاری کاملاً ساده داشته به‌گونه‌ای که تنها به یک آرگومان ورودی نیاز دارد. این آرگومان همان دنباله کاراکتری است که باید فضاهای خالی موجود در ابتدا و انتهای آن توسط تابع مذکور حذف شوند. آنچه که این تابع به برنامه فراخواننده باز می‌گرداند دنباله کاراکتری جدیدی است که محتوای آن دقیقاً مشابه قبل است؛ با این تفاوت که فضاهای خالی از ابتدا و انتهای آن حذف شده‌اند. برای روشن شدن مطلب به مثال زیر توجه کنید. در این قطعه کد تابع trim ( ) جهت اصلاح دنباله کاراکتری ذخیره شده در متغیر $text فراخوانی شده است. چنانکه ملاحظه می‌کنید ابتدا و انتهای این دنباله کاراکتری شامل قدری فضای خالی است:

```
$text =   "\t\t\t lots of room to breath       " ;
$text = trim ($text) ;
print $text ;
// prints   "lots of room to breath ",
```

شاید عملکرد تابع trim ( ) درمواقعی بیش از آنچه که شما بدان نیاز دارید، باشد. برای مثال ممکن است مایل باشید تا فضای خالی ابتدای دنباله کاراکتری را حفظ کنید و ضمناً تمایلی نیز ندارید که هیچ‌گونه فضای خالی به انتهای دنباله کاراکتری موردنظرتان ضمیمه شده باشد. باز هم PHP4 تابعی را ارائه کرده و به کمک آن می‌توانید عملکرد مورد نیازتان را تأمین نمایید. ابزار معرفی شده در این زمینه تابعی با عنوان rtrim ( ) است. این تابع از نظر ساختار (آرگومان دریافتی) کاملاً مشابه تابع trim ( ) بوده و تنها کافی است تا دنباله کاراکتری موردنظرتان را به‌عنوان آرگومان به آن ارسال نمایید.

با این حال تفاوت ظریفی در عملکرد این دو تابع وجود دارد. در حالی‌که تابع trim ( ) فضای خالی ناخواسته را از ابتدا و انتهای دنباله‌های کاراکتری پاک می‌کند، تابع rtrim ( ) ضمن حفظ فضای خالی موجود در ابتدای دنباله کاراکتری فضای خالی ناخواسته را از انتهای دنباله کاراکتری (یعنی انتهایی که تایپ آن دنباله کاراکتری در آن سمت به پایان می‌رسد) حذف می‌نماید. بار دیگر ارائه یک قطعه کد نمونه به روشن شدن موارد مبهم کمک می‌کند. در قطعه کد زیر فراخوانی تابع rtrim ( ) باعث می‌شود تا تنها فضای خالی موجود در انتهای دنباله کاراکتری موجود در متغیر $text حذف شود. فضاهای خالی ایجاد شده با استفاده از " \t " در ابتدای این دنباله کاراکتری دست‌نخورده باقی می‌ماند:

```
$text =   "\t\t\t lots of room to breath       " ;
$text = rtrim ($text) ;
print $test ;
   // prints  "       lots of room to breath";
```

همان‌گونه که به‌احتمال قوی حدس زده‌اید، PHP مکانیزم مشابهی را جهت انجام فرآیند فوق در جهت معکوس پیش‌بینی کرده است؛ بدین ترتیب که این مکانیزم فضاهای خالی موجود در انتهای یک دنباله کاراکتری را حفظ کرده و تنها فضاهای خالی موجود در ابتدای آن دنباله را حذف می‌کند. این مکانیزم در قالب تابعی با عنوان ( ) ltrim پیاده‌سازی شده است. باردیگر تأکید می‌کنیم که ساختار این تابع نیز دقیقاً مشابه دو تابع قبل یعنی ( ) trim و ( ) rtrim است؛ بدین معنی که تنها به یک آرگومان که نماینده دنباله کاراکتری موردنظر است، نیاز داشته و حاصل عملیات آن دنباله کاراکتری مشابه با آرگومان دریافتی است که فضاهای خالی ایجاد شده به‌واسطه زدن کلید Space, Tab و یا Enter از ابتدای آن حذف شده‌اند. جهت روشن شدن این موضوع به قطعه کد زیر که از این تابع استفاده می‌کند توجه نمایید:

```
$text = " \t \t \t lots of room to breath      " ;
$text = Ltrim ($text) ;
print $test ;
// prints   "lots of room to breath      " ;
```

چنانکه در جریان هستید، PHP در اصل جهت کار با متون نشانه‌گذاری شده (مانند متون HTML) طراحی شده است.

بنابراین شاید غیرمعمول نباشد اگر بخواهیم عوامل نشانه‌گذاری را از یک متن نشانه‌گذاری شده حذف کرده و متن ساده باقیمانده را مورد استفاده قرار دهیم. PHP روش بسیار آسانی را جهت تحصیل این هدف در اختیار برنامه‌نویس قرار داده است. طراحان زبان برنامه‌نویسی PHP با تعبیه نمودن کلیه مکانیزم‌ها و کدهای موردنیاز جهت حذف نشانه‌ها از یک متن نشانه‌گذاری شده در تابعی با عنوان strip _ tags ( ) زحمت برنامه‌نویس را به‌منظور انجام این فرآیند به حداقل ممکن رسانده‌اند به‌گونه‌ای که برنامه‌نویس برای انجام این فرآیند تنها کافی است تا این تابع را یک مرتبه فراخوانی کند. ساختمان این تابع از نقطه‌نظر فراخوانی بسیار ساده است به‌گونه‌ای که تنها به یک آرگومان ورودی جهت انجام عملیات موردنظر خود نیاز دارد. تنها آرگومان این تابع متن نشانه‌گذاری شده موردنظر را که می‌تواند در قالب یک متغیر ذخیره شده باشد، مشخص می‌کند. تابع فوق آرگومان دیگری را نیز به‌عنوان یک آرگومان اختیاری دریافت می‌کند. این آرگومان اختیاری لیستی شامل نشانه‌های HTML است که تابع مورد بحث آنها را در فرآیند نشانه‌زدایی از متن اصلی شرکت نمی‌دهد. تنها نکته مهمی که در استفاده از این آرگومان اختیاری وجود دارد این است که اسامی نشانه‌های موجود در لیستی که به‌عنوان آرگومان دوم به این تابع ارسال می‌شود نباید با هیچ‌گونه فضای خالی از یکدیگر جدا شوند. باردیگر بررسی یک نمونه در باب چگونگی استفاده از این تابع می‌تواند نقاط تاریک را روشن کند. در قطعه کد زیر یک متن نشانه‌گذاری شده کوتاه ابتدا در متغیری با نام $string ذخیره شده و سپس به‌عنوان اولین آرگومان به تابع strip _ tags ( ) ارسال شده است:

```
$string = "I <i > simply < / i > will not have it, < br > said Mr.
```

```
Dean < p > < b > The end < / b >" ;
Print strip _ tags ($string, "< br > < p >") ;
```

چنانکه در این قطعه برنامه کوتاه مشاهده می‌کنید متن ساده‌ای را که با بهره‌گیری از نشانه‌های HTML قالب‌بندی شده است، در قالب متغیر $string به تابع ( ) strip _ tags ارسال شده و ضمناً آرگومان دوم این تابع چنین مشخص می‌کند که نشانه‌های < br > و < p > در عملیات نشانه‌زدایی تابع نامبرده شرکت داده نمی‌شوند. بدین ترتیب آنچه که به‌عنوان نتیجه عملیات این تابع حاصل می‌شود همان متن ذخیره شده در متغیر $string است که همه نشانه‌های HTML به استثنای دو نشانه < br > و < p > از آن حذف شده‌اند.

جایگزینی بخشی از یک دنباله کاراکتری با یک دنباله کاراکتری دیگر با بهره‌گیری از تابع substr _ replace ( )

پیشتر درمورد تابع substr ( ) در درس این ساعت به بحث و بررسی پرداختیم. اگر خاطرتان باشد این تابع با دریافت یک دنباله کاراکتری و موقعیتی در درون آن اقدام به استخراج بخشی از دنباله کاراکتری ورودی، از موقعیت مشخص شده توسط دومین آرگومان تا انتهای دنباله کاراکتری یا تا موقعیتی از آن که توسط سومین آرگومان ورودی تعیین می‌شد، می‌کرد. تابع مشابهی که در این قسمت بررسی می‌کنیم تابعی با نام substr _ replace ( ) است. این تابع مشابه تابع substr ( ) است با این تفاوت که به برنامه‌نویس اجازه می‌دهد تا بخش استخراج شده از دنباله کاراکتری را با یک دنباله کاراکتری دیگر جایگزین نماید. بدین ترتیب برنامه‌نویس در صورت تمایل می‌تواند بخشهایی از یک دنباله کاراکتری را با بخشهای مورد نظرش تعویض کند. این تعویض یکی از آن تغییرات اساسی است که پیشتر به آن اشاره کردیم. ساختمان این تابع از نقطه‌نظر آرگومان‌های ورودی نسبتاً پیچیده‌است چراکه این تابع در ساده‌ترین حالت ممکن جهت اجرای عملیات پیش‌بینی شده به سه آرگومان ورودی نیاز دارد. اولین آرگومان این تابع یک دنباله کاراکتری را مشخص می‌کند که عملیات تعویض بخشی از آن توسط تابع صورت خواهد گرفت. آرگومان دوم بازهم یک دنباله کاراکتری است که با بخشی از دنباله کاراکتری اصلی تعویض خواهد شد. آرگومان سوم این تابع موقعیتی از دنباله کاراکتری اصلی است که عملیات تعویض از آن نقطه آغاز می‌شود (منظور از موقعیت شروع شاخص کاراکتری است که در آنجا واقع می‌باشد). تابع substr _ replace ( ) آرگومان دیگری را نیز به‌عنوان آرگومان اختیاری می‌پذیرد. این آرگومان اختیاری که یک عدد صحیح است طول بخش تعویضی از دنباله کاراکتری اصلی را نشان می‌دهد. تابع substr _ replace ( ) به‌سادگی بخشی از دنباله کاراکتری اصلی را که باید تعویض شود با بهره‌گیری از موقعیت نقطه شروع و طول بخش تعویضی تشخیص داده وآن‌را با دنباله کاراکتری جدیدی که در آرگومان دوم این تابع مشخص شده است، تعویض می‌کند. آنچه که این تابع به‌عنوان

نتیجه عملیات به برنامه فراخواننده باز می‌گرداند یک دنباله کاراکتری جدید است که بخشی از آن با یک دنباله کاراکتری دیگر تعویض شده است.

از آنجا که ساختار استفاده از این تابع اندکی مشکل به‌نظر می‌رسد ارائه یک مثال ساده به‌طور حتم می‌تواند نکات مبهم این فرآیند را به‌خوبی روشن کند. در قطعه برنامه‌ای که در ادامه ملاحظه می‌کنید با بهره‌گیری از تابع substr _ replace ( ) اقدام به تعویض کد عضویت یک کاربر نموده‌ایم. همان‌گونه که مشاهده می‌کنید تابع مذکور در این کد نمونه کاراکترهای موجود در موقعیتهای دوم و سوم کد عضویت کاربر را با دنباله کاراکتری " 0 0 " تعویض می‌کند:

```
< ?
$membership = "mz99xyz" ;
$membership = substr _ replace ( $membership, "00", 2, 2) ;
print "New membership number is $membership < p >" ;
// prints "New membership number : mz00xyz"
? >
```

جایگزینی دنباله‌های کاراکتری با بهره‌گیری از تابع str _ replace ( )

تابع str _ replace ( ) عملکرد تابع بررسی شده در قسمت قبل، یعنی تابع substr _ replace( ) را توسعه می‌دهد به‌گونه‌ای که این تابع تمامی نمونه‌های یک دنباله کاراکتری را در درون یک دنباله کاراکتری دیگر با دنباله جدیدی از کاراکترها تعویض می‌کند. این تابع نیز مانند تابع مشابه خود جهت اجرای عملیات پیش‌بینی شده احتیاج به دریافت سه آرگومان ورودی دارد. نخستین آرگومان، دنباله کاراکتری مورد جستجو را مشخص می‌کند. دومین آرگومان، دنباله کاراکتری جایگزین را تعیین می‌نماید و بالاخره سومین و آخرین آرگومان این تابع دنباله کاراکتری اصلی را که عملیات تعویض بر روی بخشهای مختلف آن صورت می‌پذیرد، تعیین می‌کند. حاصل عملیات این تابع دنباله کاراکتری جدیدی است که بخشهای مختلف آن با یک دنباله کاراکتری دیگر تعویض شده‌اند. باردیگر ارائه مثال به روشن‌تر شدن مطلب کمک خواهد کرد. در قطعه کد زیر تابع str _ replace ( ) تمامی سالهای 2000 در دنباله کاراکتری ذخیره شده در متغیر $string را با سال 2001 تعویض کرده و نتیجه را به‌عنوان خروجی نمایش می‌دهد:

```
$string = "Site contents copyright 2000. " ;
$string ·="The 2000 Guide to All Things Good in Eurpe" ;
print str _ replace ("2000", "2001", $string) ;
```

قابلیتهای تابع str _ replace ( ) به همین‌جا منتهی نمی‌شود. با انتشار PHP4. 05 تابع مورد بحث از قابلیتهای بیشتری نیز برخوردار شده است. به‌گونه‌ای که اکنون قادر است تا هرجا که دنباله کاراکتری را به‌عنوان آرگومان ورودی دریافت می‌کند، آرایه‌ای را نیز دریافت نماید. این توسعه ارزشمند درعملکرد تابع str _ replace ( ) به برنامه‌نویس اجازه می‌دهد تا به‌جای جستجوی یک دنباله کاراکتری در درون دنباله کاراکتری اصلی، از چندین الگوی جستجو برای یافتن بخشهای موردنظر از دنباله

کاراکتری اصلی بهره‌مند شود و حتی از آن هم بیشتر، اجازه می‌دهد تا بخشهای یافته شده در دنباله کاراکتری اصلی هریک با دنباله جدیدی از کاراکترها جایگزین شوند. به‌عبارت بسیار ساده کاری که این تابع در نسخه‌های پیشین طی چند فراخوانی پی در پی انجام می‌داد اکنون قادر است تا تنها با یک فراخوانی به انجام رساند و این به مفهوم سرعت و دقت در توسعه برنامه‌ها است. برنامه موجود در لیست ۵- ۱۷ قابلیتهای جدید این تابع را به‌سادگی نشان می‌دهد. باردیگر یادآوری می‌کنیم که دستیابی به این قابلیتها تنها در PHP4 .05 و نسخه‌های بالاتر امکان‌پذیر است.

```
<?php
$source = array (
        "The package which is at version 4.2 was released in 2000" ,
    "The year 2000 was an excellent period for PointyThing4.2" );
$search = array ("4.2" , "2000");
$replace=array ("5.0","2001");
$source=str-replace ($search, $replace, $source);
foreach ($source as $str)
   print "$str<br>";

// prints:
// The package which is at version 5.0 was released in 2001
// The year 2001 was an excellent period for PointyThing5.0
?>
```

لیست ۴- ۱۷ بهره‌گیری از قابلیتهای جدید تابع ( ) str _ replace

همان‌گونه که در این برنامه PHP مشاهده می‌کنید تابع ( ) str _ replace دو آرایه $search و $replace را به‌عنوان آرگومان‌های اول و دوم دریافت می‌کند. آرگومان سوم این تابع با نام $source خود آرایه‌ای است که شامل دو دنباله کاراکتری مختلف می‌باشد. تابع ( ) str _ replace با دریافت دو آرایه مذکور به‌عنوان آرگومان‌های اول و دوم ورودی سعی می‌کند تا بخشهایی از آرگومان سوم خود را که مشابه مقادیر ذخیره شده در عناصر اولین آرایه (دنباله‌های کاراکتری مورد جستجو) است با مقادیر متناظر موجود در عناصر دومین آرایه (آرایه جایگزین) تعویض نماید. به‌عبارت ساده‌تر این تابع جای دنباله‌های کاراکتری" 4.2 "و "2000" از دنباله کاراکتری اصلی را به‌ترتیب با دنباله‌های کاراکتری "5.0" و "2001" جایگزین خواهد کرد. چنانچه سومین آرگومان تابع ( ) str _ replace خود از نوع آرایه‌ای شامل دنباله‌های کاراکتری باشد (مانند آنچه که در این برنامه مشاهده نمودید) مقداری که به‌عنوان نتیجه عملیات این تابع به برنامه اصلی بازگردانده می‌شود نیز یک آرایه شامل دنباله‌های کاراکتری خواهد بود. دنباله‌های کاراکتری موجود در آرایه بازگشتی مشابه دنباله‌های کاراکتری موجود در آرایه اصلی(سومین آرگومان ورودی‌تابع)خواهند بود با این تفاوت که بخشهایی از آنها طبق الگوهای مشخص شده در آرگومان‌های اول و دوم تعویض شده‌اند.

تبدیل حروف کوچک و بزرگ به یکدیگر

زبان برنامه‌نویسی PHP چندین تابع مختلف را جهت تبدیل حروف کوچک و بزرگ موجود در دنباله‌های کاراکتری به یکدیگر در اختیار برنامه‌نویس قرارداده است. این‌گونه تبدیلات از بسیاری جهات برای برنامه‌نویسان مفید است. برای این مثال هنگامی که داده‌های دریافتی از طریق ورودی فرمهای HTML را در یک فایل متن ساده یا در جداولی از یک بانک اطلاعاتی ذخیره می‌کنیم، معمولاً تمایل داریم تا پیش از ذخیره آنها کلیه حروف را به شکل همگن مثلاً به‌صورت کوچک و بزرگ درآوریم. این فرآیند کمک می‌کند تا عملیات مقایسه‌ای که در آینده برروی این داده‌ها انجام می‌دهیم، بسیار ساده‌تر و مطمئن‌تر صورت بگیرد. برای تبدیل حروف کوچک یک دنباله کاراکتری به حروف بزرگ متناظر کافی است تا از تابع ( ) strtoupper استفاده کنیم. ساختار این تابع بسیار ساده و آسان است و تنها کافی است تا دنباله کاراکتری موردنظرتان را جهت انجام تبدیل فوق به‌عنوان تنها آرگومان به این تابع ارسال کنید. تابع( ) strtoupper به‌عنوان نتیجه عملیات دنباله کاراکتری جدیدی را که کلیه حروف آن اکنون‌به‌صورت بزرگ درآمده است به برنامه فراخواننده باز می‌گرداند. قطعه کد زیر نمونه‌ای از چگونگی استفاده از این تابع را جهت تبدیل دنباله کاراکتری ذخیره شده در متغیری با نام $membership نشان می‌دهد:

```
$membership = "mz00xyz" ;
$membership = strtoupper ($membership) ;
print "$membership < p >" ;   // prints  "MZ00XYZ"
```

همان‌گونه که حدس زده‌اید تابع دیگری در PHP عهده‌دار انجام فرآیند معکوس، یعنی تبدیل حروف‌بزرگ‌موجوددریک‌دنباله کاراکتری‌به‌حروف کوچک است.این تابع به‌طور مشابه ( ) strtolower نام دارد. ساختار به‌کارگیری این تابع نیز بسیار ساده بوده و جهت انجام عملیات تبدیل پیش‌بینی شده تنها به یک آرگومان ورودی که همان دنباله کاراکتری موردنظر باشد، نیاز دارد. مقدار بازگشتی حاصل از این تابع‌یک‌دنباله‌کاراکتری است که محتوای آن دقیقاً مشابه محتوای آرگومان ورودی است؛با این تفاوت که تنهاازحروف‌کوچک در آن استفاده شده است. به قطعه برنامه کوتاه زیر که در آن از تابع ( ) strtolowerجهت تبدیل موردنظر استفاده شده است، توجه نمایید:

```
$home _ url = "WWW. CORROSIVE. CO. UK" ;
$home _ url = strtolower ($home _ url) ;
if (! (strpos ($home _ url, "http : // ") = = = 0 ) )
    $home _ url = "http : // $home _ url " ;
print $home _ url ; // prints "http : // www. corrosive.co. uk "
```

علاوه بر این دو تابع که فرآیند کوچک یا بزرگ‌سازی حروف یک دنباله کاراکتری را به‌طور مطلق و بدون هیچ‌گونه پیش‌شرطی انجام می‌دهند، PHP تابع دیگری را نیز در این زمینه معرفی کرده است که فرآیند تبدیل را به‌طور محدودتری انجام می‌دهد. این تابع که ( ) ucwords نام دارد، قادر است تا تنها اولین کاراکتر از هریک از کلمات موجود در یک دنباله کاراکتری را به حروف بزرگ تبدیل نماید.

این‌تابع‌سایرحروف‌کلمات‌رادست‌نخورده باقی می‌گذارد. درقطعه‌کدزیربا بهره‌گیری از تابع ( ) ucwords اولین حروف از کلیه کلمات موجود در دنباله کاراکتری ذخیره شده در متغیر $full_name به حروف بزرگ تبدیل شده و حاصل این عملیات با استفاده از تابع ( ) print به خروجی ارسال شده است:

```
$full _ name = "violet elizabeth bott " ;
$full _ name = ucwords ($full _ name) ;
print $full _ name ; // prints "Violet Elizabeth Bott"
```

چنانکه در این قطعه کد مشاهده می‌کنید تنها حروف موجود در ابتدای کلمات از دنباله کاراکتری $full _ name یعنی سه حرف v, e و b که به‌صورت کوچک نوشته شده‌اند مشمول عملیات تبدیل تابع ( ) ucnwords شده و سایر حروف بدون تغییر مانده‌اند. از این رو درصورتی‌که کلید shift موجود بر روی صفحه کلید کاربر دچار مشکل شده باشد و او دنباله‌ای از کاراکترها را به‌صورت "VIolEt eLIZaBeTH BOTt" از طریق فرم ورودی به برنامه ارسال کرده باشد با بهره‌گیری از تابع ucwordes نمی‌توان دنباله کاراکتری فوق را مطابق انتظار اصلاح کرد؛ چرا که آنچه از تابع فوق در مورد این دنباله کاراکتری به دست می‌آید دنباله کاراکتری "ViolEt ELIZaBeTH BOTt" است که با دنباله کاراکتری مطلوب به‌کلی متفاوت است. با این حال با بهره‌گیری از این تابع به‌همراه تابع دیگری که در پاراگراف قبل مورد بحث قرار دادیم، یعنی تابع ( ) strtolower می‌توانیم به دنباله کاراکتری موردنظر دست یابیم. برای این منظور کافی است تا پیش از ارسال دنباله کاراکتری دریافتی کاربر به تابع ( ) ucwords آن‌را جهت پردازش به‌گونه‌ای که تمامی حروف بزرگ آن به حروف کوچک تبدیل شوند به تابع ( ) strtolower ارسال نماییم. قطعه برنامه زیر شیوه فوق را جهت رسیدن به نتیجه مطلوب مورد استفاده قرار داده است:

```
$full _ name = "VIolEt eLIZaBeTH bOTt" ;
$full _ name = ucwords (strtolower ($full _ name) ) ;
print $full _ name ; // prints "Violet Elizabeth Bott"
```

قالب‌بندی متون با استفاده از تابع ( ) wordwarp و ( ) nl2br

هنگامی که متن ساده‌ای (یک متن قالب‌بندی نشده) را بر روی صفحه مرورگر اینترنت به‌عنوان یک سند وب یا بخشی از آن نمایش می‌دهید در اغلب موارد با این مشکل مواجه می‌شوید که علایم خط جدید (newline) نمایش پیدا نمی‌کنند (یا به عبارت دیگر تأثیر خود را از دست می‌دهند). بدین ترتیب با متنی مواجه می‌شوید که ترکیب ظاهری مطلوب خود را از دست داده است. زبان PHP روشی را در اختیار برنامه‌نویس قرار داده است که با بهره‌گیری از آن می‌تواند کلیه علایم خط جدید را به نشانه < br > در HTML تبدیل نماید. این مکانیزم در متدی با عنوان ( ) nl2br پیاده‌سازی شده است. برای روشن شدن مطلب به قطعه کد زیر که از تابع مذکور جهت تبدیل مورد بحث استفاده می‌کند توجه نمایید:

```
$string = "one line \ n" ;
```

```
$string . = "another line \ n" ;
$string . = "a third for luck \ n" ;
print nl2br ($string) ;
```

چنانکه در این قطعه کد مشاهده می‌کنید، ابتدا یک دنباله کاراکتری طولانی که شامل سه عدد علامت خط جدید است در متغیری با نام $string ذخیره شده و سپس با بهره‌گیری از تابع ( ) nl2br سعی شده است تا علایم فوق به علامت < br > که بیانگر خط جدیدی در اسناد HTML است، تبدیل شود. خروجی حاصل از این قطعه برنامه به قرار زیر است:

```
one line < br / >
another line < br / >
a third for luck < br / >
```

توجه کنید که علامت < br > در اسناد سازگار با XHTML معادل علامت خط جدید در فایل‌های متن ساده است اما در خروجی حاصل از این قطعه برنامه که با استفاده از تابع ( ) print به نمایش درآمده است، به شکل < br / > ظاهر شده است. تابع ( ) nl2br در PHP4.0.5 و نسخه‌های بالاتر قابل استفاده است.

ساختار فراخوانی تابع ( ) nl2br بسیار ساده بوده وتنها به دنباله کاراکتری موردنظر که شامل علایم خط جدید است، نیاز دارد. این تابع جهت تبدیل علایمی از نوع خط جدید که در حال حاضر در درون دنباله کاراکتری موردنظر موجود می‌باشند، ابزار کارآمدی تلقی می‌شود. با این همه گاهی لازم است تا به‌منظور قالب‌بندی ستونی یک متن از علامت فوق به تعداد موردنیاز استفاده شود. تابع ( ) wordwrap جهت انجام این کار ابزار بسیار ایده‌آلی محسوب می‌شود. این تابع نیز مانند تابع قبل از دیدگاه فراخوانی دارای ساختاری بسیار ساده است. تنها آرگومان این تابع دنباله‌ای کاراکتری است که با بهره‌گیری از این تابع قصد قالب‌بندی آن‌را داریم. بنا به پیش‌فرض تابع( ) wordwrap متون دریافتی را در قالب خطوطی که تنها شامل ۷۵ کاراکتر است، قالب‌بندی خواهد کرد. تابع مورد بحث از کاراکتر ' \ n ' به‌عنوان کاراکتر (علامت) خط جدید بهره می‌گیرد. قطعه کد زیر از این تابع جهت قالب‌بندی دنباله کاراکتری موجود در متغیر $string استفاده می‌کند:

```
$string = "Given a long line, wordwarp ( ) is useful as ameans of" ;
$string . ="breaking it into a column and thereby making it" ; easier to read
print wordwrap ($string) ;
```

آنچه که در اثر اجرای کد فوق حاصل می‌شود، به‌صورت زیر خواهد بود:

```
Given a long line, wordwarp ( ) is useful as a means of breaking it into a cloumn and
thereby making it easier to read
```

به‌دلیل اینکه تابع ( ) wordwarp دنباله کاراکتری ورودی را با بهره‌گیری از علامت ' \ n ' قالب بندی می‌کند، تاثیر این قالب‌بندی در اسناد وب قابل مشاهده نمی‌باشد. قابلیت تابع ( ) wordwarp در قالب‌بندی متون به همین‌جا منتهی نشده بلکه قادر است تا با استفاده از دو آرگومان اختیاری دیگر فرآیند قالب‌بندی را به نحوه مطلوب‌تری توسعه دهد. اولین آرگومان اختیاری این تابع عدد صحیحی

است که حداکثر تعداد کاراکترهای قالب‌بندی شده را دریک خط از دنباله کاراکتری حاصل مشخص می‌کند. دومین آرگومان اختیاری تابع فوق یک دنباله کاراکتری است که نمایانگر مفهوم خاصی در متون است. این مفهوم خاص را با عنوان علامت انتهای خط می‌شناسیم. علامت فوق نشانه‌ای است که توسط آن انتهای خطوط موجود در متن مورد نظرمان را شناسایی می‌کنیم. در قطعه برنامه اخیر درصورتی‌که از عبارت زیر به‌جای آخرین عبارت موجود استفاده کنیم:

```
Print wordwrap ($string , 24 , "< br > \ n" ) ;
```

خروجی زیر را دریافت خواهیم کرد:

```
Give a long line, < br >
Wordwrap ( ) is useful as < br >
a means of breaking it < br >
into a column and < br >
thereby making it easier < br >
to read
```

توجه به این نکته مهم ضروری است که تابع ( ) wordwrap درصورتی‌که آخرین کلمه موجود در خط باعث متجاوز شدن از حد نهایی تعیین شده برای اندازه خط شود از شکستن خط مذکور در حالت عادی صرف‌نظر کرده و خط موردنظر را پس از آن کلمه خواهد شکست. با این همه این رفتار پیش‌فرض تابع ( ) wordwarp را می‌توان توسط آرگومان چهارم آن (یا به‌عبارت دیگر سومین آرگومان اختیاری) به گونه مطلوب تغییر داد. این آرگومان باید یک عدد صحیح مثبت باشد. با بهره‌گیری از تابع ( ) wordwarp به‌همراه این آرگومان اختیاری، می‌توانیم ترتیبی دهیم تا خطوط موجود در یک متن درست در نقطه تعیین شده توسط دومین آرگومان دچار شکست شوند و این امر کاملاً مستقل از این مطلب است که آخرین کلمه موجود در خطی از متن موردنظر باعث متجاوز شدن از حد نهایی تعیین شده برای اندازه خط مزبور شود یا خیر. برای روشن شدن این موضوع قطعه کد زیر را که از آرگومان چهارم تابع ( ) wordwarp نیز در فرآیندقالب‌بندی استفاده می‌کند، درنظر بگیرید:

```
$string = "As usual you will find me at http : // www. Wittering onaboutit.com" ;
$string  = "chat / eating _ green _ cheese / forum. php. " ;
$string  = "Hope to see you there ! " ;
print wordwarp ($string , 24 , "< br > \ n" , 1) ;
```

خروجی حاصل از این قطعه کد به شکل زیر است:

```
As usual you will find < br >
Me at < br >
http : // www. wittering onab < br >
outit. com / chat / eating _ gr < br >
een _ cheeses / forum . php . < br >
Hope to see you there !
```

تقسیم یک دنباله کاراکتری به عناصر یک آرایه با استفاده از تابع ( ) explode

تابع( ) explodeکه ظاهراً با انگیزه خصمانه‌ای نام‌گذاری شده است عملکردی مشابه تابع strtok ( ) که پیشتر آن‌را مورد بررسی قرار دادیم، دارد. همان‌گونه که تابع ( ) strtok دنباله‌های کاراکتری را به عناصر تشکیل دهنده شان تجزیه می‌کند این تابع نیز چنین فرآیندی را صورت می‌دهد؛ با این توضیح که تابع ( ) explode اجزای تشکیل‌دهنده دنباله کاراکتری موردنظر را در قالب عناصری از یک آرایه ذخیره می‌کند. این نوع عملکرد از آن جهت مفید است که می‌توان مقادیر ذخیره شده در این عناصر را به‌دلخواه‌مرتب‌سازی‌نمود یا پردازشهای دیگری را روی آنها صورت داد. ساختار این تابع از نظر فراخوانی نسبتاً ساده است به‌گونه‌ای که تنها به دو آرگومان جهت انجام عملیات پیش‌بینی شده نیاز دارد. اولین آرگومان این تابع یک دنباله کاراکتری است که مبنای فرآیند تجزیه دنباله کاراکتری اصلی به اجزای سازنده را مشخص می‌کند. به عبارت ساده‌تر این آرگومان تعیین کننده مرزهای اجزای تشکیل‌دهنده دنباله کاراکتری اصلی را مشخص می‌کند. دومین آرگومان تابع مورد بحث نماینده دنباله کاراکتری اصلی است. تابع ( ) explode آرگومان سومی را نیز به‌عنوان آرگومان اختیاری دریافت می‌کند. با بهره‌گیری از این آرگومان اختیاری که یک عدد صحیح مثبت است، تابع مذکور متوجه می‌شود که دنباله کاراکتری ورودی را حداکثر به چند جز تشکیل دهنده مجاز است تجزیه نماید. مشابه تابع ( ) strtok در مورد تابع ( ) explode نیز دنباله کاراکتری که مبنای جداسازی را مشخص می‌کند (آرگومان اول) می‌تواند شامل بیش از یک کاراکتر باشد. با این حال تفاوتی که در این مورد بین این دو تابع مشهود است، این است که مبنای جداسازی در تابع ( ) explode بدون توجه به اینکه حاوی یک یا بیش از یک کاراکتر است یک عامل جداکننده منفرد محسوب می‌شود، حال آنکه در تابع strtok ( ) هریک از کاراکترهای تشکیل دهنده دنباله‌های کاراکتری مورد بحث عاملی مستقل و مجزا می‌باشند. به قطعه کد زیر که در این رابطه تهیه شده است، توجه کنید.

در این قطعه با بهره‌گیری از تابع ( ) explode عوامل تشکیل دهنده یک تاریخ مشخص از یکدیگر جدا شده و در قالب عناصر آرایه‌ای با نام $data _ array ذخیره شده‌اند:

```
$start _ date = "2000 - 01- 12" ;
$date _ array = explode ( " _ " , $start _ date) ;
// $date [ 0 ] = "2000"
// $date [ 1 ] = "01"
// $date [ 2 ] = "12"
```

جمع‌بندی

همان‌گونه که در این ساعت متوجه شدید دنباله‌های کاراکتری یکی از اصلی‌ترین ابزارهای زبان برنامه‌نویسی PHP جهت ارتباط با دنیای خارج از برنامه‌ها و نیز ذخیره اطلاعات برای استفاده‌های

آینده محسوب می‌شوند. در درس این ساعت توابعی را که امکانات بسیار با ارزشی را جهت کنترل دنباله‌های کاراکتری چه از نظر قالب‌بندی نمایشی و چه از نظر دستکاری اساسی آنها در اختیارمان قرار می‌دهند، مورد بحث و بررسی قرار دادیم.

در درس این ساعت با نحوه قالب‌بندی دنباله‌های کاراکتری با استفاده از دو تابع ( ) printf و ( ) sprintf آشنا شدید. به‌واسطه این آشنایی اکنون باید بتوانید از هر دو تابع فوق جهت ایجاد دنباله‌های کاراکتری در قالبهای موردنیازتان بهره بگیرید. به واسطه این آشنایی اکنون باید بتوانید از هردو تابع فوق جهت ایجاد دنباله‌های کاراکتری در قالبهای مورد نیازتان بهره بگیرید. همچنین درمورد توابعی از PHP که اطلاعات مفیدی درباره دنباله‌های کاراکتری در اختیارمان قرار می‌دهند، مطالب مفیدی را فراگرفتید. به این ترتیب باید بتوانید با استفاده از تابع ( ) strlen وجود یک دنباله کاراکتری در درون دنباله کاراکتری دیگر، را تشخیص داده و با بهره‌گیری از تابع ( ) substr بخشی از یک دنباله کاراکتری را ایجاد نمایید. علاوه بر این باید قادر باشید تا با بهره‌گرفتن از تابع ( ) strtok یک دنباله کاراکتری را به اجزای تشکیل‌دهنده تجزیه نمایید.

بخش آخر از این درس به بررسی توابعی اختصاص داشت که امکان تبدیل دنباله‌های کاراکتری رادراختیارمان قرار می‌دهند. چنانکه در خلال درس فراگرفتید با بهره‌برداری از توابع ( ) trim، ( ) rtrim و ( ) ltrim می‌توانیم به‌راحتی فضاهای خالی ناخواسته را از ابتدا، انتها و یا از هردو ابتدا و انتهای یک دنباله کاراکتری حذف کنیم. همچنین با استفاده از تابع ( ) strtoupper، ( ) strtolower و ( ) ucwords می‌توانیم اندازه حروف بزرگ و کوچک موجود در یک دنباله کاراکتری را به یکدیگر تبدیل کنیم. در بخش‌انتهایی‌درس‌نیزمتوجه شدید که با بهره‌گیری از تابع ( ) str _ replace می‌توانیم تمامی نمونه‌های یک دنباله کاراکتری موجود در یک دنباله کاراکتری بزرگ‌تر را با یک دنباله کاراکتری دیگر تعویض کنیم. همچنین اشاره کردیم که قابلیتهای این تابع در نسخه‌های جدید PHP بهبود یافته و توسعه پیدا کرده‌اند.

کار شما با دنباله‌های کاراکتری به همین‌جا ختم نمی‌شود. در زبان برنامه‌نویسی PHP مبحثی وجود دارد که به عبارات منظم شهرت پیدا کرده است (این مبحث به زبان PHP محدود نمی‌شود). در رابطه با عبارات منظم، ویژگی‌ها و قابلیتهای جالب توجه‌تری از دنباله‌های کاراکتری را که از طریق مختلف در این ساعت بررسی کردیم در درس ساعت آینده مورد بحث قرار خواهیم داد. همان‌گونه که به‌زودی خواهید دید عبارات منظم ما را به ابزارهای قدرتمندی جهت برنامه نویسی مجهز خواهند کرد.

پرسش و پاسخ

پرسش: آیا توابع مفید دیگری در رابطه با دنباله‌های کاراکتری غیر از آنچه که در این درس به آنها اشاره شد موجود می‌باشد؟

پاسخ: بله. انتشار PHP4 حدود ۶۰ تابع مختلف را جهت کار با دنباله‌های کاراکتری از جنبه‌های مختلف به‌همراه آورد. جهت کسب اطلاع کامل در مورد این توابع می‌توانید به مستندات PHP در آدرس زیر مراجعه نمایید:

http : // www. php. net / manual / ref. Strings. php

پرسش: در برنامه نمونه‌ای در این درس که درمورد تابع ( ) printf عنوان شد، چگونگی استفاده از علامت < pre > و دلیل استفاده از آن‌را شرح دادیم. آیا روش فوق،یعنی استفاده از علامت < pre > بهترین روش ممکن جهت نمایش متون ساده قالب‌بندی شده بر روی صفحه مرورگر اینترنت محسوب می‌شود؟

پاسخ: چنانکه در این ساعت عنوان شد بهره‌گیری از علامت < per > درمواقعی که بخواهیم قالب متون ساده را تحت یک سند HTML نمایش دهیم، بسیار مفید است. با این همه اگر مایل باشیم که کل متن مورد نظرمان را بر روی پنجره مرورگر اینترنت نمایش دهیم، بهتر است ترتیبی دهیم تا مرورگر اینترنت کل خروجی را به‌صورت متن ساده قالب‌بندی نماید. انجام این فرآیند با بهره‌گیری از تابع ( ) header به شکل زیر قابل پیاده‌سازی است:

```
header ("Content _ Type : Text / Plain") ;
```

تمرینها

هدف از این بخش ارایه تمرینهای مختلف در قالب آزمون است. پاسخ بخش آزمون بلافاصله پس از آن آمده است. بخش فعالیتها شامل تمرینهایی است که به منظور افزایش قابلیتها و مهارتهای برنامه‌نویسی خواننده طراحی شده که البته فاقد پاسخ لازم است.

آزمون

۱- از کدام مشخصه تبدیل می‌توان جهت قالب‌بندی یک عدد صحیح به‌صورت اعشاری در تابع ( ) printf استفاده کرد؟ عبارت مورد نیاز جهت قالب‌بندی عدد صحیح 33 به‌عنوان یک عدد اعشاری را بنویسید؟

۲- با بهره‌گیری از مشخصه padding ترتیبی دهید تا نتیجه حاصل از اجرای تمرین قبل که اکنون یک عدد اعشاری است به‌گونه‌ای که بخش صحیح آن طولی برابر با چهار کاراکتر داشته باشد، قالب بندی شود؟

۳- در رابطه با تمرین قبلی با بهره‌گیری از عملگر دقت ترتیبی دهید تا بخش اعشاری از عدد مورد نظر با دقتی برابر با دو رقم بعد از اعشار در خروجی به نمایش درآید.

۴- از کدام تابع می‌توان جهت پی بردن به طول یک دنباله کاراکتری استفاده کرد؟

۵- از کدام تابع می‌توان جهت پی بردن به موقعیت اولین کاراکتر از یک دنباله کاراکتری کوچک‌تر که الگوی آن در دست است در درون یک دنباله کاراکتری بزرگ‌تر استفاده کرد؟

۶- از کدام تابع می‌توان جهت استخراج یک دنباله کاراکتری از درون دنباله کاراکتری دیگر استفاده نمود؟

۷- چگونه می‌توان فضاهای خالی ناخواسته را از ابتدای یک دنباله کاراکتری حذف کرد؟

۸- چگونه می‌توان ترتیبی داد که کلیه کاراکترهای یک دنباله کاراکتری با استفاده از حروف بزرگ نمایش یابند؟

۹- چگونه می‌توان اجزای تشکیل‌دهنده یک دنباله کاراکتری را در یک آرایه ذخیره نمود؟

پاسخ آزمون

۱- مشخصه تبدیل f قادر است تا یک عدد صحیح را به صورت یک عدد اعشاری قالب‌بندی کند:

```
Printf ("% f ", 33) ;
```

۲- مشخصه padding به ما امکان می‌دهد تا اندازه یک خروجی را بر مبنای تعداد کاراکترهایی که باید اشغال نماید، تعیین نماییم. این مشخصه به سادگی عبارت است از یک فضای خالی یا رقم صفر که به‌دنبال آن یک عدد صحیح تعداد خانه‌هایی را که باید توسط این کاراکتر (درصورت فقدان خروجی در آن موقعیت‌ها) پر شوند، مشخص می‌نماید:

```
printf ( "% 04f ", 33) ;
```

۳- مشخصه دقت به سادگی یک علامت نقطه است که به‌دنبال آن عدد صحیحی که نماینده تعداد ارقام بعد از اعشار است، واقع می‌شود. این مشخصه لزوماً باید پیش از مشخصه تبدیل واقع شود:

```
Printf ("% 04.2f ", 33) ;
```

۴- تابع ( ) strlen طول یک دنباله کاراکتری را باز می‌گرداند.

۵- تابع ( ) strstr موقعیت شروع یک دنباله کاراکتری در درون دنباله کاراکتری دیگر را به دست می‌دهد.

۶- تابع ( ) substr بخشی از یک دنباله کاراکتری را استخراج کرده و آن را به برنامه فراخواننده باز می‌گرداند.

۷- تابع ( ) ltrim فضاهای خالی ناخواسته را از ابتدای یک دنباله کاراکتری حذف می‌کند.

۸- تابع ( ) strtoupper تمامی کاراکترهای موجود در یک دنباله کاراکتری را به حروف بزرگ تبدیل می‌کند.

۹- تابع ( ) explode دنباله‌های کاراکتری ورودی را به اجزای تشکیل دهنده تجزیه کرده و هر جزء را در قالب عنصری از یک آرایه ذخیره می‌کند.

فعالیتها

۱- یک فرم HTML نظرسنجی طراحی کنید که نام و نام خانوادگی و آدرس پست الکترونیکی کاربر را دریافت کند. با بهره‌گیری از توابع تبدیل فراگیری شده در این درس ترتیبی دهید تا اولین حرف از نام و نام خانوادگی کاربر به حرف بزرگ تبدیل شده و دنباله‌های کاراکتری حاصل بر روی صفحه مرورگر اینترنت کاربر نمایش پیدا کنند. همچنین از وجود علامت @ در آدرس پست الکترونیکی کاربر اطلاع حاصل کرده و در صورت عدم وجود آن پیغام مناسبی را جهت اطلاع وی از درج اطلاعات نادرست بر روی صفحه نمایش دهید.

۲- آرایه‌ای شامل اعداد صحیح و اعشاری ایجاد کنید. با بهره‌گیری از یک ساختار تکرار مناسب ترتیبی دهید تا اعداد اعشاری موجود در این آرایه دارای دقتی معادل دو رقم اعشار شوند. عناصر این آرایه را در نهایت به‌گونه‌ای نمایش دهید که از سمت راست تراز شده و فضایی برابر با بیست کاراکتر را اشغال نمایند.

ساعت هجدهم

بهره‌گیری از عبارات منظم

عبارات منظم (Regular Expressions) روش بسیار توانمندی جهت ارزیابی و اعمال تغییرات موردنظر بر روی متون می‌باشند. با بهره‌گیری از آنها می‌توانیم الگوهای مختلف پیچیده را در درون دنباله‌های کاراکتری مورد جستجو قرارداده و بخشهای منطبق با این الگوها را با دقت و انعطاف بسیار از درون آنها استخراج نماییم. در همین ابتدای امر لازم است تا توضیحی را در این رابطه گوشزد نماییم؛ به دلیل اینکه عبارات منظم از توان بالایی جهت انجام عملیات پیش‌بینی شده برخوردارند می‌توان چنین نتیجه‌گیری کرد که در پیاده‌سازی آنها از ساختارها و منطق پیچیده‌ای استفاده شده است و همین امر باعث می‌شود که نسبت به توابع ابتدایی که در درس ساعت قبل با عنوان " بهره‌گیری از دنباله‌های کاراکتری" توضیح داده شدند، از سرعت اجرایی پایین‌تری برخوردار باشند. از این‌رو درصورتی‌که به قابلیتهای آرایه شده توسط عبارات منظم نیاز ندارید، توصیه می‌کنیم که جهت انجام کارهای ساده از توابع مذکور استفاده کنید.

زبان برنامه‌نویسی PHP دوگونه مختلف از عبارات منظم را مورد پشتیبانی قرار می‌دهد. گونه اول شامل مجموعه‌ای از توابع است که عبارات منظم به‌شیوه‌ای که در زبان برنامه‌نویسی Perl متداول است را پشتیبانی می‌کنند. گونه دوم که از جهات مختلف عملکردهای محدودتری را نسبت به شیوه موجود در زبان Perl مورد پشتیبانی قرار می‌دهد شامل توابعی است که بر مبنای استانداردهای تدوین شده در POSIX توسعه یافته‌اند. در درس این ساعت هر دو گونه موجود را به‌قدر کافی مورد بحث و بررسی قرار خواهیم داد.

آنچه که در این ساعت فرامی‌گیرید، شامل موارد زیر خواهد بود:

- **چگونگی تطبیق الگوها در دنباله‌های کاراکتری با استفاده از عبارات منظم**
- **دستورالعملهای ایجاد عبارات منظم**

- **چگونگی جایگزین کردن متن موجود در یک دنباله کاراکتری با استفاده از عبارات منظم**
- **چگونگی بهره‌گیری از قابلیتهای بسیار توانمند موجود در عبارات منظم زبان برنامه‌نویسی Perl جهت تطبیق و جای‌گزینی الگوهای یافت شده در یک متن**

در ادامه به بحث در مورد هریک از مطالب فوق خواهیم پرداخت.

توابع ارائه شده سازگار با استاندارد POSIX جهت استفاده از عبارات منظم

بهره‌گیری از توابع ویژه‌ای که در بخش عبارات منظم از مجموعه استانداردهای POSIX ارائه شده است امکانات مناسب و مطلوبی را جهت تطبیق و جای‌گزینی الگوهای پیچیده در متون را در اختیار برنامه‌نویسان قرار می‌دهد. این توابع معمولاً به‌سادگی با همین عنوان، یعنی توابع عبارات منظم یا Regular Expression Functions مورد اشاره قرار می‌گیرند اما در این درس آنها را با عنوان توابع عبارات منظم سازگار با مجموعه استاندارد POSIX خواهیم شناخت. این‌گونه نام‌گذاری از آن‌جهت است که مابین این دسته از توابع و توابع عبارات منظم سازگار با زبان برنامه‌نویسی Perl که از نوع اول توانمندتر و کارآمدتر است، تفاوت قائل شویم. دلیل دیگر این است که توابع مذکور بر مبنای استانداردهای POSIX جهت ایجاد عبارات منظم توسعه پیدا کرده‌اند.

عبارت منظم ترکیبی است از علایمی که می‌توان از آن جهت تطبیق یک الگوی خاص در یک متن استفاده نمود. بنابراین فراگیری عبارات منظم و چگونگی بهره‌گیری از قابلیتهای آن چیزی بیش از کسب اطلاع در مورد نوع آرگومان‌های دریافتی و نوع داده مقادیر بازگشتی توابعی از PHP است که درمورد عبارات منظم طراحی شده‌اند. جهت آغاز کار ما با معرفی توابع فوق و چگونگی استفاده از آنها به‌منظور معرفی و شناخت هرچه بیشتر عبارات منظم شروع می‌کنیم.

استفاده از تابع ( ) ereg جهت تطبیق الگوها در یک دنباله کاراکتری

به احتمال قوی ساختار این تابع که در گام نخست آن‌را معرفی می‌کنیم به دلیل وجود تعداد آرگومانهای ورودی اندکی پیچیده به‌نظر خواهد رسید. تابع( ) ereg جهت انجام عملیات پیش‌بینی شده مستلزم دریافت سه آرگومان می‌باشد. آرگومان اول این تابع یک دنباله کاراکتری است که نماینده یک الگو می‌باشد. آرگومان دوم نیز یک دنباله کاراکتری است که فرآیند جستجو در درون آن و به‌منظور یافتن موارد قابل تطبیق با الگوی تعیین شده در اولین آرگومان صورت خواهد گرفت. آخرین آرگومان این تابع آرایه‌ای است که موارد یافت شده در دومین آرگومان طی فرآیند جستجو ذخیره و نگهداری خواهد شد. آنچه که تابع ( ) ereg به‌عنوان نتیجه عملیات به برنامه فراخواننده باز می‌گرداند عدد صحیحی است که نماینده تعداد کاراکترهای موجود در مورد یافت شده در فرآیند جستجو بر مبنای الگوی تعیین شده می‌باشد. البته این درصورتی است که مورد مذکور یافت شده باشد. چنانچه تابع مورد بحث موفق به یافتن موردی که با الگوی تعیین شده در آرگومان اول این تابع تطبیق کند نشود، مقدار false را به‌عنوان نتیجه عملیات به برنامه فراخواننده باز می‌گرداند. ارائه مثالی در زمینه

استفاده از این تابع در رفع نکات مبهم کمک شایانی می‌کند. قطعه کد زیر با بهره‌گیری از تابع ereg ( ) جستجویی را برای یافتن موارد تطبیقی در دنباله کاراکتری "Aardvark advocacy" بر مبنای الگوی "aa" ترتیب می‌دهد:

```
Print ereg ( "aa", "aardvark advocacy", $array) ;
Print "< br > $array [ 0 ] < br >" ;
// output :
// 2
// aa
```

چنانکه در این قطعه کد کوتاه مشاهده می‌کنید الگوی مورد جستجو به‌صورت "aa" تعریف شده است و به دلیل اینکه بخشی ازدنباله کاراکتری" aardvark advocacy" با این الگو منطبق است تابع ereg ( ) عدد صحیح 2 را به نشانه اینکه تعداد کاراکتری تطبیق داده شده با الگوی جستجو برابر با 2 است به برنامه فراخواننده باز می‌گرداند. در این فرآیند اولین عنصر آرایه $array با دنباله کاراکتری تطبیق داده شده با الگوی مورد استفاده مقداردهی شده و با بهره‌گیری از تابع print ( ) بر روی صفحه چاپ می‌شود. شاید در نگاه نخست ثبت مورد یافت شده در آرایه اندکی عجیب به‌نظر برسد، چراکه کاملاً واضح است که ما به دنبال الگوی جستجوی "aa" در دنباله کاراکتری موردنظر از تابع ereg ( ) استفاده‌کرده‌ایم. اما وضعیت همواره به این وضوح و سادگی نمی‌باشد. برای مثال ممکن است با بهره‌گیری از کاراکتر . در الگوی مورد جستجو بخشهای قابل تطبیق را از دنباله کاراکتری اصلی مورد بازیابی قرار دهیم. در قطعه کد زیر از الگوی جستجوی "d ." به همین منظور بهره‌گرفته‌ایم:

```
Print ereg ( "d .", "aardvark advocacy", $array) ;
Print "< br > $array [ 0 ] < br >" ;
// output :
// 2
//dv
```

الگوی جستجوی "d ." بدین معنی است که هدف ما یافتن هر دو کاراکتری است که اولین آنها حرف d باشد. بدین ترتیب علامت نقطه در این الگو جانشینی برای تمامی کاراکترهای موجود محسوب می‌گردد. به‌عبارت دیگر هیچ‌گونه اطلاعی در مورد اینکه کاراکتر دوم چه خواهد بود در دست نمی‌باشد. از این‌رو مقدار ذخیره شده در عنصر $array [0] یعنی اولین عنصر از آرایه $array جهت تشخیص آن بسیار مؤثر خواهد بود.

بهره‌گیری از شاخصهای کمیت جهت تطبیق یک کاراکتر خاص به تعداد بیش از یک مرتبه

هنگامی‌که جستجویی را جهت یافتن کاراکتر خاصی در یک دنباله کاراکتری ترتیب می‌دهید، می‌توانید با بهره‌گرفتن از شاخصهای ویژه‌ای با عنوان شاخصهای کمیت یا quantifier تعداد دفعاتی را

که کاراکتر موردنظر باید تکرار شده باشد تا با الگوی جستجو تطبیق پیداکند، مشخص نمایید. برای نمونه استفاده از شاخص کمیت + در الگویی مثل "a +" باعث خواهد شد تا تابع ( ) ereg جستجویی را جهت یافتن بخشی از دنباله کاراکتری اصلی که دست کم شامل یک کاراکتر ' a ' بوده و به‌دنبال آن تعداد صفر یا چند (بیش از صفر) کاراکتر ' a ' واقع شده باشد، ترتیب دهد. قطعه کد کوتاه زیر با بهره‌گرفتن از الگوی جستجوی "a +" تابع ( ) ereg را جهت جستجوی موارد تطبیقی در دنباله کاراکتری "aaaa" فراخوانی می‌کند:

```
if ( ereg ("a +", "aaaa", $array) )
    print $array [ 0 ] ;
// prints "aaaa";
```

همان‌گونه‌که مشاهده می‌کنید این عبارت منظم هرتعداد کاراکتر ' a ' را که در دنباله کاراکتری اصلی (آرگومان دوم تابع) پیدا کند در لیست موارد قابل تطبیق با الگوی جستجو قرار می‌دهد. جدول ۱- ۱۸، لیست کاملی از شاخصهای کمیت قابل استفاده در عبارات منظم را جهت تطبیق نشان می‌دهد.

جدول ۱- ۱۸ شاخصهای کمیت قابل استفاده در عبارات منظم

علامت شاخص	توضیح	مثال
*	این شاخص شامل نمونه‌هایی می‌شود که به تعداد هیچ یا بیشتر تکرار شده باشند،	a *
+	این شاخص شامل نمونه‌هایی می‌شود که به تعداد یک یا چندمرتبه تکرار شده باشند.	a +
?	این شاخص شامل نمونه‌هایی می‌شود که دقیقاً هیچ یا یک مرتبه تکرار شده باشند.	a ?
{n}	این شاخص شامل نمونه‌هایی می‌شود که دقیقاً n مرتبه تکرار شده باشند.	a {3}
{n ,}	این شاخص شامل نمونه‌هایی می‌شود که دست‌کم n مرتبه تکرار شده باشند.	a {3 ,}
{, n}	این شاخص شامل نمونه‌هایی می‌شود که حداکثر n مرتبه تکرار شده باشند.	a {, 2}
{n1 , n2}	این شاخص شامل نمونه‌هایی می‌شود که دست‌کم تعداد n1 مرتبه و حداکثر به تعداد n2 مرتبه تکرار شده باشند، (به عبارت دیگر نمونه‌هایی را شامل می‌شود که مابین این دو یا برابر با تعداد این دو عدد باشند.	a {1 , 3}

اعداد ذکر شده در بین جفت علامت { } در جدول فوق با عنوان مرزها یا حدود شاخصهای کمیت شناخته می‌شوند. با بهره‌گیری هوشمندانه از این حدود همواره می‌توانید تعداد دفعاتی را که کاراکترهای مورد نظرتان باید تکرار شوند تا بر الگوی جستجو منطبق گردند به‌طور دقیق تعیین نمایید.

واژه جدید حدود شاخص کمیت، عاملی است که تعداد دفعاتی که کاراکتری بخصوص یا طیفی از کاراکترها باید تکرار شده باشند تا بر الگوی جستجوی یک عبارت منظم منطبق شوند را مشخص می‌کند. همان‌گونه که در جدول ۱- ۱۸ نیز مشاهده نمودید می‌توانیم حدود بالا و پایین یک کاراکتر بخصوص از یک متن (یا به‌عبارت دیگر بیشترین و کمترین تعداد دفعات تکرار) را دردرون یک جفت علامت { } پس از ذکر نام کاراکتر موردنظر قرار دهیم. برای مثال شاخص کمیت زیر:

a {4 , 5}

نمونه‌هایی از دنباله‌های کاراکتری مورد بررسی را که تعداد کاراکترهای متوالی ' a ' در آن کمتر از ۴ و بیشتر از ۵ نباشد، جهت انتخاب مشخص خواهد کرد.

اجازه دهید تا در این قسمت جهت روشن‌تر شدن مفهوم شاخص کمیت و حدود آن مثالی را ارائه دهیم که از تابع ( ) ereg بهره می‌برد. فرض کنید باشگاهی به اعضای خود کدهای عضویتی را نسبت داده است. کدهای عضویت این باشگاه از قالب ویژه‌ای برخوردار است به‌گونه‌ای که با یک تا چهار حرف ' y ' آغاز شده و به‌دنبال آن تعدادی دلخواه از کاراکترهای الفبا عددی و در نهایت از عدد ۹۹ استفاده می‌شود. اکنون وظیفه ما به‌عنوان یک برنامه نویس این است که جستجویی را در میان تمامی کدهای عضویت این باشگاه در چند سال اخیر انجام داده و کدهای عضویتی که در این قالب جدید می‌گنجد را جهت ارائه به مدیریت باشگاه عرضه نماییم (فرض کنید که این شیوه کدگذاری به تازگی ابداع شده و با روشی که در گذشته استفاده می‌شد به‌کلی تفاوت دارد).

قطعه کد کوتاهی که در ادامه مشاهده می‌کنید شامل عبارت منظمی است که جستجوی موردنظر را انجام خواهد داد:

```
$test = "the code is yyXGD99 _ have you received my club?" ;
if (ereg ("y {1 , 4}. *99", $test, $array) )
    print "Found membership : $array [ 0]" ;
// prints "Found membership : yyXGDH99"
```

چنانکه در قطعه کد بالا مشاهده می‌کنید کد عضویت نمونه‌ای که جهت ارزیابی عملکرد عبارت منظم مورد استفاده طراحی شده است شامل دو کاراکتر متوالی ' y ' است که به‌دنبال آن از سه حرف بزرگ الفبا شامل X, G و D و در نهایت نیز از دو کاراکتر متوالی ' 9 ' استفاده شده است. واضح است که y {1 , 4} با دو کاراکتر متوالی ' y ' منطبق است.

همچنین سه کاراکتر از حروف بزرگ الفبا، یعنی XGD با * . تطبیق می‌کند. مورد اخیر درواقع با هر تعداد از کاراکترهای الفبا عددی متوالی کاملاً تطبیق می‌کند (حتی از این هم بیشتر با هر تعداد کاراکتر متوالی از هر نوعی که باشند، تطبیق می‌کند).

از این‌رو چنین به‌نظر می‌آید که فرآیند خواسته شده به‌خوبی انجام شده است. اما حقیقت چیز دیگری است چراکه اقدامات مهم دیگری نیز در این میان باقی‌مانده است. برای اطمینان از اینکه مورد تطبیقی با "99" خاتمه یافته است همان‌گونه که ملاحظه می‌کنید در الگوی تطبیق بعد از "99" از فضای خالی به میزان یک کاراکتر استفاده شده است. اما اشکال کار هنگامی آشکار می‌شود که متن موردنظر ما شامل دنباله کاراکتری خاصی مثل این باشد:

"my code is yyXGDH99 did you get my 1999 sub?

خبر بد اینکه عبارت منظم موجود در قطعه کد فوق مورد زیر را به‌عنوان یک مورد تطبیقی بازیابی خواهد کرد:

"y code is yyXGDH99 did you get my 1999"

واضح است که عبارت منظم کوچک ما اندکی اشکال دارد. چنانکه ملاحظه می‌کنید این عبارت منظم کاراکتر ' y ' را در اولین کلمه یعنی my منطبق بر شاخص کمیت {4 , 1} y فرض کرده و هر تعداد از هر نوع تا مشاهده "99" را منطبق بر شاخص کمیت * • فرض می‌کند. عبارات منظم، موجودات بسیار فعالی هستند به‌گونه‌ای که تا حد ممکن تمایل دارند تا کاراکترهای بیشتری را با الگوی جستجوی موجود تطبیق دهند. به همین جهت نیز در مثال اخیر عبارت منظم مورد استفاده ما به‌جای اینکه فرآیند تطبیق را در بخش "99" از کد عضویت متوقف کند، آن‌را تا بخش مذکور از 1999 ادامه داده است. اگر تنها می‌توانستیم به روشی مطمئن شویم که کاراکترهای موجود مابین y و 99 در الگوی جستجو از نوع کاراکترهای الفبا عددی (کاراکترهای A تا Z، a تا z و 0 تا 9) هستند، ممکن بود بتوانیم گامی در جهت رفع این مشکل برداریم. در حقیقت با بهره‌گیری از مفهوم ویژه‌ای با عنوان " کلاس کاراکتر" قادریم تا مجموعه کاراکترها را محدود نماییم.

تطبیق محدوده خاصی از کاراکترها با بهره‌گیری از کلاسهای کاراکتر

همان‌گونه که احتمالاً متوجه شده‌اید تا به حال یا تنها کاراکترهای ویژه و یا تمامی کاراکترهای موجود از هر نوع را در فرآیند جستجو مورد استفاده قرار دادیم. با این حال چنین وضعیتی کاربردهای محدود داشته و در اغلب موارد لازم است تا کاراکترهای موردنظرمان را در قالب محدوده‌ای خاص از کاراکترها (زیرمجموعه‌ای از کل کاراکترهای موجود) انجام دهیم. برای تعریف یک کلاس کاراکتر کافی است تا کاراکترهای مورد نظرتان را با بهره‌گرفتن از جفت علامت [ ] گروه‌بندی نمایید. بدین ترتیب کلاس کاراکتر [a , b] تنها منطبق بر دو کاراکتر ' a ' و ' b ' خواهد بود. پس از تعریف یک چنین کلاسی می‌توانید آن‌را یک کاراکتر تنها فرض کرده و با آن رفتاری مشابه یک کاراکتر تنها داشته باشید. از این‌رو الگوی + [ab] با هر سه مورد "aaa"، "bbb" و "ababab" منطبق خواهد بود.

علاوه بر این می‌توانید محدوده‌ای از کاراکترهای موجود را با استفاده از یک کلاس کاراکتر محک بزنید. از این‌رو کلاس کاراکتر [a - z] با هر کاراکتر الفبایی که از نوع حروف کوچک باشد، تطبیق

خواهدکرد.همچنین‌کلاس کاراکتر [A- Z] نیز با هر کاراکتر الفبایی که از نوع حروف بزرگ باشد، تطبیق می‌کند. و به‌طور مشابه واضح است که کلاس کاراکتر [9 - 0] با هر کاراکتر عددی تطبیق می‌نماید. امکانات کلاس کاراکتر به همین‌جا ختم نشده بلکه درصورت تمایل می‌توانیم محدوده‌های کاراکتری را با کاراکترهای مجزا ترکیب کرده و کلاسهای جدیدی از کاراکترها ایجاد نماییم. برای نمونه کلاس کاراکتر [a - z5] با هر کاراکتر الفبایی از نوع حروف کوچک یا عدد 5 تطبیق می‌کند.

قابلیتهای کلاس کاراکتر تمامی ندارد. با بهره‌گیری از کاراکتری از صفحه کلید با عنوان caret که با علامت ^ مشخص شده است، می‌توانید فرآیند تطبیق را در جهت عکس کاراکترهای ذکر شده در کلاس انجام دهید. برای مثال کلاس کاراکتر [^ A - Z] با هرنوع کاراکتری غیر از کاراکترهای الفبایی حروف بزرگ تطبیق می‌کند.

اکنون که به ابزار جدیدی مجهز شده‌ایم اجازه دهید تا به مبحث مطرح شده در قسمت قبل و مثالی که در آن قسمت طرح کردیم، بازگردیم. چنانکه در این مثال مشاهده نمودید، جهت تطبیق لازم است تا کاراکتر ' y ' بین یک تا چهار مرتبه تکرار شده و به دنبال آن به هر تعداد دلخواهی از کاراکترهای الفبا عددی (حروف بزرگ و کوچک یا ارقام از صفر تا ۹) تکرار شده باشند و درنهایت نیز رقم ۹ دقیقاً دو بار تکرار شود. با بهره‌گیری از کلاس کاراکتر که در این قسمت فراگرفتید، می‌توانیم عبارت منظم را به‌گونه‌ای مطلوب به‌صورت زیر اصلاح کنیم:

```
$test = "my code is yyXGDH99 did you get my 1999 sub?" ;
if (ereg (" y {1 , 4} [a- zA- Z0- 9]* 99" , $test, $array) )
   print "Found membership : $array [0]" ;
// print "Found membership : yyXGDH99"
```

همان‌گونه که مشاهده می‌کنید به آنچه که موردنظرمان بود نزدیک شده‌ایم. کلاس کاراکتری که در این عبارت منظم استفاده کرده‌ایم اکنون مانع از آن می‌شود که کاراکترهای غیر الفبا عددی در فرآیند تطبیق شرکت داده شوند و بدین ترتیب کد عضویت موردنظرمان را بازیابی کرده‌ایم. با این حال هنوز عبارت منظم مورد استفاده ما نقطه ضعف دیگری نیز دارد که می‌تواند در فرآیند تطبیق ما را اندکی دچار سردرگمی نماید. برای اثبات موضوع کافی است تا بلافاصله بعد از کد عضویت موردنظر در دنباله کاراکتر اصلی از یک علامت کاما استفاده نمایید تا مشکل فوق خود را آشکار کند. برای این منظور قطعه کد فوق را به صورت زیر مجدداً مورد بازنویسی قرار می‌دهیم:

```
$test = "my code is yyXGDH99, did you get my 1999 sub? " ;
if (ereg ("y {1 , 4} [a- zA- Z0- 9]* 99" , $test, $array) )
   print "Found membership : $array [0]";
// regular expression fails
```

این ناکامی بدان دلیل است که ما جهت اطمینان از اینکه انتهای کد عضویت حاصل شده‌است از یک فضای خالی به‌اندازه یک کاراکتر در انتهای عبارت منظم خود بهره‌گرفته‌ایم (به‌عبارت دیگر انتهای کد موردنظر را با یک فضای خالی نشان کرده‌ایم). از این جهت درصورتی‌که کد عضویتی که از

ساختار قابل تطبیقی نیز برخوردار است در درون پرانتز واقع شده و یا پس از آن از علامتی نظیر خط فاصله یا کاما استفاده شده باشد، مورد بازیابی قرار نخواهد گرفت. برای تصحیح این خطای کوچک بازهم به سراغ کلاس کاراکتر می‌رویم. این بار از کلاسی استفاده می‌کنیم که دقیقاً عکس کلاس استفاده شده در مرحله قبل است. به بیان دیگر کاراکترهای قابل تطبیق ما پس از کد عضویت دقیقاً کاراکترهای غیر الفبا عددی هستند. با بهره‌گیری از کلاس کاراکتری [^ a -zA- Z0- 9] آنچه که مورد نظر ماست در این مثال کاملاً تأمین می‌شود:

```
$test = "my code is yyXGDH99, did you get my 1999 sub? " ;
if (ereg ("y {1 , 4} [a- zA- Z0- 9]* 99 [^ a- zA- Z0- 9]" , $test, $array) )
   print "Found membership : $array [0]";
// prints "Found membership : yyXGDH99, "
```

چنانکه مشاهده می‌کنید ما به جواب مسأله نزدیک و نزدیک‌تر می‌شویم، اما هنوز تا آنچه که مدنظرمان است، فاصله‌ای وجود دارد. حقیقت این است که عبارت منظم مورد استفاده ما هنوز درگیر دو مشکل کوچک است. اول آنکه همان‌گونه که در خروجی حاصل از قطعه کد بالا مشاهده می‌کنید علامت کاما به‌گونه‌ای ناخواسته در خروجی بازیابی شده است و دوم آنکه چنانچه کد عضویت در انتهای دنباله کاراکتری اصل باشد عبارت منظم در تطبیق آن با ناکامی مواجه می‌شود چراکه مطابق آنچه که در عبارت منظم ملاحظه می‌کنید، وجود یک کاراکتر پس از کد عضویت جهت تطبیق آن در دنباله کاراکتری اصلی موردنیاز است (این ضرورت به واسطه استفاده از کلاس کاراکتر [^ a- zA- Z0- 9] به‌وجود آمده است). به‌عبارت دیگر، لازم است تا راهی قابل اطمینان جهت بررسی حدود کلمات در دنباله کاراکتری اصلی پیدا کنیم. پس از یافتن چنین راهی مجدداً جهت اصلاح عبارت منظم مسأله فوق را مورد بررسی قرار خواهیم داد. قسمت بعد به بررسی این روش اختصاص دارد.

مفهوم اتم و بهره‌گیری از آن در عبارات منظم

واژه جدید واژه اتم (atom) به الگویی اطلاق می‌شود که دردرون یک پرانتز محصور شده باشد. گاهی جهت اشاره به اتم از اصطلاح الگوی فرعی یا زیرالگو (subpattern) نیز استفاده می‌شود. پس از تعریف اتم موردنظر، می‌توان آن‌را به‌عنوان یک کاراکتر تنها یا یک کلاس کاراکتر فرض کرده و رفتاری مشابه این دو را در مورد اتم موردنظر خود اعمال کرد. به بیان دیگر، می‌توان یک الگوی ثابت را بارها و بارها در یک دنباله کاراکتری با استفاده از مشخصه‌های کمیت موجود در جدول ۱- ۱۸ مورد تطبیق قرار داد.

در قطعه برنامه کوتاهی که در ادامه مشاهده می‌کنید ابتدا الگویی را تعریف کرده و آن‌را در داخل یک جفت پرانتز قرار داده‌ایم (بدین ترتیب یک الگوی فرعی یا اتم ایجاد کرده‌ایم). سپس با بهره‌گیری از یک مشخصه کمیت مناسب بر این نکته تأکید کرده‌ایم که جهت موفقیت عبارت منظم لازم است تا اتم مورد بحث به تعداد دو بار مورد تطبیق قرار بگیرد:

```
$test = "abbaxabbaxabbax" ;
if (ereg (" ( [ ab ] + X) { 2 }" , $test , $array) )
   print "$array [0]" ;
// prints "abbaxabbax"
```

دقت کنید که الگوی [ab] + x با دنباله کاراکتری "abbax" تطبیق می‌کند. این در حالی‌است که الگوی ([ab] + X) {2} با دنباله کاراکتری "abbaxabbax" تطبیق خواهد کرد.

در قطعه کد بالا اولین عنصر از آرایه‌ای که به‌عنوان آخرین آرگومان به تابع ( ) ereg ارسال می‌شود، شامل دنباله‌ای از کاراکترها خواهد بود که به‌طور کامل با الگوی مورد استفاده تطبیق می‌کند. عناصر بعدی موجود در این آرایه هریک شامل بخشهایی از دنباله کاراکتری مذکور که با اتمهای موجود در عبارت منظم تطبیق می‌کنند، خواهند بود. این بدان مفهوم است که علاوه بر کل دنباله کاراکتری تطبیق داده شده می‌توانیم بخشهای مختلف آن‌را نیز که با اتم‌های مختلف عبارت منظم منطبق هستند، مورد دستیابی قرار دهیم. لازم به ذکر است که حداکثر تعداد الگوهای فرعی برابر با ۱۰ است.

در قطعه کد زیر با بهره‌گیری از این موضوع سعی کرده‌ایم تا آدرس IP خاصی را مورد جستجو قرار داده و علاوه بر دستیابی به کل آدرس بازیابی شده به هریک از بخشهای مجزای آن دست پیدا کنیم (آدرس IP از چهار بخش تشکیل می‌شود که این بخشها با استفاده از علامت نقطه از یکدیگر جدا می‌شوند):

```
$test = "158. 152. 55. 35" ;
if (ereg ( "( [0 - 9] +) \. ( [0 - 9] +) \. ( [0 - 9] +) \. ( [0 - 9] +)" , $test , $array) ) {
   foreach ($array as $val)
     print "$val < BR >" ;
}
// output :
// 158. 152. 1. 58
// 158
// 152
// 1
// 58
```

توجه کنید که ما در این قطعه کد از علامت \ پیش از هر علامت نقطه‌ای که در عبارت منظم آمده است، استفاده کرده‌ایم. به این ترتیب بر این نکته تاکید کرده‌ایم که PHP از مفهوم علامت نقطه در عبارات منظم (که به‌عنوان مشخصه کمیت مورد استفاده قرار می‌گیرد) صرف‌نظر کرده و مفهوم کلی آن موردنظر قرار بگیرد (به‌عنوان کاراکتر ساده .) . انجام چنین کاری در مورد هر کاراکتری که مفهوم و معنای خاصی در الگوهای موجود در عبارات منظم دارند، ضروری است.

انشعاب

با بهره‌گیری از علامت pipe (|) در ترکیب الگوهای موجود در عبارات منظم می‌توانیم انشعاباتی را به دلخواه در عبارات منظم ایجاد نماییم. این علامت در الگوی مذکور عملکرد مشابهی با عملگر

"OR" در عبارات منطقی دارد. به بیان دیگر، عبارت منظمی که شامل دو انشعاب است می‌تواند با هریک از الگوهای موجود در عبارت مذکور (الگوی اول یا الگوی دوم) مورد تطبیق قرار گیرد. بدین ترتیب شاهد انعطاف بیشتری در مورد استفاده از عبارت منظم خواهیم بود. در قطعه کد زیر با بهره‌گیری‌از مفهوم انشعاب هر دو دنباله کاراکتری " "com . و "co. uk . "را مورد جستجو قرار داده‌ایم:

```
$test = "www. adomain . com" ;
if (ereg (" \ . com | \ . co . uk" , $test , $array) )
   print "it is a $array [0] domain < BR >" ;
// print "it is . com domain "
```

تعیین موقعیت تطبیق

علاوه‌بر اینکه می‌توانیم از الگوهای موجود در عبارات منظم برای تطبیق (جستجوی) دنباله‌های کاراکتری موردنظرمان استفاده کنیم؛ این امکان وجود دارد تا موقعیتی را که انتظار داریم فرآیند تطبیق را انجام دهیم، مشخص نماییم. برای نمونه می‌توانیم با بهره‌گیری از کاراکتر caret یا (^) پیش از الگوی موردنظر بر این مطلب تاکید کنیم که انتظار داریم تا موارد قابل تطبیق را در ابتدای دنباله کاراکتری اصلی پیدا کنیم. برای مثال الگوی a ^ در یک عبارت منظم با دنباله کاراکتری "apple" تطبیق کرده حال آنکه در مورد دنباله کاراکتری "bannana" چنین اتفاقی روی نمی‌دهد چراکه کاراکتر مذکور باعث می‌شود تا تنها کاراکترهای ' a ' که در ابتدای دنباله‌های کاراکتری واقع شده‌اند، با اهمیت فرض شوند.

به‌طور مشابه، می‌توانیم با بهره‌گرفتن از کاراکتر دلار($) در انتهای الگوی موردنظر بر این مطلب تاکید کنیم که انتظار داریم تا موارد قابل تطبیق را در انتهای دنباله کاراکتری اصلی پیدا کنیم. برای مثال الگوی a $ در یک عبارت منظم با دنباله کاراکتری "flea" تطبیق می‌کند حال آنکه در مورد دنباله‌کاراکتری"game"چنین اتفاقی روی نمی‌دهد زیرا کاراکتر مذکور باعث می‌شود تا تنها کاراکترهای ' a ' که در انتهای دنباله‌های کاراکتری واقع شده‌اند، با اهمیت فرض شوند.

بررسی مجدد مثال کد عضویت

همان‌گونه که شاهد هستید اکنون به ابزارهای بسیار کارآمد و توانمندی جهت حل و فصل مشکلات موجود در مثال کد عضویت مجهز شده‌ایم. باردیگر یادآوری می‌کنیم که در میان کدهای عضویت اعطایی به اعضای باشگاه فرضی به‌دنبال آن دسته از کدهایی هستیم که با یک تا چهار کاراکتر ' y ' آغاز شده و به‌دنبال آن شامل هر تعداد دلخواهی از کاراکتر الفبا عددی (حروف بزرگ و کوچک a تا z و A تا Z و ارقام صفر تا ۹) و در نهایت دو رقم ۹ پشت سرهم باشند. مشکل فعلی ما این است که به‌نحوی بتوانیم حد و مرز مواردی را که طی فرآیند تطبیق شناسایی کرده‌ایم، دقیقاً معین کنیم. چنانکه خاطرتان است در استفاده از فضای خالی و علایم مختلف نقطه‌گذاری در این راه ناموفق بودیم.

این بدان دلیل بود که ممکن است مورد یافت شده در ابتدا یا انتهای دنباله کاراکتری اصلی و یا در درون پرانتز واقع شود یا اینکه بعد از آن به‌سادگی از یک علامت خط فاصله استفاده شده باشد. مطالبی را که در قسمتهای قبل مورد بحث قرار دادیم، در این مورد به کمک ما می‌آیند.

اکنون که با چگونگی استفاده از انشعابات و نحوه تعیین موقعیت تطبیق (دو قسمت قبل در درس این ساعت) آشنا شدید، می‌توانیم ترتیبی دهیم تا مواردی که بعد از آنها از کاراکتری غیر الفبا عددی استفاده شده‌اند یا مواردی را که در انتهای دنباله کاراکتری اصلی ظاهر شده‌اند، تطبیق دهیم. همچنین می‌توانیم از روش مشابهی جهت تعیین حدود ابتدای مورد تطبیقی استفاده نماییم. علاوه بر این قادریم تا با بهره‌گیری مناسب از پرانتزها ترتیبی دهیم تا درصورتی که مورد تطبیقی مابین هرگونه کاراکتر نقطه‌گذاری یا فضای خالی محصور شده باشد، مورد بازیابی قرار بگیرد. در قطعه کدی که در ادامه ملاحظه می‌کنید کلیه این قابلیتها را با بهره‌گیری از امکانات بحث شده پیاده‌سازی کرده‌ایم:

```
$test = "my code is yyXGDH99, did you get my 1999 sub? " ;
if ( ereg ( "( ^ | [ ^ a - zA - Z0 - 9] ) (y {1 , 4}[ a - zA - Z0 - 9]* 99) ( [ ^ a - zA - Z0 - 9] |
      $)" , $test , $array) )
   print "Found membership : $array [2]" ;
// prints "Found membership : yyXGDH99"
```

همان‌گونه که در قطعه کد فوق مشاهده می‌کنید ظاهر عبارات منظم اندکی مبهم و ناشناخته به‌نظر می‌رسد اما با تقسیم آن به بخشهای کوچک‌تر معمولاً می‌توانیم به اسرار پنهان در هر یک از آن بخشها و بنابراین در کل عبارت پی ببریم. با استفاده از این عبارت منظم اکنون می‌توانیم اطمینان حاصل کنیم که در تشخیص موارد قابل تطبیق با الگوی مورد نظر به موفقیت دست پیدا خواهیم کرد. طراحی عبارت منظم به‌صورت فوق این اطمینان را می‌دهد که موارد قابل تطبیق آن مواردی هستند که (علاوه بر شرایط لازم، یعنی وجود یک تا چهار کاراکتر ' y ' در ابتدا و به دنباله آن تعداد دلخواهی از کاراکترهای الفبا عددی و در نهایت دو کاراکتر ' 9 ' پشت سر هم) یا در ابتدای دنباله کاراکتری اصلی واقعند و یا پیش از آنها یک کاراکتر غیر الفبا عددی واقع شده باشد. درمورد مرزهای انتهایی نیز این اطمینان به دست آمده که موارد قابل تطبیق یا در انتهای دنباله کاراکتری اصلی واقعند و یا پس از آنها یک کاراکتر غیر الفبا عددی واقع شده است. نکته دیگر آنکه همان‌گونه که مشاهده می‌کنید الگویی را که مایل به استخراج موارد تطبیق داده شده با آن هستیم در درون پرانتز قرار داده‌ایم. این از آن جهت است که مایل به ثبت کاراکترهای ناخواسته ابتدایی و انتهایی موارد یافت شده نمی‌باشیم. چنانکه شاهد هستید، با دسترسی به سومین عنصر از آرایه $array (عنصری با شاخص عددی 2) می‌توانیم موارد قابل تطبیق را مشاهده نماییم.

نکته مهمی که در رابطه با عملکرد تابع ( ) ereg باید به‌خاطر داشته باشید این است که تابع مذکور به بزرگی و کوچکی حروف بسیار حساس است. درصورتی که مایلید تا این رفتار پیش‌فرض تابع ( ) ereg را به‌گونه‌ای تغییر دهید که حساسیت فوق از میان برود، می‌توانید از تابع دیگری که عملکرد کاملاً مشابه و یکسانی را بدون اهمیت به بزرگی و کوچکی حروف در اختیارتان قرار می‌دهد، استفاده کنید. نام این تابع ( ) eregi است.

استفاده‌از تابع ( ) ereg _ replace جهت جایگزین کردن الگوها در دنباله‌های کاراکتری

تا بدین جای درس این ساعت توان خود را بر روی یافتن مواردی در یک دنباله کاراکتری که با الگوی خاصی تطبیق می‌کنند، صرف کردیم. به عبارت دیگر عبارات منظمی را که مورد بحث و بررسی قراردادیم هیچ‌گونه تغییری را بر روی دنباله کاراکتری اصلی اعمال نمی‌کردند. تابعی را که با عنوان ereg _ replace ( ) در این قسمت از درس مورد بحث قرار می‌دهیم، قادر است تا پس از یافتن مواردی از یک دنباله کاراکتری که با الگوی خاصی تطبیق می‌کنند آنها را با دنباله کاراکتری جدیدی تعویض نمایند (این فرآیند مشابه فرآیند find / replace است که سیستمهای عامل مختلف در رابطه با فایل‌های متن در اختیار کاربران خود قرار می‌دهند.

تابع ( ) ereg _ replace ساختار نسبتاً ساده‌ای دارد. این تابع جهت انجام عملیاتی که برای آن درنظر گرفته است مستلزم بهره‌گیری از سه آرگومان می‌باشد. آرگومان اول این تابع یک عبارت منظم است که جهت تطبیق الگو مورد استفاده قرار می‌گیرد. آرگومان دوم متن جایگزین را مشخص می‌کند. به‌عبارت دیگر متنی را مشخص می‌کند که در صورت یافتن مورد تطبیقی در دنباله کاراکتری اصلی باید جایگزین آن مورد گردد و بالاخره آرگومان سوم دنباله کاراکتری اصلی را که باید در نهایت طی فرآیند مورد بحث دستخوش تغییر شود، مشخص می‌نماید. آنچه که تابع موردنظر به عنوان نتیجه عملیات خود به برنامه فراخواننده باز می‌گرداند بستگی به موفقیت این تابع در یافتن (و جایگزین کردن) موارد قابل تطبیق دارد. چنانچه این تابع موفق به انجام این کار شود دنباله کاراکتری اصلی را که اکنون در اثر جایگزینی بخشهایی از آن با دنباله کاراکتری جایگزین دستخوش تغییر شده است به برنامه فراخواننده باز می‌گرداند. اما درصورتی که این تابع موفق به جایگزینی نشود (این وضعیت هنگامی رخ می‌دهد که هیچ بخش قابل تطبیقی با الگوی موجود وجود نداشته باشد)، دنباله کاراکتری اصلی را بدون کوچک‌ترین تغییری به برنامه فراخواننده باز خواهد گرداند. ارائه یک مثال وضعیت را روشن‌تر خواهد کرد. در قطعه کد کوتاهی که در ادامه مشاهده خواهید کرد سعی ما بر این است که دنباله‌ای از کاراکترها را جهت یافتن نام موردنظر و جایگزین کردن آن با نام جدید مورد جستجو قرار دهیم:

```
$test = "Our Secretary, sarah Williams is pleased to welcome you ." ;
print ereg _ replace (" Sarah Williams" , "Rev. P.W. Goodchild" , $test) ;
// prints "Our Secretary , Rev. P.W. Goodchild is pleased to Welcome
//          you . "
```

به عدم تشابه مابین دو تابع ereg ( ) و ereg _ replace ( ) توجه کنید. درحالی‌که تابع ereg ( ) تنها اولین مورد قابل تطبیق با الگو را انتخاب می‌کند، تابع ereg _ replace ( ) تمامی موارد قابل تطبیق با الگوی موجود را یافته و با دنباله کاراکتری جایگزین تعویض می‌کند.

بهره‌گیری از پس مرجعها (Back References) به‌همراه تابع ereg _ replace ( )

بااستفاده‌ازپس مرجعها می‌توانیم بخشی از یک الگوی تطبیقی را در دنباله کاراکتری جای‌گزین مورد بهره‌برداری قرار دهیم. جهت استفاده از این قابلیت لازم است تا هریک از عناصر تشکیل دهنده عبارت منظم موردنظرتان را که احتمال می‌دهید در فرآیند شرکت دهید در درون جفت پرانتز محصور نمایید. در این صورت می‌توانید با بهره‌گیری از دو علامت \ به‌صورت \\ و به دنباله شماره اتم موردنظر به‌ترتیبی که در الگوی تطبیقی ظاهر شده‌اند، متون قابل تطبیق با این الگوهای فرعی را در دسترس دنباله‌های کاراکتری جانشین قرار دهید (برای مثال 1 \\). دقت کنید که اتم‌ها به‌ترتیب شماره‌گذاری شده و ترتیب آنها از خارج به‌داخل و از چپ به راست می‌باشد (شماره‌گذاری از عدد 1 آغاز می‌شود). همچنین توجه کنید که 0 \\ نماینده کل آن چیزی خواهد بود که با الگوی تطبیقی منطبق است.

به مثال زیر در رابطه با این مفهوم توجه نمایید. در این قطعه کد سعی کرده‌ایم تا تاریخها را از قالب dd / mm / yy به قالب mm/ dd / yy تبدیل کنیم:

```
$test = "25 / 12 / 2000" ;
print ereg _ replace ( "( [0 - 9] +) / ( [0 - 9] +) / ( [0 - 9] +)" , "\\ 2 / \\ 1 / \\ 3" , $test) ;
// prints "12 / 25 / 2000"
```

> باردیگر توجه کنید که تابع(ereg _ replace بنا به پیش‌فرض نسبت به بزرگی و کوچکی حروف حساس است. اگر مایلید تا این رفتار پیش‌فرض تابع مذکور را به‌گونه‌ای تغییر دهید که حساسیت فوق از میان برود، می‌توانید از تابع دیگری که عملکرد کاملاً مشابه و یکسانی را بدون توجه به بزرگی و کوچکی حروف در اختیارمان قرار می‌دهد، استفاده کنیم. نام این تابع ereg _ replace ( ) است.

استفاده از تابع split ( ) جهت تقسیم دنباله‌های کاراکتری

همان‌گونه که در درس ساعت هفدهم با عنوان "بهره‌گیری از دنباله‌های کاراکتری" مشاهده کردید؛ با استفاده از تابعی با عنوان explode ( ) می‌توانیم اجزای تشکیل‌دهنده یک دنباله کاراکتری را برمبنای مشخصی تجزیه کرده و هربخش را به‌طور مجزا در قالب عنصری از یک آرایه ذخیره نماییم.

این تابع ابزار توانمندی در نوع خود محسوب می‌شود اما در عین حال شامل محدودیت آشکاری نیز می‌باشد به‌گونه‌ای که هنگام استفاده از این تابع به مجموعه منفردی از کاراکترها جهت تعیین عناصر جداکننده محدود هستیم. یعنی عواملی را که مبنای تجزیه دنباله کاراکتری به اجزای تشکیل دهنده آن هستند تنها از یک مجموعه محدود منفرد از کاراکترها می‌توانیم انتخاب کنیم. PHP جهت گریز از این محدودیت تابع دیگری را با عنوان ( ) split معرفی کرده است که با بهره‌گیری از این تابع می‌توان از توان بی‌منتهای عبارات منظم جهت تعیین مبنای تجزیه یک دنباله کاراکتری به اجزای تشکیل‌دهنده آن استفاده نمود. تابع ( ) split، ساختار نسبتاً ساده‌ای دارد. این تابع جهت انجام عملیات پیش‌بینی شده مستلزم دریافت دو آرگومان ورودی می‌باشد. آرگومان اول دنباله کاراکتری است که در واقع نمایانگر الگوی مورد استفاده جهت تجزیه دنباله کاراکتری به اجزای تشکیل‌دهنده آن است و آرگومان دوم به‌سادگی دنباله کاراکتری موردنظر است که توسط این تابع مورد تجزیه قرار می‌گیرد. تابع ( ) split علاوه بر این آرگومان‌های ضروری آرگومان دیگری را نیز در قالب یک آرگومان اختیاری دریافت می‌کند. این آرگومان اختیاری عدد صحیحی است که حداکثر تعدادی را که دنباله کاراکتری باید به آن تعداد تقسیم شود، مشخص می‌کند. تابع ( ) split پس از انجام فرآیند پیش‌بینی شده آرایه‌ای راکه‌شامل اجزای تشکیل‌دهنده دنباله کاراکتری اصلی است به‌عنوان نتیجه عملیات به برنامه فراخواننده باز می‌گرداند.

ارائه یک مثال، چگونگی استفاده از این تابع و نیز نحوه عملکرد آن‌را به‌خوبی نشان می‌دهد. در قطعه کدی که در ادامه مشاهده می‌کنید با بهره‌گیری از تابع ( ) split و همچنین استفاده از یک عبارت منظم که خود شامل دو انشعاب می‌باشد، سعی شده است تا یک دنباله کاراکتری به اجزای تشکیل‌دهنده خود تجزیه شود. مبنای این تجزیه براساس یک علامت کاما و به دنبال آن یک فضای خالی یک کاراکتری است. مبنای دیگری که این قطعه کد از آن جهت تجزیه دنباله کاراکتری استفاده می‌کند، واژه "and" است که توسط دو کاراکتر فضای خالی در دو طرف محصور شده باشد:

```
$text = "apples , oranges , peaches and grapefruit " ;
$fruitarray = split ( " , | and " , $text) ;
foreach ($fruitarray as $item)
   print "$item < BR >" ;
// output :
// apples
// orange
// peaches
// grapefruit
```

توجه کنید که تابع ( ) split نیز به‌مانند دو تابع ( ) ereg و ( ) ereg _ replace نسبت به استفاده از حروف بزرگ و کوچک حساس است. درصورتی‌که مایل باشید تا فرآیند تجزیه یک دنباله کاراکتری را به عناصر تشکیل‌دهنده آن به شکلی مستقل از اندازه حروف موجود در آن دنباله کاراکتری انجام دهید می‌توانید به‌جای استفاده از این تابع از تابع دیگری که ( ) spliti نام دارد، استفاده نمایید. تابع ( ) spliti از نقطه‌نظر ساختار فراخوانی و همچنین عملکرد کاملاً مشابه تابع ( ) split است. تنها تفاوتی که مابین این دو تابع وجود دارد این است که تابع مذکور برخلاف تابع ( ) split در تجزیه دنباله‌های کاراکتری نسبت به اجزای تشکیل‌دهنده آنها نسبت به بزرگی و کوچکی کاراکترهای به‌کار رفته در آنها حساسیت نشان می‌دهد.

عبارات منظم سازگار با زبان برنامه‌نویسی Perl

در صورتی که پیش از این با زبان برنامه‌نویسی perl اقدام به توسعه برنامه‌ها نموده باشید به‌احتمال قوی عبارات منظم سازگار با مجموعه استانداردهای POSIX را اندکی نامأنوس خواهید یافت. خبر خوشایندی که جا دارد از آن مطلع باشید این است که PHP4 علاوه بر آنچه که در قسمتهای قبل در این درس مشاهده نمودید از عبارات منظمی که به شیوه زبان برنامه‌نویسی perl ایجاد شده باشند نیز به‌خوبی پشتیبانی به‌عمل می‌آورد. نکته جالب توجهی که در اینجا باید به آن اشاره کنیم این است که عبارات منظم سازگار با زبان برنامه‌نویسی perl حتی توانمندی‌ها و قابلیتهای بیشتری را نسبت به آنچه که در مورد استاندارد POSIX مشاهده کردید، در اختیار برنامه‌نویسان قرار می‌دهد. در این قسمت از درس ضمن بررسی این نوع از عبارات منظم سعی خواهیم کرد تا در حد امکان تفاوتهای مابین هر دو شیوه را مورد بحث قرار دهیم.

تطبیق الگوها با استفاده از تابع ( ) preg _ match

تابع( ) preg _ match جهت تطبیق الگوها برمبنای عبارات منظم سازگار با perl مورد استفاده قرار می‌گیرد. این تابع ساختار نسبتاً ساده‌ای دارد به‌گونه‌ای که از سه آرگومان ورودی جهت انجام عملیات پیش‌بینی شده استفاده می‌کند. اولین آرگومان این تابع مطابق انتظار یک عبارت منظم است که برمبنای اصول بیان شده در زبان برنامه‌نویسی perl نوشته می‌شود (به‌زودی با این شیوه آشنا خواهید شد). آرگومان دوم یک دنباله کاراکتری است که فرآیند جستجو در داخل آن انجام می‌شود. سومین آرگومان این تابع نیز یک آرایه است. این آرایه موارد قابل تطبیق با الگوی تعریف شده را ذخیره خواهد کرد. تابع مورد بحث درصورتی‌که موردی قابل تطبیق را در درون دنباله کاراکتری مورد جستجو

(آرگومان دوم) پیدا کند، مقدار true و در غیر این‌صورت مقدار false را به‌عنوان نتیجه عملیات خود به برنامه فراخواننده باز می‌گرداند. تفاوت مابین این تابع و تابع مشابه با آن در استاندارد POSIX، یعنی preg _ match ( ) در اولین آرگومان که نماینده عبارت منظم مورد استفاده جهت تطبیق الگوست، نهفته است. در مورد عبارات منظم سازگار با زبان برنامه‌نویسی perl یک نکته عمومی وجود دارد و آن این است که این گونه عبارات منظم همواره مابین علایمی که آنها را از سایر بخشها جدا و متمایز می‌کند، واقع می‌شوند و نکته جالب‌تر اینکه این علایم در بیشتر مواقع همان علامت / یا slash هستند. این علامت جهت جداکردن عبارت منظم از سایر بخشها کاملاً در بین برنامه‌نویسان perl متداول شده است. با این حال درصورتی‌که تمایل داشته باشید، می‌توانید از کاراکتر دیگری که از نوع کاراکترهای الفبای عددی (کاراکترهای a تا z ، A تا Z و ارقام صفر تا ۹) نباشد به‌جای این علامت متداول بهره بگیرید (در این مورد تنها یک استثنا وجود دارد و آن این است که باید از به‌کار بردن کاراکتر \ خودداری کنید). جهت بررسی دقیق‌تر این تابع در اینجا قطعه برنامه‌ای را ارائه می‌کنیم که از این تابع جهت تطبیق یک الگو استفاده می‌کند. الگوی مورد استفاده در این قطعه کد جهت یافتن کاراکتر p است که به‌دنبال آن یک کاراکتر دلخواه و در نهایت کاراکتر t واقع شده باشد:

```
$text = "pepperpot " ;
if (perg _ match ( "/ p . t / " , $text , $array) )
   print $array [0] ;
// prints " pot "
```

طمع عبارات منظم سازگار با perl در یافتن موارد قابل تطبیق

بنا به پیش‌فرض، عبارات منظم سازگار با perl تا آنجا که ممکن باشد سعی می‌کنند تا هر تعداد از کاراکترها را که با الگوی موردنظر منطبق است، پیدا کنند. از این‌رو الگوی زیر:

```
"p . * t / "
```

منجر به این می‌شود که عبارت منظمی که آن‌را مورد استفاده قرار می‌دهد ابتدا کاراکتر ' p ' و سپس به‌دنبال آن هر تعداد کاراکتری را از دنباله کاراکتری اصلی که در نهایت به کاراکتر ' t ' منتهی می‌شود به‌همراه خود کاراکتر ' t ' ارزیابی قرار دهد. به‌همین جهت الگوی فوق در عبارت منظمی که قطعه کد زیر آن‌را مورد استفاده قرار داده باعث می‌شود تا کل دنباله کاراکتری اصلی (دنباله کاراکتری ذخیره شده در متغیر $text) مورد تطبیق واقع شود:

```
$text = "pot post pat patent" ;
if (perg _ match ( "/ p . * t / " , $text , $array ) )
   print $array [0] ;
// prints "pot post pat patent"
```

با بهره‌گیری از یک علامت سؤال (?) درست بعد از هر یک از مشخصه‌های کمیت جدول ۱-۱۸ که در ابتدای درس مشاهده کردید، می‌توان عبارت منظم سازگار با perl را به‌گونه‌ای مجبور کرد

تا از طمع خود در بازیابی هر تعداد کاراکتری که در سر راه آن از کاراکتر ' p ' به کاراکتر ' t ' وجود دارد، بکاهد. بدین ترتیب الگوی زیر:

"p . * t"

که به مفهوم بازیابی مواردی از دنباله کاراکتری که با حرف ' p ' آغاز شده و با حرف ' t ' خاتمه می‌یابند و مابین این دو ممکن است هر تعداد دلخواهی از کاراکترها موجود باشد را می‌توان به‌صورت زیر تعدیل نمود:

"p . * ? t"

الگوی فوق نیز به دنبال دنباله‌هایی از کاراکتر که با حرف ' p ' آغاز شده و با حرف ' t ' به اتمام می‌رسند جستجو می‌کند با این تفاوت که از بین موارد دارای شرایط فوق آن موردی را انتخاب می‌کند که در آن کاراکترهای کمتری مابین دو حرف فوق واقع شده باشند. به‌عبارت دیگر الگوی فوق از دو دنباله کاراکتری "pot" و "post" مورد اول را برمی‌گزینند.

در قطعه کد کوتاهی که در ادامه مشاهده می‌کنید سعی بر آن است تا با بهره‌گیری از این روش کوتاه‌ترین دنباله از کاراکترهایی را که دارای شرایط فوق هستند انتخاب نمود:

```
$text = "pot post pat patent " ;
if (preg _ match ( " / p . * ? t / ", $text , $array) )
    print $array [0] ;
// prints " pot "
```

استفاده از کاراکترهایی که پیش از علامت آنها علامت \ واقع شده است.

همان‌گونه که پیشتر نیز در این باره بحث کردیم، می‌توانید با بهره‌گیری از تکنیک escaping از کاراکترهای ویژه‌ای که معنای خاصی دارند، استفاده نمایید. در مورد عبارات منظم سازگار با perl، درست مثل دنباله‌های کاراکتری معمولی، هرجا که به یکی از این گونه کاراکترها جهت اعمال عملکرد خاصی احتیاج داشتید، می‌توانید پیش از آن کاراکتر از علامت \ استفاده کنید. برای نمونه استفاده از \ t در عبارت منظم به معنی کاراکتر Tab (که در اثر زدن کلید مربوطه از صفحه کلید حاصل می‌شود) می‌باشد. همچنین بهره‌گیری از \ n به معنای یک خط جدید است (مشابه زدن کلید Enter). عبارات منظم سازگار با perl گروه دیگری از این‌گونه کاراکترها را نیز که به‌واسطه آنها می‌توان تمامی انواع کاراکترهای موجود را مورد تطبیق قرار داد، معرفی کرده است. لیست این کاراکتر را به‌همراه توضیح مربوط به هر یک در جدول ۲- ۱۸ مشاهده می‌کنید.

جدول ۲- ۱۸ کاراکترهای ویژه‌ای که با انواع کاراکترهای موجود تطبیق می‌کنند

عنوان کاراکتر	موارد قابل تطبیق
\d	تمامی اعداد
\D	هر موردی به غیر از اعداد
\s	هر نوع فضای خالی
\S	هر موردی به غیر از فضای خالی
\w	تمامی کاراکترهای الفبا عددی (شامل کاراکتر underscor)
\W	تمامی موارد به‌غیر از کاراکترهای الفبا عددی یا underscore

کاراکترهای موجود در جدول فوق به‌گونه اعجاب انگیزی قادرند تا ساختار عبارات منظم موردنظرمان را ساده‌تر کنند. بدون وجود چنین کاراکترهایی تنها کاری که می‌شد در زمینه تطبیق با محدوده خاصی از کاراکترها انجام داد استفاده از کلاسهای کاراکتری بود. در اینجا برای اینکه به اهمیت کاری که این کاراکترها می‌توانند برای ساده کردن عبارات منظم انجام دهند، پی ببرید دو عبارت منظم را که یکی به سبک استاندارد POSIX و با بهره‌گیری از تابع ( ) ereg و دیگری به سبک زبان برنامه نویسی perl و با استفاده از تابع ( ) preg _ match عملیات موردنظر را انجام می‌دهد، ارائه می‌کنیم. توجه کنید که هر دوی این فراخوانی‌ها نتیجه یکسانی دربر خواهند داشت:

```
ereg ( "p [a - zA - Z0 - 9] + t " , $text , $array) ;
preg _ match ( "/ p \ w + t / " , $text , $array ) ;
```

علاوه بر کاراکترهای خاص موجود در جدول ۲- ۱۸، عبارات منظم سازگار با زبان perl تعداد دیگری از کاراکترهای خاص را نیز مورد پشتیبانی قرار داده است. با بهره‌گیری از این کاراکترهای خاص در عبارات منظم می‌توانیم موقعیت موردنظرمان را جهت تطبیق مشخص نماییم. جدول ۳- ۱۸ لیست این کاراکترها را به‌همراه توضیح مربوط به هر یک از آنها نشان می‌دهد.

جدول ۳- ۱۸ کاراکترهای خاص جهت تعیین موقعیت تطبیق الگو

عنوان کاراکتر	موقعیت تطبیق الگو
\A	ابتدای دنباله کاراکتری
\b	حدود (مرزهای) کلمه مورد نظر
\B	هر موقعیتی به‌غیر از حدود کلمه موردنظر

\Z	انتهای دنباله کاراکتری (پیش از آخرین علامت خط جدید یا انتهای دنباله)
\z	انتهای دنباله کاراکتری (تنها در انتهای دنباله)

چنانکه خاطرتان است ما در مثالهایی که در رابطه با عبارات منظم سازگار با استاندارد POSIX در نیمه اول این درس ارائه کردیم با مشکلی در رابطه با چگونگی تعیین مرزها یا حدود کدهای عضویت اعضای باشگاه فرضی خود مواجه بودیم. اکنون با امکاناتی که عبارات منظم سازگار با perl ارائه می‌کند. می‌توانیم مشکل فوق را به شیوه‌ای بسیار ساده‌تر از قبل حل و فصل نماییم. در اینجا بار دیگر سعی می‌کنیم تا فراخوانی تابع ( ) ereg در این رابطه را که در قسمتهای قبل نیز مشاهده کردید در کنار فراخوانی تابع ( ) preg _ match که از امکانات موجود در جداول ۲- ۱۸ و ۳- ۱۸ بهره‌گرفته است، قرار دهیم. بدین ترتیب خود می‌توانید بر میزان سادگی و قابلیت عبارات منظم سازگار با perl نسبت به مشابه خود گواهی دهید:

```
ereg ( "( ^ | [ ^ a - zA - Z0 - 9] ) ( p [a - zA - Z0 - 9] + t ) ([ ^ a - zA - Z0 - 9] |$) " ,
        $text , $array);
preg _ match ( "\ bp \ w + t \ b" , $text , $array) ;
```

چنانکه فراخوانی مربوط به تابع ( ) preg _ match را مشاهده می‌کنید ابتدا و انتهای عبارت منظم مورد استفاده شامل کاراکتر ویژه \ b است. همان‌گونه که در جداول مربوطه مشاهده کردید این کاراکتر خاص باعث می‌شود تا فرآیند تطبیق تنها درمورد کلماتی از یک دنباله کاراکتری انجام شود که مرزهای آنها با الگوی مورد استفاده در عبارت منظم منطبق باشد. به این ترتیب اولین بخش از الگوی مورد بحث، یعنی \ bp تنها منجر به آن کاراکترهای ' p ' خواهد شد که در ابتدای کلمه واقع شده باشند (واژه post نمونه‌ای از این کلمات است حال آنکه واژه tape چنین نیست). بخش بعدی الگوی موردنظر یعنی + w \ از کاراکتر خاص w \ استفاده کرده است. این کاراکتر خاص به مفهوم استفاده از کاراکتر از نوع الفبای عددی (تمامی حروف بزرگ و کوچک و ارقام صفر تا ۹) است. بدین ترتیب بخش مذکور به مفهوم استفاده از تعداد دلخواهی (علامت +) کاراکتر الفبای عددی است. پس تا بدین‌جا الگوی موردنظر حرف ' p ' در ابتدای کلمه و به دنباله آن تعداد دلخواهی کاراکتر الفبا عددی را مشخص می‌کند. بخش آخر الگوی مورد استفاده در عبارت منظم از فراخوانی تابع ( ) preg _ match یعنی t \ b باز هم بر شرایط مرزی اصرار دارد.

به بیان ساده این بخش از الگو تنها با آن حروف ' t ' که در انتهای کلمه واقع شده باشند، تطبیق خواهد کرد (بدین ترتیب واژه pet شامل این شرایط بوده ولی واژه byte چنین نیست). به‌عبارت بسیار ساده‌تر، الگوی مورد استفاده در این عبارت منظم واژه‌هایی را که با حرف ' p ' آغاز شده و با حرف ' t ' خاتمه پیدا کنند و مابین این دو شامل چند کاراکتر الفبای عددی (اختیاری) باشد، مورد تطبیق

قرار می‌دهد. واژه post یک چنین واژه‌ای می‌باشد. دقت کنید که کاراکتر b \ یعنی کاراکتر ویژه‌ای که به حدود مرزی جهت تطبیق تکیه می‌کند، در اصل با هیچ کاراکتری تطبیق نمی‌کند. این کاراکتر صرفاً بیان می‌دارد که وجود کاراکتر قابل تطبیق با الگو تنها درصورتی که در موقعیت مرزی ابتدا یا انتهای کلمه مورد نظر واقع باشد، دارای ارزش است. در مورد تابع ( ) ereg _ match لازم است تا ابتدا الگویی را جهت کاراکتر موردنظر ایجاد کرده و در گام بعدی شرایط مرزی را مورد بررسی قرار دهیم. با این حساب ساده بودن استفاده از عبارت منظم سازگار با perl کاملاً آشکار و واضح می‌باشد.

علاوه بر کاراکترهای فوق می‌توانید از تکنیک escaping جهت غیر فعال نمودن کاراکترهایی که معنی خاصی نیز می‌دهند، استفاده کنید. برای نمونه جهت تطبیق با کاراکتر نقطه (.) باید از علامت \ پیش از آن استفاده نمایید.

تطبیق سراسری با استفاده از تابع ( ) preg _ match _ all

یکی از مشکلاتی که در رابطه با عبارات منظم سازگار با POSIX وجود دارد، این است که تطبیق تمامی نمونه‌های موجود با الگوی مورد استفاده در این‌گونه عبارات فرآیندی است که به سختی انجام می‌شود. همان‌گونه که مطلع هستید با بهره‌گیری از تابع ( ) ereg جهت جستجوی کلماتی از یک دنباله کاراکتری که با حرف ' p ' آغاز شده و با حرف ' s ' خاتمه پیدا می‌کنند. تنها می‌توانیم اولین مورد قابل تطبیق با این الگو را پیدا کنیم. اجازه دهید موضوع فوق را با ارائه قطعه کد کوتاه زیر بیشتر مورد بررسی قرار دهیم. در این قطعه کد متغیر $text شامل دنباله کاراکتری اصلی است:

```
$text = "I sell pots , plants , pistachios , pianos and parrots" ;
if (ereg ( "( ^ | [ ^ a - zA - Z0 - 9 -] ) ( p [a - zA - Z0 - 9 -] + s ) ([ ^ a - zA - Z0 - 9 -] |$) " ,
          $text , $array) ) {
      for ($x = 0 ; is _ string ($array [$X] ) ; $X + +)
         print " \ $array [$X] : $array [$X] < br > \ n " ;
}
// out put :
// $array [ 0 ] : pots ,
// $array [ 1 ] :
// $array [ 2 ] : pots
// $array [ 3 ] : ,
```

همان‌گونه که انتظار می‌رفت اولین مورد قابل تطبیق با الگوی مورد استفاده با عنوان " pots " در سومین عنصر از آرایه ذخیره شده‌است. اولین عنصر از این آرایه یعنی $array [0] شامل تطبیق کاملی است که توسط تابع ( ) ereg صورت گرفته است. دومین عنصر شامل یک فضای خالی و چهارمین عنصر نیز شامل یک علامت کاما می‌باشد. برای اینکه کلیه موارد قابل تطبیق با الگو را بتوانیم مورد دستیابی قرار دهیم، لازم است تا از تابع ( ) ereg _ replace در درون یک حلقه تکرار استفاده کرده و در هر بار گذر از حلقه موردی را که اخیراً توسط تابع مورد تطبیق با الگو قرار گرفته است از

لیست حذف کرده و مجدداً فرآیند تطبیق را از سربگیریم. این روشی است که در واقع با بهره‌گیری از یک ساختار تکرار فقدان تطبیق کامل توسط تابع مورد بحث را به‌نوعی جبران می‌کنیم. به‌عبارت دیگر تابع فوق را باید به تعداد دفعاتی که موارد قابل تطبیق وجود دارد، فرابخوانیم.

از طرف دیگر، تابع preg _ match _ all ( ) قادر است تا تنها با یک مرتبه فراخوانی کلیه موارد قابل تطبیق با الگوی مورد استفاده را بازیابی نماید. تابع preg _ match _ all ( ) جهت انجام عملیاتی که برای آن درنظر گرفته شده است به سه آرگومان ورودی نیاز دارد. اولین آرگومان این تابع یک عبارت منظم است که شامل الگوی تطبیق می‌باشد. آرگومان دوم تابع فوق یک دنباله کاراکتری است که فرآیند جستجو در درون آن انجام می‌شود و بالاخره آرگومان سوم این تابع آرایه‌ای است که موارد قابل تطبیق با الگو را ثبت می‌کند. درصورتی‌که مورد قابل تطبیقی با این الگو وجود داشته باشد، تابع preg _ match _ all ( ) مقدار true را به برنامه فراخواننده باز می‌گرداند. دقت کنید که در ساختار تابع فوق سومین آرایه یک آرایه چندبعدی است، بدین معنی که هر یک از عناصر این آرایه خود از نوع آرایه هستند. عناصر اول هریک از این آرایه‌ها شامل یکی از مواردی است که با الگوی تعیین شده در عبارت منظم تطبیق می‌کند. بدین ترتیب مجموع آنها کل آن چیزی را که با الگوی مذکور تطبیق می‌کند، به دست خواهد داد.

برنامه‌موجوددرلیست۱- ۱۸ با بهره‌گیری از تابع preg _ match _ all ( ) و استفاده از دو ساختار for به صورت تودرتو اقدام به بازیابی و نمایش تمامی مقادیر قابل تطبیق با الگوی موجود نموده است.

```
 1: <html>
 2: <head>
 3: <title>Using preg_match_all() to match a pattern globally</title>
 4: </head>
 5: <body>
 6: <?php
 7: $text = "I sell pots, plants, pistachios, pianos and parrots";
 8: if ( preg_match_all( "/\bp\w+s\b/", $text, $array ) ) {
 9:     for ( $x=0; $x< count( $array ); $x++ ) {
10:         for ( $y=0; $y< count( $array[$x] ); $y++ )
11:             print "\$array[$x][$y]: ".$array[$x][$y]."<BR>\n";
12:         }
13: }
14: // Output:
15: // $array[0][0]: pots
16: // $array[0][1]: plants
17: // $array[0][2]: pistachios
18: // $array[0][3]: pianos
19: // $array[0][4]: parrots
20: ?>
21: </body>
22: </html>
```

لیست ۱- ۱۸ استفاده از تابع preg _ match _ all ( )

چنانکه در این برنامه ملاحظه می‌کنید اولین و تنها عنصر موجود در آرایه $array که در خط از برنامه فوق به‌عنوان آرگومان سوم به تابع preg _ match _ all ( ) ارسال شده است شامل آرایه‌ای از دنباله‌های کاراکتری می‌باشد. این آرایه تمام کلماتی از دنباله کاراکتری اصلی (دومین آرگومان تابع) که با حرف ' p ' آغاز شده و با حرف ' s ' خاتمه پیدا می‌کنند را شامل می‌شود.

تابع preg _ match _ all ( ) یک آرایه چند بعدی را جهت ذخیره کلیه موارد قابل تطبیق با الگوهای فرعی مقداردهی می‌کند. اولین عنصر از آرایه ارسالی به تابع مذکور شامل کلیه مواردی خواهد بود که با الگوی تعیین شده در عبارت منظم تطبیق می‌کنند.

به‌عبارت دیگر این عنصر از آرایه شامل یک تطبیق سراسری می‌باشد. عناصر بعدی آرایه $array هر یک شامل یکی از موارد قابل تطبیق با هر یک از الگوهای فرعی (اتم‌ها) که در داخل پرانتز قرار گرفته‌اند خواهد بود. از این‌رو با درنظر گرفتن دنباله کاراکتری موجود در متغیر $text و فراخوانی تابع preg _ match _ all ( ) به‌صورت زیر:

```
$text = "01 - 05 - 99 , 01 - 10 - 99 , 01 - 03 , 00 " ;
preg _ match _ all ( " / ( \ d +) - ( \ d +)-(\d+) / " , $text , $array) ;
```

اولین عنصر از آرایه $array یعنی $array [0] که یک آرایه چند بعدی است شامل کلیه موارد قابل تطبیق با الگوی تعیین شده در قالب عبارت منظم فوق می‌باشد:

```
$array [ 0 ] [ 0 ] : 01 - 05 - 99
$array [ 0 ] [ 1 ] : 01 - 10 - 99
$array [ 0 ] [ 2 ] : 01 - 03 - 00
```

همچنین دومین عنصر از آرایه $array یعنی $array [1] خود به مانند عنصر اول یک آرایه دوبعدی خواهد بود که شامل موارد قابل تطبیق با اولین الگوی فرعی (اتم) موجود در الگوی کلی می‌باشد:

```
$array [ 1 ] [ 0 ] : 01
$array [ 1 ] [ 1 ] : 01
$array [ 1 ] [ 2 ] : 01
```

عنصر سوم از آرایه $array یعنی $array[2] نیز وضعیتی مشابه عنصر دوم دارد. با این تفاوت که عناصر آن شامل موارد قابل تطبیق با دومین الگوی فرعی یا اتم موجود در الگوی کلی می‌باشد:

```
$array [ 2 ] [ 0 ] : 05
$array [ 2 ] [ 1 ] : 10
$array [ 2 ] [ 2 ] : 03
```

و این وضعیت در مورد سایر عناصر آرایه $array، به طرز مشابهی تکرار می‌شود.

استفاده از تابع preg _ replace ( ) جهت جایگزین کردن الگوها

تابعی که PHP جهت جایگزین کردن الگوهایی که برمبنای زبان برنامه‌نویسی perl ایجاد شده‌اند ارائه کرده است، تابعی است با عنوان preg _ replace ( ) که فرآیندی مشابه با تابع همتای خود

یعنی ereg _ replace ( ) را انجام می‌دهد به استثنای اینکه تابع مورد بحث دست برنامه‌نویس را جهت استفاده از عبارات منظم سازگار با زبان perl نیز باز می‌گذارد و این موضوع چنانکه می‌دانید به مفهوم قابلیت و توانمندی بیشتر از این تابع در جایگزینی الگوها نسبت به همتای خود در استاندارد POSIX یعنی تابع ereg _ replace ( ) می‌باشد. تابع preg _ replace ( ) ساختمان ساده‌ای از نقطه‌نظر فراخوانی دارد. وجود سه آرگومان کافی است تا این تابع فرآیند پیش‌بینی شده را به طرز مطلوب انجام دهد. اولین آرگومان این تابع یک عبارت منظم است که جهت تطبیق الگوها مورد استفاده قرار می‌گیرد. دومین آرگومان دنباله کاراکتری جایگزین و سومین آرگومان نیز دنباله کاراکتری اصلی را مشخص می‌کنند. چنانچه تابع preg _ replace ( ) موفق به یافتن دست کم یک مورد قابل تطبیق با الگوی مورد استفاده در عبارات منظم شود، مقداری را به برنامه فراخواننده باز می‌گرداند و دنباله کاراکتری جدیدی خواهد بود که بخشی از آن به‌دلیل جایگزینی دستخوش تغییر شده است. درغیر این‌صورت تابع فوق یک کپی کامل یکسان با دنباله کاراکتری اصلی را به برنامه فراخواننده باز خواهد گرداند. تابع preg _ replace ( ) علاوه براین آرگومان‌های ضروری می‌تواند با دریافت چهارمین آرگومان به‌عنوان یک آرگومان اختیاری توانمندی خود را توسعه دهد. این آرگومان اختیاری یک عدد صحیح است که بیشترین تعداد جایگزینی‌ها را که باید توسط این تابع صورت بگیرد، مشخص می‌نماید. قطعه برنامه‌ای که در ادامه مشاهده می‌کنید با بهره‌گیری ازتابع preg _ replace ( ) کلیه تاریخهای موجود در دنباله کاراکتری $t را که در قالب dd / mm / yy هستند به قالب mm / dd / yy تبدیل می‌کند:

```
$t = "25 / 12 / 99 , 14 / 5 / 00 " ;
$t = preg _ replace (" |\ b ( \ d +) / ( \ d +) / ( \ d +) \ b | " ,
      "$2 / $1 / $3 " , $t ) ;
print "$t < br > " ;
// prints  " 12 / 25 / 99 , 5 / 14 / 00 "
```

همان‌گونه که مشاهده می‌کنید، در این قطعه کد از علامتی موسوم به پایپ که ظاهری شبیه به یک پاره خط عمودی دارد (|) به‌عنوان عاملی جهت جداسازی استفاده کرده‌ایم. این علامت اجبار استفاده از شیوه escaping جهت بهره‌گیری از کاراکتر / به‌مفهوم اصلی خود را از میان می‌برد. به‌عبارت دیگر، درصورت عدم استفاده از علامت پایپ مجبور بودیم تا هر جا که در درون الگو عبارت منظم به علامت / نیاز داشتیم پیش از آن از کاراکتر \ استفاده کنیم (یعنی به‌صورت / \ ). لازم به توضیح است که استفاده از پس‌مرجعها، همان‌گونه که دررابطه با تابع ereg _ replace ( ) مشاهده کردید، به‌همراه تابع preg _ replace ( ) نیز امکان‌پذیر می‌باشد (یادآوری می‌کنیم که پس‌مرجع یا back reference دو علامت متوالی \ است که به‌دنبال آن عدد صحیحی واقع می‌شود. این عدد صحیح شماره اتم متناظر در الگوی مورد استفاده در عبارت منظم است).

نقطه‌قابل‌توجه‌دیگردراین‌کدشیوه‌متفاوتی است که برای استفاده از پس مرجعها موردبهره‌برداری واقع شده است. با وجودی که در عبارات منظم سازگار با زبان perl نیز می‌توانید به‌مانند عبارات منظم

POSIX از شیوه توضیح داده شده در پاراگراف قبل جهت به‌کارگیری پس‌مرجعها استفاده کنید، بهتر است مانند آنچه که در میان برنامه‌نویسان perl متداول است، از علامت دلار استفاده نمایید. در این شیوه جهت اشاره به یک اتم خاص از عبارت منظم موردنظر کافی است به‌جای استفاده از دو علامت متوالی \ از یک علامت $ استفاده کرده و به‌دنبال آن شماره اتم مورد نظر خود را ذکر کنید.

تابع ( ) preg _ replace قابلیت دیگری نیز دارد. به‌جای اینکه از یک دنباله کاراکتری به‌عنوان سومین آرگومان استفاده کنید، می‌توانید از آرایه‌ای که عناصر آن شامل دنباله‌های کاراکتری است، بهره بگیرید. در این صورت تابع ( ) preg _ replace فرآیند جایگزینی را به‌طور جداگانه بر روی هر یک از عناصر آرایه مذکور انجام می‌دهد. آنچه که در این حالت تابع فوق به برنامه فراخواننده باز می‌گرداند، آرایه‌ای‌از دنباله‌های کاراکتری است که بخشهایی از هریک از آنها به واسطه فرآیند جایگزینی دستخوش تغییر شده است. این نکته نیز به‌دلیل دیگری بر توانمندی عبارات منظم perl است.

اما اوج قابلیت و توانمندی این تابع در فرآیند جایگزینی از آن جهت است که تابع ( ) preg_replace می‌تواند به‌جای یک عبارت منظم تنها مجموعه‌ای از عبارات منظم را به‌عنوان اولین آرگومان دریافت نماید. در این حالت هر یک از این الگوها به‌طور مجزا به دنباله کاراکتری اصلی (آخرین آرگومان تابع) اعمال شده و دنباله کاراکتری جایگزین متناظر درصورت تطبیق الگو مورد استفاده قرار می‌گیرد. قطعه برنامه زیر با بهره‌گیری از این قابلیت تابع ( ) preg _ replace علاوه بر تغییر نحوه قالب‌بندی تاریخ اطلاعات مربوط به کپی رایت را نیز تغییر می‌دهد:

```
$text = "25 / 12 / 99 , 14 / 5 / 00. copyright 1999" ;
$regs = array ( " |\ b ( \ d +) / ( \ d + ) / ( \ d + ) \ b | " ,
        "/ ( [ Cc] opyright) 1999 / ") ;
$reps = array ( "$2 / $1 / $3" ,          "$1 2000") ;
$text = preg _ replace ($regs , $reps , $text) ;
print "$text < br >" ;
// prints " 12 / 25 / 99 , 5 / 14 / 00. Copyright 2000"
```

چنانکه در قطعه کد بالا مشاهده می‌کنید دو آرایه با اسامی $regs و $reps را جهت بهره‌مندی از قابلیت مورد بحث تابع ( ) preg _ replace ایجاد کرده‌ایم. آرایه اول با عنوان $regs شامل دو عبارت منظم است که یکی برای تطبیق تاریخ و دیگری برای تطبیق اطلاعات مربوط به کپی رایت ایجاد شده‌اند. آرایه دوم با نام $reps شامل دنباله‌های کاراکتری جایگزین می‌باشند. اولین عنصر از آرایه $reps با عنصر اول از آرایه $regs و دومین عنصر از آرایه $reps نیز با عنصر دوم از آرایه $regs متناظر است. درصورتی که مایل به تنوع بیشتری در فرآیند تطبیق باشیم می‌توانیم از آرایه‌هایی با تعداد بیشتری ازالگوها و دنباله‌های کاراکتری جایگزین استفاده نماییم. اما همواره باید توجه کنیم که رابطه یک به یک مابین دو آرایه فوق باید تأمین شود چه در غیر این‌صورت فرآیند جایگزینی به‌درستی صورت نمی‌پذیرد.

درصورتی‌که آرایه شامل دنباله‌های کاراکتری جایگزین (آرایه $reps در قطعه کد بالا) شامل تعداد عناصر کمتری نسبت به آرایه شامل عبارات منظم (آراپه $regs در قطعه کد بالا) باشد، موارد قابل تطبیق با آن عبارات منظمی که فاقد دنباله کاراکتری جایگزین هستند، بادنباله کاراکتری تهی تعویض می‌شوند.

همچنین درصورتی‌که آرایه شامل عبارات منظم به تابع ( ) preg _ replace ارسال شده باشد وتنها از یک دنباله کاراکتری جایگزین استفاده گردد، یک دنباله کاراکتری به‌عنوان دنباله کاراکتری جایگزین تمام موارد قابل تطبیق با الگوهای موجود در آرایه مذکور مورد استفاده قرار خواهد گرفت.

علایم تغییر دهنده

با بهره‌گیری از علایم ویژه‌ای که با عنوان علایم تغییردهنده می‌شناسیم می‌توانیم عبارات منظمی را به شیوه زبان برنامه‌نویسی perl بنویسیم که روش اعمال الگو را حین اجرای برنامه دستخوش تغییر نمایند.

واژه جدید علامت تغییردهنده حرف یا کاراکتری الفبایی است که باید پس از آخرین عامل جداکننده در عبارات منظم سازگار با زبان برنامه‌نویسی perl واقع شود. علامت تغییر دهنده تحت این شرایط قادر است تا رفتار عبارت منظم را دستخوش تغییر نماید.

لیستی از اسامی این‌گونه علایم به‌همراه توضیح مربوطه در جدول ۴- ۱۸ آمده است.

جدول ۴- ۱۸ علایم تغییردهنده قابل استفاده با عبارات منظم perl

علامت تغییردهنده	توضیح
/ i	این علامت باعث می‌شود تا اهمیت بزرگی و کوچکی حروف در فرآیند تطبیق نادیده گرفته شود.
/ e	این علامت باعث می‌شود تا دنباله کاراکتری جایگزین مورد استفاده به‌عنوان دومین آرگومان تابع ( ) preg _ replace به منزله کد PHP درنظر گرفته شود.
/ m	این علامت باعث می‌شود تا دوکاراکتر ^ و $ در الگوی عبارت منظم دنباله‌های کاراکتری را علاوه بر موقعیتهای شروع و پایان در خط جدید نیز مورد تطبیق قراردهند.
/ s	این علامت باعث می‌شود تا الگوی مورد استفاده در عبارت منظم، دنباله‌های کاراکتری را در موقعیت شروع خط جدید مورد تطبیق قرار دهد.
/ x	فضاهای خالی موجود در خارج ازکلاس کاراکترها در حالت استفاده از این

/ A	علامت به‌منظور افزایش میزان خوانایی عبارت منظم مورد ارزیابی قرار نمی‌گیرند. جهت تطبیق آنها می‌توان از علایم ویژه \s, \t و یا . \ استفاده کرد. این علامت باعث می‌شود تا فرآیند تطبیق تنها در ابتدای دنباله کاراکتری انجام شود (این علامت ویژه را نمی‌توان به‌همراه عبارات منظم سازگار با زبان perl مورد استفاده قرار داد).
/ E	این علامت باعث می‌شود تا فرآیند تطبیق تنها در انتهای دنباله کاراکتری انجام شود (این علامت ویژه را نمی‌توان به‌همراه عبارات منظم سازگار با زبان perl مورد استفاده قرار داد).
/ U	این علامت باعث می‌شود تا عبارت منظم مربوطه از طمع خود در رابطه با تطبیق‌هرچه بیشتر موارد قابل تطبیق صرف‌نظر نماید. بدین معنی که کمترین تعداد قابل تطبیق از دنباله کاراکتری مربوطه مورد تطبیق عبارت منظمی که از این علامت ویژه استفاده می‌کند، قرار می‌گیرد (علامت ویژه فوق را نیز نمی‌توان به‌همراه عبارات منظم سازگار با زبان برنامه‌نویسی perl مورد استفاده قرار داد).

از آنجا که ممکن است یک علامت خاص نظرات شما را جهت تطبیق با الگوی موردنظرتان تأمین نکند، به‌شرطی که این علایم با یکدیگر تناقضی نداشته باشند به‌راحتی می‌توانید آنها را جهت ساخت الگوی موردنظرتان با یکدیگر ترکیب نمایید. برای نمونه ممکن است ترجیح دهید تا از علامت ویژه x / جهت افزایش خوانایی عبارت منظم مورد نظرتان استفاده کرده و ضمناً از علامت ویژه دیگری مثلاً i / به‌منظور از بین‌بردن حساسیت موجود در رابطه با حروف بزرگ و کوچک استفاده نمایید. عبارت منظم " t / ix * \ $ \ b /" برای مثال با دنباله کاراکتری "bat" و همچنین "BAT" تطبیق می‌کند، حال آنکه درمورد دنباله کاراکتری "B A T " چنین تطبیقی صورت نمی‌گیرد. چنانکه ملاحظه می‌کنید فضاهای خالی موجود در این عبارت منظم به‌واسطه وجود علامت x / درفرآیند تطبیق الگو شرکت داده نمی‌شوند.

علامت ویژه m / برای مواقعی مفید است که بخواهیم به‌جای تطبیق الگو در ابتدا و انتهای یک دنباله کاراکتری این تطبیق را به ابتدا و انتهای هر یک از خطوط موجود در آن دنباله کاراکتری توسعه دهیم (البته درصورتی‌که دنباله کاراکتری مزبورشامل چندین خط باشد). دو الگوی ^ و $ چنانکه می‌دانید جهت تطبیق ابتدا و انتهای کل یک دنباله کاراکتری مورد استفاده قرار می‌گیرند. با بهره‌گیری از علامت ویژه m / به‌طریقی که در قطعه کد زیر ملاحظه می‌کنید، می‌توانیم رفتار الگوی $ را تغییر دهیم:

```
$text = "name : matt \ noccupation : coder \ neyes : blue \ n" ;
```

```
preg _ match _ all ( "/ ^ \ w + : \ s + (. *) $ / m" ; $text , $array ) ;
foreach ($array [ 1 ] as $val)
   print "$val < br >" ;
//output :
// matt
// coder
// blue
```

چنانکه در این قطعه برنامه ملاحظه می‌کنید عبارت منظم را به‌گونه‌ای ایجاد کرده‌ایم که ترکیب هر تعداد کاراکتر الفبایی (یا کاراکتر underscore) و به‌دنبال آن علامت کولون (:) و در نهایت هر تعداد فضای خالی دلخواهی را مورد تطبیق قرار دهد. این عبارت منظم سپس در ادامه هر تعداد از کاراکترهای‌دلخواه را که به دنبال آن از علامت خط جدید استفاده شده باشد، مورد تطبیق قرار می‌دهد. از آنجا که از علامت تغییردهنده m / بلافاصله پس از علامت $ استفاده شده است، علامت اخیر تاثیر خود را از انتهای دنباله کاراکتری به انتهای هر یک از خطوط موجود در دنباله کاراکتری توسعه می‌دهد.

علامت تغییردهنده s / برای مواقعی مفید است که بخواهیم با بهره‌گیری از علامت نقطه (.) کاراکترهای موجود در چندین خط از یک دنباله کاراکتری چندخطی را تطبیق دهیم. چنانکه در قطعه برنامه زیر مشاهده می‌کنید با بهره‌گیری از کاراکترهای تطبیقی معمولی و بدون استفاده از هیچ علامت تغییردهنده‌ای سعی کرده‌ایم تا اولین و آخرین کلمات موجود در یک دنباله کاراکتری را مورد دستیابی قرارداده واز طریق تابع ( ) print آنها را به خروجی ارسال نماییم:

```
$text = "start with this line \ nand you will reach \ na
      conclusion in the end \ n " ;
preg _ match ( " / ^ ( \ w + ) . * ? ( \ w + ) $ / " , $text , $array ) ;
print " $array [ 1 ] $array [ 2 ] < br >" ;
```

حقیقت این است که قطعه کد فوق هیچ‌گونه خروجی را نمایش نمی‌دهد. با وجودی که عبارت منظم به‌کار رفته در این کد کاراکترهای الفبایی (شامل underscore) را در ابتدای دنباله کاراکتری تشخیص می‌دهد اما علامت نقطه موجود در آن منجر به تطبیق کاراکترهای خط جدید تعبیه شده در متن‌اصلی‌نمی‌شود. با استفاده از علامت تغییردهنده s / به طریقی که در قطعه کد زیر مشاهده می‌کنید، می‌توانیم این وضعیت را تغییر دهیم:

```
$text = "start with this line \ nand you will reach \ na
      conclusion in the end \ n " ;
preg _ match ( " / ^ ( \ w + ) . * ? ( \ w + ) $ / s " , $text , $array ) ;
print " $array [ 1 ] $array [ 2 ] < br >" ;
// prints "start end "
```

علامت تغییردهنده e / ازجمله علایم توانمند این مجموعه است. به‌واسطه این علامت می‌توان دنباله کاراکتری جانشین موجود در تابع ( ) preg _ replace را به منزله دنباله کاراکتری ساده‌ای در PHP تصور کرد. بدین ترتیب برای نمونه می‌توان از این طریق ارسال پس از مرجعها به توابع مختلف یا پردازش لیستی از اعداد را به سهولت هرچه تمام‌تر انجام داد. در قطعه برنامه‌ای که در ادامه مشاهده می‌کنید از این علامت تغییر دهنده جهت ارسال اعداد قابل تطبیق موجود در یک تاریخ به تابعی که همان تاریخ را در قالب نمایش دیگری باز می‌گرداند، استفاده کرده‌ایم:

```
< ? php
function convDate ($month , $day , $year) {
   $year = ($year < 70) ? $year + 2000 : $year;
   $time = (mktime (0, 0, 0, $month , $day, $year) ) ;
   return date (" l d F Y" , $time) ;
}
$dates = "3 / 18 / 99 < br > \ n 7 / 22 / 00" ;
$dates = preg _ replace (" / ( [ 0 - 9]+ ) \ / ( [ 0 - 9] + ) \ / ( [ 0 - 9] + ) / e" ,
      "convDate ($ 1 , $2, $3)" , $dates) ;
print $dates ;
// prints
// Thursday 18 March 1999
// Saturday 22 July 2000
? >
```

چنانکه در کد فوق مشاهده می‌کنید، با استفاده از پرانتزهایی که جهت تعیین اعداد استفاده کرده‌ایم هر تعداد مجموعه سه عددی را که با علامت / از یکدیگر جدا شده‌اند (قالب تاریخ)، مورد تطبیق قرار داده‌ایم. به‌دلیل استفاده از علامت تغییردهنده e / در این عبارت منظم می‌توانیم در فراخوانی تابع ( ) convDate جهت مشخص نمودن دنباله کاراکتری جانشین از ارسال پس‌مرجعها به این تابع بهره‌برداری نماییم. تابع ( ) convDate عملکرد کاملاً ساده و مشخصی دارد. این تابع با دریافت سه عدد صحیح از ورودی، تاریخ خاصی را قالب‌بندی کرده و حاصل را به برنامه فراخواننده بازمی‌گرداند. این تاریخ جهت جای‌گزین شدن با تاریخی که از طریق عبارت منظم مورد تطبیق قرار گرفته است، استفاده می‌شود. به‌دلیل اینکه در برنامه فوق هدف ما تطبیق اعداد بوده است، نیازی نیست تا پس‌مرجعها را توسط علامت کوتیشن مشخص نماییم. درصورتی‌که قصد ما تطبیق دنباله‌های کاراکتری بوده باشد استفاده از علامت مذکور جهت محصور نمودن هر یک از پس مرجعها به امری ضروری مبدل می‌شود.

استفاده از تابع ( ) preg _ replace_ callback جهت جایگزین کردن الگوها

با بهره‌گیری از تابع ( ) preg _ replace_ callback می‌توان اقدام به تعیین تابعی کرد که به ازای هر تطبیق کامل یافت شده یا الگوی موجود در عبارت منظم فراخوانی شود (به تابعی که این

چنین فراخوانی گردد، تابع callback گفته می‌شود). تابع ( ) preg _ replace_ callback ساختار نسبتاً ساده‌ای داشته و جهت انجام عملیات موردنظر خود مستلزم دریافت سه آرگومان ورودی می‌باشد. آرگومان اول این تابع یک عبارت منظم است که جهت یافتن موارد قابل تطبیق با الگو به‌کار می‌رود. آرگومان دوم مرجعی است که تابع callback موردنظر را مشخص می‌کند و بالاخره آرگومان سوم دنباله کاراکتری را مشخص می‌کند که تابع ( ) preg _ replace_ callback جهت پردازش آن فراخوانی می‌شود. تابع مورد بحث به مانند تابع ( ) preg _ replace آرگومان دیگری را نیز به‌عنوان آرگومان اختیاری مورد استفاده قرار می‌دهد. آرگومان مذکور که یک عدد صحیح است حداکثر تعداد موارد جایگزین را مشخص می‌کند.

تابع callback در این میان باید به‌گونه‌ای تعریف شود که یک آرایه تنها را به‌عنوان آرگومان ورودی دریافت نماید.

اولین عنصر از این آرایه شامل موردی است که با الگوی عبارت منظم تطبیق کامل دارد. عناصر بعدی آرایه مذکور، هر یک شامل موارد قابل تطبیق با الگوهای فرعی یا اتمهای موجود در الگوی کلی خواهند بود. هر آنچه که تابع callback به‌عنوان نتیجه عملیات باز گرداند، به‌عنوان بخشی از دنباله کاراکتری حاصل از فراخوانی تابع ( ) preg _ replace_ callback مورد بهره‌برداری قرار خواهند گرفت.

جهت درک بهتر چگونگی عملکرد تابع ( ) preg _ replace_ callback به قطعه کد زیر که نسخه بازنویسی شده مثال تغییر قالب تاریخ است، توجه نمایید:

```
Function convDate ($matches) {
   $year = ($year < 70) ? $matches [ 3 ] + 2000 : $mathches [ 3 ] ;
   $time = (mktime (0, 0, 0, $matches [ 1 ] , $matches [ 2 ] ,
         $matches [ 3 ] ) ) ;
   return date ( "l d F Y " , $time) ;
}
$dates = "3 / 18 / 99 < br > \ n7 / 22 / 00 " ;
$dates = preg _ replace _callback (" / ( [ 0 - 9]+ \ / ( [ 0 - 9] + ) \ / ( [ 0 - 9] + ) / " ,
          "convDate" , $dates ) ;
print $dates ;
// prints
// Thursday 18 March 1999
// Soturday 22 July 2000
```

چنانکه در این قطعه برنامه مشاهده می‌کنید، تابع ( ) convDate را با دوبار مورد فراخوانی قرار داده‌ایم. هر یک از فراخوانی‌های فوق به‌ازای یافتن یک مورد قابل تطبیق با الگو صورت گرفته است. بدین ترتیب بخشهای روز و ماه و سال به‌آسانی از آرایه‌ای که به‌عنوان آرگومان به تابع ( ) convDate ارسال شده‌اند، قابل استخراج می‌باشند. بخشهای یاد شده به‌ترتیب در عناصر اول تا سوم این آرایه ذخیره شده‌اند.

جمع‌بندی

مبحث عبارات منظم به‌واقع مبحث سنگین و حجیمی است و ما در درس این ساعت تنها به‌طور خلاصه گوشه‌هایی از آن‌را با هم مورد بررسی قرار دادیم. اکنون با دانشی که به‌دست آوردید باید بتوانید عبارات منظم مختلفی را جهت تطبیق و جایگزین کردن بخشهای موردنظرتان از یک دنباله کاراکتری مورد استفاده قرار دهید.

همچنین باید بتوانید با بهره‌گیری از تابع ( ) ereg و ارسال عبارت منظم مناسبی به آن، الگوهای موردنظرتان را در دنباله‌های کاراکتری مختلف مورد تطبیق قرارداده و نیز با بهره‌گرفتن از تابع ( ) ereg _ replace تمامی موارد قابل تطبیق با یک الگوی خاص را با دنباله کاراکتری موردنظرتان تعویض نمایید. علاوه براین باید بتوانید با استفاده از کلاسهای کاراکتر که در این درس با آنها آشنا شدید اقدام به یافتن محدوده‌ای از کاراکترها نموده و با بهره‌گیری از مشخصه‌های کمیت از چندین الگو جهت تطبیق استفاده کنید. در درس این ساعت با چگونگی دخالت دادن عامل موقعیت در فرآیند تطبیق با استفاده از دو مشخصه $ و ^ آشنا شدید. همچنین متوجه شدید که عبارات منظمی که به شیوه زبان برنامه‌نویسی perl توسعه پیدا می‌کنند از قابلیتها وامکانات مؤثرتری نسبت به همتای خود در استاندارد POSIX برخوردارند. با دانشی که اکنون در مورد این‌گونه عبارات منظم به دست آوردید باید بتوانید با بهره‌گرفتن از تکنیکهای مختلفی مانند escaping درمورد کاراکترهای خاص اقدام به تطبیق انواع مختلف کاراکترها نموده و موقعیت تطبیق را به‌دلخواه خود توسعه دهید، همچنین باید قادر باشید تا با به‌کارگرفتن علایم تغییردهنده ویژه‌ای که در این درس فراگرفتید شیوه‌ای را که عبارات منظم سازگار با perl جهت تطبیق الگوها مورد استفاده قرار می‌دهند، دستخوش تغییر نمایید.

در درس ساعت آینده که به بررسی کوکی‌ها ودنباله‌های پرس و جو اختصاص دارد با برخی از تکنیکهایی که جهت ثبت وضعیت ما بین درخواستهای مختلف مورد استفاده قرار می‌گیرند، آشنا خواهید شد.

پرسش و پاسخ

پرسش: آن‌گونه که از ظواهر امر پیداست عبارات منظم سازگار با perl از توان بالایی در فرآیند تطبیق و بازیابی مواردی از دنباله‌های کاراکتری برخوردارند. آیا منابع دیگری که بتوان از طریق آن اطلاعات بیشتری در مورد آنها به دست آورد، وجود دارد؟

پاسخ: بخش مربوط به عبارات منظم PHP به آدرس http: // www. php. Net، اطلاعات مفید و مختلفی را در مورد چگونگی ایجاد عبارات منظم دراختیارتان قرارمی‌دهد. علاوه بر این اطلاعات مفید بسیار مختلفی را در این مورد می‌توانید از طریق آدرس http: // www. perl. Com به دست آورید.

به‌ویژه مقدمه‌ای بر عبارات منظم perl که در آدرس http: // www. perl. Com / pub/ doc / manual/ html / pod/ perler. Html آمده و همچنین مقاله ارزشمندی به آدرس http: // www. perl. Com / pub / doc/ manual / html / pod / perlfaq6 . html که در این‌باره توسط Tom Christiansen به رشته تحریر درآمده، منابعی هستند که با مطالعه آنها اطلاعات بسیار مفید و جالبی درمورد عبارات منظم perl فرامی‌گیرید. اما به‌عنوان یک منبع بسیار عالی و ارزشمند و جامع توصیه می‌کنیم که کتاب Mastering Regular Expression نوشته Jeffry Friedl را از انتشارات O'Reilly تهیه نمایید.

تمرینها

هدف از این بخش ارائه تمرینهایی در قالب آزمون است. پاسخ بخش آزمون بلافاصله بعد از آن آمده است. بخش فعالیتها شامل تمرینهایی است که به‌منظور افزایش مهارت خواننده در رابطه با قابلیتهای برنامه‌نویسی وی طراحی شده است. بخش فوق فاقد پاسخ است.

آزمون

۱- کدام تابع از سری توابعی که استاندارد POSIX جهت کار با عبارات منظم ارائه کرده است را می‌توان به منظور تطبیق الگو در درون یک دنباله کاراکتری مورد استفاده قرار داد؟

۲- از کدام عبارت منظم می‌توان جهت تطبیق حرف ' b ' به تعداد دست‌کم یک مرتبه و حداکثر شش مرتبه استفاده کرد؟ عبارت منظم مربوطه را بنویسید؟

۳- عبارت منظمی جهت تطبیق محدوده کاراکترهای ' d ' تا ' f ' بنویسید.

۴- عبارت منظمی جهت تطبیق محدوده‌ای از کاراکترها بنویسید که شامل محدوده تمرین قبل نباشد.

۵- عبارت منظمی جهت تطبیق هر عدد دلخواه یا واژه "tree" بنویسید.

۶- از کدام تابع PHP در رابطه با عبارات منظم استاندارد POSIX می‌توان جهت جایگزین کردن یک الگو استفاده کرد؟

۷- عبارت منظم زیر را درنظر بگیرید:

. * bc

همان‌گونه که می‌دانید این عبارت با طمع بسیار زیاد فرآیند تطبیق را انجام می‌دهد به‌گونه‌ای که تمام موارد قابل تطبیق را از دنباله "abc 000000 bc " گرفته تا " abc " مورد بازیابی قرار می‌دهد. با بهره‌گیری از تکنیکهایی که در درس این ساعت راجع به عبارات منظم سازگار با زبان perl فراگرفتید، ترتیبی دهید تا فرآیند تطبیق تنها درباره اولین مورد یافت شده توسط عبارت منظمی که می‌نویسید انجام شود.

۸- در مورد عبارات منظم سازگار با زبان perl از چه کاراکتر ویژه‌ای به‌همراه علامت \ می‌توان جهت تطبیق فضاهای خالی موجود در دنباله‌های کاراکتری استفاده کرد؟
۹- در مورد عبارات منظم سازگار با زبان perl از کدام تابع می‌توان جهت تطبیق تمامی موارد منطبق با الگو در یک دنباله کاراکتری استفاده کرد؟
۱۰- از کدام علامت تغییردهنده می‌توان در توابع مربوط به عبارات منظم سازگار با زبان perl جهت تطبیق الگوها بدون توجه به بزرگی و کوچکی حروف استفاده نمود؟

پاسخ آزمون

۱- از تابع ( ) ereg می‌توان جهت تطبیق یک الگو در دنباله‌ای از کاراکترها استفاده کرد.
۲- جهت تعیین تعداد دفعات تکرار یک کاراکتر در الگوی عبارت منظم لازم است کمترین و بیشترین دفعات تکرار نمونه مورد نظر را در داخل جفت علامت { } پس از نمونه مذکور نوشته و آنها را با استفاده از علامت کاما از یکدیگر جدا کرد:

```
b {1, 6}
```

۳- تعیین حدود کاراکترها با بهره‌گیری از جفت علامت [ ] به‌صورت زیر امکان‌پذیر است:

```
[ d - f ]
```

۴- تعیین عکس محدوده کاراکتری موردنظر با بهره‌گیری از علامت caret (کاراکتر ^) درست پیش از محدوده مذکور و دردرون جفت علامت [ ] به‌صورت زیر امکان‌پذیر است:

```
[ ^ d - f ]
```

۵- با استفاده از علامت pipe (کاراکتر |) مابین دو الگو می‌توان به‌صورت زیر این کار را انجام داد:

```
[ 0 - 9 ] | tree
```

۶- با بهره‌گیری از تابع ( ) ereg _ replace می‌توان اقدام به جای‌گزینی الگوهای تعریف شده بر مبنای استاندارد POSIX نمود.
۷- با اضافه کردن یک علامت سؤال به یک مشخصه کمیت می‌توان در عبارات منظم سازگار با زبان perl ترتیبی داد تا الگوی مورد استفاده از طمع خود در مورد بازیابی تمامی موارد قابل تطبیق صرف‌نظر نماید:

```
/ . * ? bc /
```

۸- علامت ویژه s \ در عبارات منظم سازگار با زبان perl منجربه تطبیق فضای خالی خواهد شد.
۹- با بهره‌گیری از تابع ( ) preg _ match _ all در عبارات منظم سازگار با زبان برنامه‌نویسی perl می‌توان کلیه موارد قابل تطبیق با الگو را مورد بازیابی قرار داد.
۱۰- استفاده از علامت تغییردهنده i / در عبارات منظم سازگار با زبان perl باعث خواهد شد تا تطبیق الگوها بدون توجه به بزرگی و کوچکی حروف انجام شود.

فعالیتها

۱- با استفاده از عبارات منظم ترتیبی دهید تا کلیه آدرسهای email موجود در یک فایل بخصوص مورد بازیابی قرار گیرند. سپس آنها را به آرایه‌ای از نوع دنباله‌های کاراکتری اضافه نموده و محتوای این آرایه را بر روی صفحه مرورگر اینترنت نمایش دهید. عبارت منظم خود را به‌گونه‌ای تغییر دهید که قابلیت خود در بازیابی آدرسها را به چندین فایل مختلف توسعه دهد.

ساعت نوزدهم

ثبت وضعیت با استفاده از کوکی‌ها ودنباله‌های پرس‌وجو

قراداد HTTP یک قرارداد بدون حالت یا اصطلاحاً stateless است. این عبارت بدان معنی است که هر صفحه‌ای که کاربر از یک وب سرور بارگذاری می‌کند به‌منزله یک ارتباط مستقل درنظر گرفته می‌شود. به بیان‌دیگر، سایتهای وب توسط کاربران و ناشران محتویات این سایتها به‌عنوان محیط خاصی تلقی می‌شوند که در داخل آن هریک از صفحات منفرد به‌عنوان بخشی از یک فضای بزرگ‌تر درنظر گرفته می‌شود. از این‌رو تعجبی ندارد اگر بدانیم روشهای مربوط به ارسال اطلاعات مختلف مورد نیاز از یک صفحه دیگر که به قصد حفظ وضعیت صورت می‌پذیرد قدمتی به‌اندازه خود وب دارند؛ یعنی بستری که محیط مذکور به‌همراه کلیه صفحات و اسناد وب در آن واقع است.

دردرس‌این‌ساعت با دو روش مهم و متداول جهت ثبت و ذخیره اطلاعات موجود در یک صفحه آشنا می‌شوید. همان‌گونه که به‌زودی مشاهده خواهید کرد این اطلاعات توسط سایر صفحات موجوددر آن‌سایت‌مورددستیابی‌قرارخواهندگرفت.عناوین‌مطالبی‌که در این درس بررسی می‌کنیم، به قرار زیر است:

- **مفهوم کوکی و چگونگی عملکرد آن**
- **چگونگی بازخوانی اطلاعات ذخیره شده در یک کوکی**
- **چگونگی ایجاد کوکی و ثبت اطلاعات مورد نظر در آن**
- **چگونگی بهره‌گیری از کوکی‌ها جهت ثبت اطلاعات مربوط به کم و کیف استفاده از وب سایت در یک بانک اطلاعاتی**
- **مفهوم دنباله پرس و جو**
- **چگونگی ایجاد تابعی جهت تبدیل یک آرایه انجمنی به دنباله پرس و جو**

مفهوم کوکی

شرکت Netscape با ارائه اولین نسخه از مرورگر اینترنت معروف خود با همین نام برای اولین بار کوکی را معرفی کرد. اینکه مرجع نام کوکی (cookie) چه بوده و به چه دلیل شرکت مورد بحث از این عنوان جهت نام‌گذاری آن استفاده کرده است، جای بحث و گفتگو دارد. با این حال چنین به‌نظر می‌رسد که به معنی ظاهری این واژه بی‌ربط نبوده و این معنی مبنایی را برای نام‌گذاری تشکیل می‌دهد. از آن زمان تا به‌حال کوکی به‌عنوان ابزار استانداری مورد استفاده مرورگرهای مختلف ایجاد شده توسط شرکتهای کامپیوتری قرار گرفته و وب سایتهای بسیاری نیز برمبنای آن ایجاد شده است.

واژه جدید کوکی مجموعه کوچکی از داده‌هاست که توسط مرورگر اینترنت و طی فرآیند درخواستی که از وب سرور یا یک برنامه اسکریپت صورت می‌گیرد، برروی کامپیوتر ذخیره می‌شود. هر میزبانی می‌تواند تعداد حداکثر ۲۰ کوکی را جهت ذخیره بر روی کامپیوتر کاربر ارسال نماید. هر کوکی از یک نام، مقدار، تاریخ انقضا ، نام میزبان و اطلاعات مسیر تشکیل می‌شود. حداکثر اندازه یک کوکی نباید از ۴ کیلو بایت متجاوز شود.

پس از ارسال یک کوکی بر روی کامپیوتر کاربر، تنها میزبانی که اقدام به ارسال آن نموده حق بازخوانی داده‌های موجود در آن‌را دارد. به‌این ترتیب این اطمینان حاصل می‌شود که اطلاعات کاربر در دسترس سایر میزبانها قرار نمی‌گیرد. علاوه بر این مرورگرهای اینترنت تنظیماتی دارند که از طریق آنها می‌توان ترتیبی داد که به‌محض ایجاد کوکی توسط میزبان و ارسال آن به مرورگر کاربر را در جریان قرار داده و از این رو کاربر می‌تواند با قبول آن اقدام به ذخیره کوکی ارسالی بر روی کامپیوتر خود نموده یا اینکه به کل از قبول آن سرباز زند. از این‌رو همواره باید به این نکته توجه کرد که نباید از کوکی به‌عنوان ابزار اصلی و مهم‌ترین عاملی که قابلیتهای ارائه شده در وب سایت کاملاً متکی به آن است استفاده کرد مگر آنکه پیشتر در این مورد کاربر را از این وضعیت آگاه نمود. درعوض می‌توان از این عناصر کوچک جهت اهداف میانی با اهمیت معمولی استفاده نمود.

با مطالب عنوان شده می‌توان چنین بیان کرد که کوکی‌ها ابزارهایی بسیار عالی جهت ذخیره حجم کوچکی از اطلاعات درباره کاربر محسوب می‌شوند؛ چراکه می‌توان کوکی‌ها را از یک صفحه به صفحه دیگر و حتی از یک نشست به نشست دیگر حفظ نموده و در اختیار گرفت.

بررسی ساختار یک کوکی

کوکی‌ها معمولاً از طریق یک هدر HTTP تنظیم و ارسال می‌شوند (با این وجود JavaScript اجازه می‌دهد تا بتوان کوکی را مستقیماً بر روی مرورگر اینترنت ارسال کرد). لیست ۱- ۱۹ اجزای

مختلف یک هدر HTTP را که جهت ارسال به مرورگر آماده شده است، نشان می‌دهد. دقت کنید که تنظیم کوکی بخشی از این هدر است.

```
HTTP/1.1 200 OK
Date: Tue, 02 Oct 2001 13:39:43 GMT
Server: Apache/1.3.12 Cobalt ( Unix ) PHP/4.0.6 mod_perl/1.24
X-Powered-By: PHP/4.0.6
Set-Cookie: vegetable=artichoke; expires=Tue,
[ic:ccc]02-Oct-01 14:39:43 GMT; path=/;
domain=corrosive.co.uk
connection: close
Content-Type: text/html
```

لیست ۱- ۱۹ یک هدر HTTP نمونه که جهت ارسال به مرورگر آماده شده است.

همان‌گونه که در تنظیم مربوط به کوکی در این لیست مشاهده می‌کنید هدر set _ cookie شامل یک نام و مقدار متناظر با آن، یک تاریخ بر مبنای GMT، اطلاعات مسیر و درنهایت نام میزبان می‌باشد. نام و مقدار متناظر با آن در این میان از نوع رمزگذاری شده است (جهت توضیح بیشتر در این زمینه به درسهای قبل مراجعه کنید). وجود فیلد expire با مقدار true آن‌چنان که در این هدر نشان داده شده است، باعث خواهد شد تا مرورگر اینترنت پس از انقضای زمان مربوطه وجود یک چنین کوکی را فراموش نماید. اطلاعات مسیر موجود در فیلد path نیز موقعیتی را بر روی وب سایت مشخص می‌کند که کوکی مورد نظر به محض بارگذاری اسنادی تحت آن موقعیت بر روی مرورگر اینترنت، باید به ارسال کننده آن بازگردانده شود. به‌طوردقیق‌تر، فیلد مذکور جهت تعیین مسیر عمومی‌تری از وب سایت که کوکی مذکور متعلق به آن است، مورد استفاده قرار می‌گیرد. فیلد domain یک حوزه اینترنتی (نام میزبان) را که کوکی موردنظر باید به آن ارسال شود، مشخص می‌کند. مقدار این فیلد نباید متفاوت با نام حوزه‌ای که کوکی از طریق آن ارسال شده است، باشد با این حال مقدار فیلد فوق می‌تواند بیانگر کمی آزادی عمل باشد. در مثال لیست فوق چنانکه ملاحظه می‌کنید مرورگر اینترنت باید کوکی تنظیم شده را به سروری که با عنوان corrosive. co. uk تعیین شده است، ارسال نماید، ضمناً این کوکی با عنوان www. corrosive. co. uk نیز ارسال می‌شود. برای مطالعه مطالب بیشتر در این زمینه توصیه می‌کنیم درس ساعت سیزدهم با عنوان " بررسی عملیات سمت سرور" را بار دیگر مرور نمایید.

درصورتی که مرورگر اینترنت جهت پذیرفتن و ذخیره کوکی‌ها تنظیم شده باشد، اطلاعات موجود در آنها را تا تاریخ انقضای آنها حفظ خواهد کرد. در این صورت چنانکه مرورگر سندی را که مسیر آن با مسیر تعیین شده در فیلد path از یک کوکی بخصوص منطبق باشد درخواست کند یا اسکریپتی را که در آن موقعیت واقع است اجرا نماید، مرورگر اینترنت کوکی مذکور طی این فرآیند به سرور ارسال خواهد شد. همانند هدری که سرور به برنامه مرورگر ارسال می‌کند مرورگر اینترنت نیز

طی هر درخواست هدری را به سرور ارسال خواهد کرد. لیست ۲-۱۹ نمونه‌ای از یک چنین هدری را نشان می‌دهد.

```
GET / HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.73 (Macintosh; U; PPC)
Host: www.corrosive.co.uk
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
image/png, */*
Accept-Encoding: gzip
Accept-Language: en, pdf
Accept-Charset: iso-8859-1, *, utf-8
Cookie: vegetable=artichoke
```

لیست ۲- ۱۹ یک نمونه از هدری که مرورگر اینترنت به سرور ارسال می‌کند

پس از ارسال کوکی از برنامه مرورگر به سرور کوکی موردنظر توسط یک برنامه اسکریپت PHP نمونه که جهت دریافت کوکی مذکور مجهز به کد مربوطه شده است، مورد دستیابی قرارمی‌گیرد. برنامه PHP کوکی مورد بحث را از طریق متغیر سیستمی ویژه‌ای با عنوان HTTP _ COOKIE (که شامل اسامی و مقادیرمربوطه از کوکی مورد نظر می‌باشد) مورد دستیابی قرار می‌دهد. در مورد مثال لیست فوق تنها زوج نام و مقدار موجود به صورت vegetable = artichoke تعریف شده است که در این صورت می‌توان مقدار متناظر با این نام را از طریق متغیر سراسری همنام، یعنی $vegetable و یا از طریق آرایه سراسری HTTP_COOKIE_VARS که یک آرایه انجمنی است به صورت HTTP_COOKIE_VARS[' vegetable'] مورد دستیابی قرارداد. هر دوی این روشها در قطعه کد زیر نشان داده شده‌اند:

```
Print "$HTTP_ COOKIE < BR >" ; // prints "vegetable = artichoke"
Print getenv ( "HTTP_ COOKIE" ). "< BR >" ; // prints " vegetable = artichoke"
Print "$ vegetable < BR >" ; // print  "artichoke"
Print $HTTP_ COOKIE _ VARS [ ' vegetable ' ] . "<BR>";
```

نحوه تنظیم و ارسال کوکی با استفاده از کد PHP

با تسهیلاتی که در زبان برنامه‌نویسی PHP پیش‌بینی شده است می‌توان به‌سادگی اقدام به تنظیم و ارسال کوکی‌های مورد نظر بر روی مرورگرهای اینترنت مورد استفاده کاربران نمود. در زبان برنامه‌نویسی PHP جهت انجام یک چنین کاری دو روش متداول وجود دارد. روش اول استفاده از تابع ویژه‌ای با نام ( ) header است. به کمک این تابع می‌توان هدر cookie _ set از هدر HTTP ارسالی به مرورگر اینترنت را تنظیم نمود. چنانکه به‌خاطر دارید، در درس ساعت نهم با عنوان " بهره‌گیری از فرم‌ها" تابع ( ) header را مورد بحث و بررسی قرار دادیم. تابع مورد بحث جهت تنظیمات مربوط به

هدر موردنظر تنها به یک دنباله کاراکتری به‌عنوان آرگومان ورودی نیازدارد. این دنباله کاراکتری شامل تنظیمات بخش هدر است که به‌عنوان بخشی از پاسخ سرور برای برنامه مرورگر اینترنت ارسال می‌گردد. از آنجا که هدرهای HTTP طی فرآیند ارتباط مابین مرورگر و سرور به‌طور خودکار مابین این دو رد و بدل می‌شوند، همواره لازم است تا بهره‌گیری از تابع ( ) header جهت تنظیم آن در اولویت بالاتری نسبت به کدی که اطلاعات درخواستی را برای مرورگر ارسال می‌کند، باشد. قطعه کد زیر نمونه‌ای از چگونگی فراخوانی این تابع را نشان می‌دهد:

```
header ( "Set_ Cookie: vegetable = artichoke; expires = Tue,
    [ ic : ccc] 02 _ Oct _ 01  14 : 39 : 58 GMT; path = / ; domain = corrosive . co .
uk" ) ;
```

باوجودی‌که‌روش فوق جهت تنظیم کوکی‌های ارسالی به مرورگر اینترنت روش سختی محسوب نمی‌شود، اماهمان‌گونه که ملاحظه می‌کنید مستلزم سازماندهی تابعی با نام ( ) header و تأمین آرگومان مورد نیاز آن است؛ آن‌هم در قالب‌بندی که در قطعه برنامه فوق مشاهده می‌کنید. البته قالب‌بندی تاریخ به روشی که در این قطعه کد ملاحظه کردید و نیز تدارک نام و مقدار کوکی به‌صورت فوق را نمی‌توان کار دشواری تلقی نمود. با این حال روش فوق به‌دلیل اینکه در PHP توابع کارآمدی جهت انجام یک چنین قالب‌بندی‌هایی پیش‌بینی شده است، به‌نظر می‌رسد که کارایی لازم را در اختیار برنامه‌نویس قرار نمی‌دهد.

روش دیگری را که PHP جهت تنظیم کوکی‌ها پیش‌بینی کرده، استفاده از تابع ( ) setcookie است. تابع( ) setcookie همان‌گونه که از نامش پیداست منجر به تنظیم هدر set_ cookie از هدر HTTP ارسالی به مرورگر اینترنت مورد استفاده کاربر خواهد شد. از این‌رو لازم است تا فراخوانی این تابع پیش از ارسال هرگونه اطلاعاتی به مرورگر اینترنت صورت بگیرد. تابع مذکور نام کوکی، مقدار کوکی، تاریخ انقضای کوکی در قالب epoch (قالب متداول سیستم عامل UNIX جهت کار با تاریخ و ساعت)، اطلاعات مربوط به مسیر، نام حوزه و یک عدد صحیح را به‌عنوان آرگومان‌های ورودی دریافت می‌کند. عدد صحیح اخیر که به‌عنوان آخرین آرگومان تابع ( ) setcookie مورد استفاده قرار می‌گیرد در صورتی‌که کوکی تحت یک اتصال امن ارسال شود باید با مقدار عددی 1 تنظیم شود. لازم به ذکر است که تمامی آرگومان‌های تابع ( ) setcookie منهای اولین آرگومان آن، که بیانگر نام کوکی است اختیاری می‌باشند.

جهت بررسی بیشتر درمورد چگونگی استفاده از این تابع برنامه موجود در لیست ۳- ۱۹ را که از تابع فوق جهت تنظیم یک کوکی با نام vegetable استفاده می‌کند، درنظر بگیرید.

```
1: <?php
2: setcookie( "vegetable", "artichoke", time()+3600, "/",
3:            "corrosive.co.uk", 0 );
4: ?>
5: <html>
6: <head>
7: <title>Listing 19.1 Setting and printing a cookie value</title>
8: </head>
9: <body>
10: <?php
11: if ( isset( $vegetable ) )
12:     print "<p>Hello again, your chosen vegetable is $vegetable</p>";
13: else
14:     print "<p>Hello you. This may be your first visit</p>";
15: ?>
16: </body>
17: </html>
```

لیست ۳- ۱۹ استفاده از تابع ( ) setcookie جهت تنظیم کوکی

همان‌گونه که در این برنامه مشاهده می‌کنید فراخوانی تابع ( ) setcookie در خط ۲ موجب تنظیم کوکی به هنگام اجرای این برنامه اسکریپت برای اولین مرتبه خواهد شد. اما توجه کنید که در این مقطع متغیری با عنوان vegetable ایجاد نخواهد شد. اطلاعات موجود در درون کوکی تنها هنگامی مورد بازخوانی واقع می‌شود که برنامه مرورگر اینترنت آن‌را به برنامه سرور ارسال نماید. این اتفاق چنانکه اکنون می‌دانید تا زمانی‌که کاربر هیچ سندی واقع در مسیر مشخص شده در فیلد path را مورد دستیابی قرار ندهد، روی نمی‌دهد. همان‌گونه که در فراخوانی تابع ( ) setcookie در خط ۲ برنامه ملاحظه می‌کنید، نام کوکی و مقدار متناظر با آن به‌ترتیب با دنباله‌های کاراکتری "vegetable" و "artichocke" مشخص شده‌اند. همچنین برچسب زمان جاری با بهره‌گیری از تابع ( ) time ایجاد شده و عدد صحیح 3600 به آن اضافه شده است (3600 عدد معادل تعداد ثانیه‌های موجود در یک ساعت است). این مجموعه نماینده تاریخ انقضای کوکی ایجاد شده از طریق این آدرس می‌باشد. مقدار فیلد path که شامل دنباله کاراکتری " / " است، بدین معنی است که این کوکی درازای دستیابی کاربر به هر سندی که در مسیر فوق یا مسیرهای فرعی تحت آن (مثلاً / cgi_ bin / sample. cgi) واقع شده باشد، به برنامه سرور ارسال خواهد شد. همچنین مشاهده می‌کنید که مقدار فیلد domain با استفاده از دنباله کاراکتری "corrosive . co. uk" تنظیم شده است. این بدان معنی است که کوکی موردنظر به هر سروری که در گروه فوق یعنی corrosive . co. uk باشد، ارسال خواهد شد. برای مثال دو سرور با آدرسهای www. corrosive. co. uk و dev. corrosiv. co. uk از این‌گونه سرورها هستند. درصورتی‌که مایل باشید تا کوکی مورد بحث تنها و تنها به سروری که میزبان برنامه اسکریپت مورد درخواست است، ارسال شود به‌جای ذکر نام سرور به شیوه فوق می‌توانید از یک متغیر سیستمی ویژه با عنوان $SERVER_ NAME استفاده نمایید. منفعتی که از انجام چنین کاری نصیب ما می‌شود این است که برنامه مذکور حتی درصورت انتقال به یک سرور جدید نیز برخلاف روش قبل به‌خوبی کار می‌کند.

چنانکه در برنامه فوق مشاهده می‌کنید، به‌عنوان آخرین آرگومان تابع( ) setcookie از عدد صحیح صفر استفاده کرده‌ایم. این مقدار صحیح بدان معنی است که امکان ارسال کوکی از طریق یک اتصال ناامن نیز قابل انجام می‌باشد.

هرچند همان‌گونه که پیشتر نیز اظهار کردیم، غیر از اولین آرگومان تابع ( ) setcookie تعیین سایر آرگومان‌های این تابع به‌منظور تنظیم کوکی موردنظر امری اختیاری محسوب می‌شود اما همواره ارسال‌تمامی آرگومان‌های فوق (به استثنای فیلدهایdomain وآخرین آرگومان) را به تابع ( )setcookie توصیه‌می‌کنیم. این از آن جهت است که برخی از مرورگرهای اینترنت به‌منظور بهره‌مندی از قابلیتهای کوکی نیازمند آرگومان path می‌باشند. گذشته از این بدون وجود این آرگومان تنها به‌واسطه دستیابی به اسناد موجود در فهرست جاری و زیرفهرستهای مربوطه ارسال خواهد شد.

توجه‌کنیدکه‌بهره‌گیری از دنباله‌کاراکتری تهی (" ") به‌عنوان آرگومان‌هایی ازتابع ( ) setcookie که از نوع دنباله کاراکتری هستند و همچنین بهره‌گیری از مقدار عددی صفر به‌عنوان آخرین آرگومان این تابع که از نوع عدد صحیح است، باعث خواهد شد تا تابع مذکور این آرگومان‌ها را در فرآیند تنظیم کوکی نادیده بگیرد.

نحوه حذف کوکی

PHP علاوه بر تنظیم کوکی‌ها امکانی را نیز جهت حذف آنها در اختیار برنامه‌نویس قرارمی‌دهد. جالب این است که این ابزار همان ابزاری است که جهت تنظیم آن به‌کار می‌رود؛ برای حذف کوکی موردنظر کافی است تابع ( ) setcookie را فراخوانی کرده و نام کوکی را به‌عنوان تنها آرگومان در قالب یک دنباله کاراکتری به این تابع ارسال نمایید. بدین ترتیب فراخوانی زیر قادر خواهد بود تا کوکی ایجاد شده در لیست ۱- ۱۹ را حذف نماید:

```
setcookie ( " vegetable " );
```

با این حال فراخوانی تابع ( ) setcookie به روش فوق همواره نتیجه مطلوب را جهت حذف کوکی موردنظر تامین نمی‌کند و بدین جهت نباید کاملاً برروی آن تکیه کرد. مطمئن‌ترین روش حذف کوکی موردنظر این است که از آرگومان دیگری که نماینده تاریخ انقضای کوکی است، استفاده کنیم. کافی است تاریخ انقضای مذکور را قبل از تاریخ جاری تنظیم نماییم. بنابراین با فراخوانی زیر این اطمینان حاصل می‌شود که کوکی موردنظر با نام " vegetable " حذف خواهد شد.

```
setcookie (" vegetable", " ", time ( ) - 60, " / " , "corrosive. co. uk ", 0 ) ;
```

چنانکه ملاحظه می‌کنید تاریخ انقضای کوکی دقیقاً ۶۰ ثانیه یا یک دقیقه قبل از تاریخ فعلی تنظیم شده است. همچنین دقت کنید که در این فراخوانی از سایر آرگومان‌هایی که هنگام تنظیم کوکی مورد استفاده قرار گرفته‌اند، بهره‌برداری شده است. وجود این آرگومان‌ها ضریب اطمینان فرآیند

حذف کوکی را بالا می‌برد اما همواره باید دقت کنید که مقادیر آرگومان‌ها دقیقاً باید با مقادیری که هنگام تنظیم آن کوکی مورد استفاده قرار دادید، یکسان باشد.

ایجاد کوکی‌هایی با عمر کوتاه

در برخی موارد و با توجه به محدودیت تعداد کوکی‌های قابل تنظیم بر روی مرورگر به‌ازای هر سرور ممکن است کوکی‌هایی با عمر کوتاه مورد نیاز باشند. منظور از عمر کوتاه این است که کوکی مورد نظر طی یک جلسه‌ای که کاربر از طریق مرورگر خود با برنامه سرور برقرار می‌کند ایجاد شده و در پایان جلسه مذکور منقضی شود. به این نوع کوکی‌ها کوکی جلسه یا session cookie نیز اطلاق می‌شود (جلسه از زمانی آغاز می‌شود که کاربر آدرس URL وب سایت یا سند موردنظر خود را در فیلد آدرس مرورگر اینترنت خود وارد کرده و کلید Enter را بزند یا روی دکمه خاصی کلیک کند. همچنین جلسه ممکن است با کلیک کاربر روی فرا پیوندی که منتهی به موقعیتی از وب سایت می‌شود، آغاز گردد.هرجلسه ایجاد شده با بسته شدن پنجره مرورگر اینترنت خاتمه می‌یابد). برای ایجاد کوکی جلسه کافی است تا از مقدار صفر به‌عنوان آرگومان تعیین کننده تاریخ انقضای کوکی در تابع setcookie( ) استفاده کنیم. مادامی که پنجره مرورگر اینترنت توسط کاربر یا به هر علت دیگری بسته نشده باشد، کماکان جلسه به قوت خود باقی بوده و بنابراین کوکی ایجاد شده به سرور مربوطه ارسال خواهد شد. با این حال توجه کنید که مرورگر اینترنت پس از خروج (بسته شدن پنجره مربوطه) و راه‌اندازی مجدد، (بازشدن مرورگر جدید) هیچ چیزی را درمورد یک چنین کوکی به‌خاطر نمی‌آورد.

علی‌رغم کوتاه بودن طول زندگی این نوع کوکی‌ها می‌توان استفاده‌های مفید و جالب توجهی از آنها نمود. یکی از موارد استفاده این نوع کوکی‌ها اعتبارسنجی کاربرانی است که از وب سایت مربوطه استفاده می‌کنند. کاربر پس از وارد کردن نام و کلمه عبوری که پیشتر در جلسات قبلی تعیین نموده است، خود را معرفی می‌کند. کوکی جلسه بدین ترتیب با مشخص نمودن آن کاربر به‌عنوان یک کاربر معتبر طی زمانی که جلسه به قوت خود باقی است، باعث می‌شود تا برنامه‌های اسکریپت موجود بتوانند اسناد و سرویسهای معتبر و مشخصی را دراختیار وی قرار دهند (انعکاس تنظیمات شخصی کاربر مثلاً درموردرنگهای استفاده شده در صفحه وب و یا اخباری را که مربوط به یک رویداد یا منطقه جغرافیایی خاصی هستند از این جمله سرویسها محسوب می‌شوند). به‌هرحال پس از اتمام جلسه (بسته شدن پنجره مرورگر اینترنت) هیچ لزومی ندارد که این سرویسهای شخصی دوام داشته باشند چراکه به‌دلیل ویژگی خاص قرار داد مابین برنامه مرورگر اینترنت و برنامه وب سرور که HTTP نام دارد (چنانکه در ابتدای درس نیز عنوان کردیم این ویژگی خاص با عنوان stateless شناخته می‌شود)، نمی‌توان مطمئن شد که جلسه جدید که با باز شدن پنجره جدید مرورگر آغاز می‌شود توسط همان کاربر قبلی که اخیراً جلسه قبلی را پایان داده ایجاد شده است، مگر آنکه نام کاربر و کلمه عبوری که به وی اختصاص داده

شده است، مجدداً در جلسه جدید مورد بازیابی قرار بگیرد. فراخوانی تابع ( ) setcookie به‌صورت زیرمنجر به ایجاد یک کوکی جلسه با عنوان session_ id خواهد شد:

```
setcookie (" session _ id ", " 55435 ", 0 ) ;
```

بررسی یک مثال نمونه در مورد ردیابی کم و کیف بهره‌گیری کاربر از سایت

فرض کنید به‌عنوان یک برنامه‌نویس وب از شما خواسته شده است تا با بهره‌گیری از کوکی‌ها و همچنین‌بانک اطلاعاتی MySQL اطلاعاتی را در مورد نحوه استفاده کاربران از یک وب سایت بخصوص جمع‌آوری نموده و آنها را به‌منظور تصمیم‌گیری‌های آتی درمورد تعیین سیاستها در اختیار مدیران مربوطه قراردهید. یک هدف از این پی‌گیری تعیین تعداد کاربرانی است که از وب سایت مورد بازدید به‌عمل آورده‌اند.اهداف‌دیگرعبارت‌ازتعیین میانگین تعداد کلیک‌هایی است که هر کاربر طی بازدید خود ازوب‌سایت‌صورت‌می‌دهد و بالاخره هدف نهایی تعیین زمان میانگین بازدید هر کاربر از وب سایت است.

پیش از انجام هر اقدامی ابتدا لازم است تا توضیحی قانع‌کننده و جامع در مورد محدودیت عملکرد کوکی‌ها به کارفرمای خود بدهید. به‌عنوان نکته اول می‌توانید چنین بیان کنید که همه کاربران وب سایت ممکن است گزینه مربوط به بهره‌گیری از کوکی‌ها را به‌گونه‌ای که مدنظرمان است، فعال نکرده باشند (برخی کاربران حتی ترجیح می‌دهند تا بنا به دلایل شخصی یا امنیتی این ویژگی را غیر فعال نمایند). درصورتی که کوکی از طریق یک مرورگر اینترنت به سرور ارسال نشود، برنامه‌ای که مسئول کنترل فرآیند است چنین فرض خواهد کرد که کاربر مربوطه اولین بازدید خود از وب سایت را انجام می‌دهد. در این حالت شاید حتی بتوان چنین نتیجه گرفت که مرورگر اینترنت مورد استفاده کاربر توانایی پشتیبانی از کوکی‌ها را ندارد. از این گذشته نمی‌توان اطمینان داشت که کاربر مورد نظر جهت انجام بازدیدهای خود همواره از یک مرورگر خاص استفاده می‌کند و بالاخره نمی‌توان از این موضوع نیز اطمینان حاصل کرد که مرورگر اینترنت مورد استفاده کاربر ما توسط سایر کاربران نیز مورد بهره‌برداری قرار می‌گیرد یا خیر.

با تذکر این نکات اکنون می‌توانیم کار خود را برای ایجاد اجزای برنامه خواسته شده انجام دهیم. حقیقت این است که می‌توانیم با کمتر از ۹۰ خط کد نویسی قابلیتهای خواسته شده را پیاده‌سازی کنیم.

پیش از هر چیزی لازم است تا یک بانک اطلاعاتی جهت ذخیره وبازیابی داده‌ها و اطلاعات موردنیازمان ایجاد نماییم. فیلدهای مورد نیاز از جدول موجود در این بانک اطلاعاتی به شرحی است که در جدول ۱- ۱۹ مشاهده می‌کنید:

نام فیلد	نوع داده فیلد	توضیح
id	عدد صحیح	این فیلد شامل عدد صحیحی است که به‌ازای هر بازدیدکننده از وب سایت، عدد منحصر به‌فردی را به‌عنوان شناسه کاربر مشخص می‌کند. مقدار این فیلد به‌ازای هر کاربر جدید به‌طور خودکار یک واحد افزایش می‌یابد.
first_ visit	عدد صحیح	این فیلد یک برچسب زمان است و موعدی که کاربر اولین صفحه موردنظر خود از وب سایت مورد درخواست قرار داده است را مشخص می‌کند.
last_ visit	عدد صحیح	این فیلد نیز به‌مانند فیلد قبلی یک برچسب زمان بوده و موعدی که کاربر صفحه جاری (آخرین صفحه) از وب سایت را مورد درخواست قرار داده است را مشخص می‌کند.
num_visits	عدد صحیح	این فیلد نماینده تعداد جلساتی است که بازدیدکننده اقدام به برقراری آن نموده است.
total_ duration	عدد صحیح	این فیلد نماینده مدت زمانی برحسب ثانیه است که بازدیدکننده صرف بازدید از سایت نموده است.
total_ clicks	عدد صحیح	این فیلد نماینده تعداد درخواستهایی است که بازدید کننده جهت بازدید از اسناد مختلف وب سایت راهی سرور نموده است.

جهت ایجاد جدول موردنظرمان در درون بانک اطلاعاتی MySQL لازم است تا از عبارت CREATE استفاده کنیم. چگونگی ایجاد این جدول که با نام track_ visit مشخص شده است در لیست ۴- ۱۹ آمده است.

```
CREATE TABLE track_visit(
            id INT NOT NUUL AUTO_INCREMENT,
            PRIMARY KEY( id ),
            first_visit INT,
            last_visit INT,
            num_visits INT,
            total_duration INT,
            total_clicks INT
            );
```

لیست ۴- ۱۹ بهره‌گیری از عبارت CREATE جهت ایجاد جدول موردنظر از بانک اطلاعاتی

اکنون که جدول موردنیاز خود را جهت ذخیره اطلاعات موردنظرمان در اختیار داریم وقت آن است تا کد برنامه‌ای را که جهت برقراری اتصال با بانک اطلاعاتی و بررسی وجود کوکی مورد نیاز است، توسعه دهیم. درصورتی‌که کوکی موردنظرمان موجود نباشد، لازم است تا سطر (رکورد) جدیدی را در جدول مذکور ایجاد کرده و فیلدهای موجود در آن‌را با مقادیر اولیه مقداردهی نماییم. لیست ۵- ۱۹ برنامه‌ای را که چنین فرآیندی برای ما انجام می‌دهد، نشان داده است.

```
<?php
$link = connect( "localhost", "", "", "test" );

if ( ! isset( $visit_id ) ) {
    newuser( );
    print "Welcome, first time user!";
} else {
    print "Welcome back $visit_id<P>";
}

function newuser( ) {
    $visit_data = array (
                'first_visit' => time(),
                'last_visit' => time(),
                'num_visits' => 1,
                'total_duration' => 0,
                'total_clicks' => 1
                );

    insert_visit( $visit_data );
    setcookie( "visit_id", $visit_data['id'],
               time()+(60*60*24*365*10), "/" );
    return $visit_data;
}

function connect( $host, $user, $pass, $db ) {
    $link = mysql_connect( $host, $user, $pass ) or
        die("Connection error");
    mysql_select_db( $db, $link ) or die ( mysql_error() );
    return $link;
}

function insert_visit( &$visit_data ) {
    global $link;
    $query  = "INSERT INTO track_visit ( ";
    $query .= implode( ", ", array_keys( $visit_data ) );
    $query .= " ) VALUES( ";
    $query .= implode(", ", array_values( $visit_data ) );
    $query .= " );";
    $result = mysql_query( $query, $link );
    $visit_data['id'] = mysql_insert_id();
}
?>
```

لیست ۵- ۱۹ برنامه مورد نیاز جهت اضافه کردن اطلاعات کاربر جدید به بانک اطلاعاتی موجود

چنانکه در این لیست مشاهده می‌کنید، با بهره‌گیری از روش متداول برقراری اتصال با بانک یعنی استفاده از تابعی با نام ( ) connect که تعریف آن‌را در خط ۲۶ ارائه داده‌ایم، اقدام به برقراری اتصال با بانک اطلاعاتی MySQL موردنظر خود نموده‌ایم. فراخوانی تابع ( ) mysql_ connect در خط ۲۷ به‌عنوان بخشی از تعریف تابع ( ) connect منجر به برقراری این اتصال خواهد شد. همچنین فراخوانی تابع ( ) myqle_ select_ db در خط ۲۹ از این برنامه، باز هم به‌عنوان بخشی از تعریف تابع ( ) connect موجب انتخاب بانک اطلاعاتی موردنظرمان که شامل جدول تعریف شده در لیست ۴- ۱۹ است، خواهد شد (جهت مطالعه مطالب مربوط به چگونگی کار با بانک اطلاعاتی MySQL به مباحثی که در درس ساعت دوازدهم مورد بررسی قرار دادیم مراجعه نمایید). تابع ( ) connect به‌گونه‌ای طراحی شده است که به‌عنوان نتیجه عملیات خود مرجعی به یک بانک اطلاعاتی را به برنامه فراخواننده باز می‌گرداند. همان‌گونه که ملاحظه می‌کنید این مرجع نهایتاً در یک متغیر سراسری با نام $link ذخیره می‌شود. از این‌رو متغیر نامبرده در دسترس کلیه توابعی از برنامه که با بانک اطلاعاتی موردنظر کار می‌کنند، قرار خواهد داشت. در خط ۴ برنامه با بهره‌گیری از تابع ( ) isset وجود متغیری با نام $visit_ id مورد بررسی قرار می‌گیرد. این متغیر درصورت وجود شامل نام کوکی خواهد بود که نماینده شناسه یک کاربر (بازدید کننده) می‌باشد. چنانچه متغیری با این نام در برنامه موجود نباشد فرض ما بر این خواهد بود که کاربر مورد نظر یک کاربر جدید بوده و برای اولین مرتبه از وب سایت بازدید به‌عمل می‌آورد. در چنین حالتی تابعی با نام ( ) newuser جهت مقداردهی عوامل و فیلدهای مربوط به این کار جدید از جدول موجود در بانک اطلاعاتی فراخوانی می‌شود.

همان‌گونه که مشاهده می‌کنید تابع ( ) newuser در خط ۱۱ از برنامه تعریف شده است. این تابع جهت انجام عملیات پیش‌بینی شده به هیچ آرگومانی نیاز نداشته و از این جهت تابع ساده‌ای محسوب می‌شود. تابع فوق در مقابل آرایه‌ای را به‌عنوان نتیجه عملیات خود به برنامه فراخواننده باز می‌گرداند. این آرایه شامل مقادیر فیلدهای جدول تعریف شده در قسمت قبل خواهد بود. در خط ۱۲ از برنامه و در بخش مربوط به تعریف تابع ( ) newuser ، آرایه‌ای با نام $visit _ data ایجاد شده است. مقادیر عناصر first _ visit و last _ clicks از این آرایه با مقدار تاریخ جاری و برحسب تعداد ثانیه‌های سپری شده از مبدا زمانی epoch که بارها راجع به آن توضیح دادیم، مقداردهی شده‌اند. از آنجا که تابع ( ) newuser به‌واسطه اولین بازدید کاربر از وب سایت فراخوانی می‌شود، مقادیر عناصر num_ visit و total_ clicks از آرایه مورد بحث با مقدار عددی 1 مقداردهی شده‌اند و به‌دلیل اینکه هیچ زمانی به‌واسطه اولین بازدید سپری نشده است، عنصر total_ duration با مقدار عددی صفر مقداردهی می‌شود (فیلد اخیر بیانگر مدت زمانی است که بازدیدکننده صرف بازدید از صفحات موجود در وب سایت می‌کند).

در خط ۲۰ از برنامه مورد بحث تابعی با نام ( ) insert_ visit که تعریف آن در خط ۳۳ آمده است، مورد فراخوانی واقع می‌شود. این تابع ساختار ساده‌ای داشته به‌گونه‌ای که تنها از یک آرگومان ورودی بهره می‌برد. تابع مذکور با بهره‌گیری از عناصر آرایه‌ای که به‌عنوان آرگومان ورودی دریافت می‌کند اقدامی جهت ایجاد یک سطر جدید در اول جدول صورت داده و هریک از فیلدهای سطر مذکور را با مقدار موجود در عنصری از آرایه که همنام با آن است، مقداردهی می‌کند. چنانکه ملاحظه می‌کنید ما درخط ۳۶ از برنامه به‌عنوان بخشی از تعریف تابع ( ) insert_ visit جهت ایجاد عبارت SQL موردنظرمان از تابع سیستمی ویژه‌ای با نام ( ) implode استفاده کرده‌ایم. از آنجا که مقدار فیلد id از جدول ما به‌ازای ورود هر سطر جدید در این جدول به‌طور خودکار به میزان یک واحد افزایش پیدا می‌کند، نیازی به واردکردن دستی آن به جدول نیست. بدین ترتیب همان‌گونه که عبارت موجود در خط ۴۱ از برنامه نیز نشان می‌دهد به‌راحتی می‌توانیم مقدار موجود در این فیلد را با فراخوانی تابع دیگری با نام ( ) insert_ id _ mysql مورد دستیابی قرار دهیم. اکنون با تخصیص یک شناسه منحصر به فرد به بازدیدکننده از وب سایت می‌توانیم آن شناسه را به آرایه $visit _ data اضافه نماییم. ما از این عنصر آرایه جهت بازیابی اطلاعات مربوط به بازدیدکنندگان مختلف از جدول track _ visit از بانک اطلاعاتی استفاده خواهیم کرد.

از آنجا که ارسال آرایه $visit _ data به‌عنوان آرگومان به تابع ( ) insert _ visit از طریق مرجع صورت گرفته است، هرگونه دستکاری و اعمال تغییر در مقادیر ذخیره شده در عناصر موجود در این آرایه از طریق متغیر همنام با آن در فراخوانی تابع ( ) newuser در اختیار تابع( ) insert _ visit قرار می‌گیرد.

به‌عنوان آخرین نکته، در خط ۲۱ از برنامه مورد بحث در درون تابع ( ) newuser تابع دیگری با نام ( ) setcookie که در قسمتهای پیشین مورد بحث قرارگرفت، فراخوانی می‌شود. این تابع چنانکه ملاحظه می‌کنید یک کوکی با نام visit_ data_ را به‌عنوان نتیجه عملیات به برنامه فراخواننده باز می‌گرداند.

تحت یک چنین شرایطی که مشاهده نمودید دفعه بعدی که بازدیدکننده ما به‌واسطه ارسال درخواست خود به سرور منجر به اجرای این برنامه اسکریپت شود، عبارت شرطی if موجود در خط ۴ به‌صورت false ارزیابی می‌شود: چراکه متغیر $visit_ id تعیین می‌شود و به‌دلیل اینکه متغیر مذکور حاوی یک مقدار می‌باشد، تنها اتفاقی که روی می‌دهد این است که بازدیدکننده با پیغام خوش‌آمدگویی برنامه مواجه شده و هیچ‌گونه فرآیند دیگری در رابطه با فراخوانی توابع مختلف یا مقداردهی آرایه‌ها و یا تعریف یک کاربر جدید انجام نمی‌شود.

حقیقت این است که به‌محض مشاهده یک بازدیدکننده قدیمی (بازدیدکننده‌ای که پیشتر به‌واسطه اولین بازدید وی از وب سایت سطری شامل اطلاعات مربوط به وی در درون جدول ایجاد شده

است) لازم است تا اطلاعات موجود در سطر مربوط به وی از جدول visit _ track به‌روز رسانی شود. برای‌انجام‌این‌عمل‌لازم‌است‌تا یک بررسی در مورد اینکه آیا درخواست جاری بازدیدکننده جهت مشاهده سندی از وب سایت اولین درخواست وی در جلسه کاری جدید است یا به‌سادگی یکی از درخواستهای او را طی جلسه مزبور تشکیل می‌دهد. برای پی بردن به این مطلب مهم از یک متغیر سراسری که حاوی عدد صحیح معادل با یک زمان می‌باشد، استفاده خواهیم نمود. درصورتی‌که مجموع زمان مذکور بازمان‌مربوط به آخرین درخواست بازدیدکننده بزرگ‌تر از زمان جاری باشد، می‌توان چنین نتیجه گرفت که درخواست جاری تنها یکی از درخواستهای بازدیدکننده طی این جلسه کاری است. اما درحالتی که مجموع دو زمان فوق کوچک‌تر از زمان جاری باشد، چنین برداشت می‌کنیم که این درخواست کاربر قدیمی ما اولین درخواست وی طی این جلسه کاری جدید بوده و از این‌رو برنامه پیغام خوش‌آمدگویی مجددی را برای او نمایش خواهد داد.

برنامه موجود در لیست ۶- ۱۹ جهت پیاده‌سازی قابلیتهای مذکور، کد برنامه موجود در لیست ۵- ۱۹ را به‌گونه‌ای متناسب با نیازهای دستخوش تغییر کرده است.

```
1:<?php
2:$slength = 300;
3:$link = connect( "localhost", "", "", "test" );
4:$user_stats;
5:if ( ! isset( $visit_id ) ) {
6:    $user_stats = newuser( );
7:    print "Welcome, first time user!";
} else {
    print "Welcome back $visit_id<P>";
    $user_stats = olduser( $visit_id );
}

function newuser( ) {
    $visit_data = array (
                'first_visit' => time(),
                'last_visit' => time(),
                'num_visits' => 1,
                'total_duration' => 0,
                'total_clicks' => 1
                );

    insert_visit( $visit_data );
    setcookie( "visit_id", $visit_data['id'],
               time()+(60*60*24*365*10), "/" );
    return $visit_data;
}
function olduser( $visit_id ) {
    global $slength;
    $now = time();
```

```
    $visit_data = get_visit( $visit_id );
    if ( ! $visit_data )
        return newuser( );
    $visit_data['total_clicks']++;
    if ( ( $visit_data['last_visit'] + $slength ) > $now )
        $visit_data['total_duration'] +=
                    ( $now - $visit_data['last_visit'] );
    else
        $visit_data['num_visits']++;

    $visit_data['last_visit'] = $now;
    update_visit( $visit_data );
    return $visit_data;
}
function connect( $host, $user, $pass, $db ) {
    $link = mysql_connect( $host, $user, $pass ) or
        die("Connection error");
    mysql_select_db( $db, $link ) or die ( mysql_error() );
    return $link;
}

function get_visit( $visit_id ) {
    global $link;
    $query = "SELECT * FROM track_visit WHERE id=$visit_id";
    $result = mysql_query( $query, $link );

    if ( ! mysql_num_rows( $result ) )
        return false;
    return mysql_fetch_assoc( $result, $link );
}

function update_visit( &$visit_data ) {
    global $link;
    $update_pairs = array();
    foreach( $visit_data  as $field=>$val )
        if ( ! is_int( $field ) )
            array_push( $update_pairs, "$field=$val" );
    $query  = "UPDATE track_visit SET ";
    $query .= implode( ", ", $update_pairs );
    $query .= " WHERE id=".$visit_data['id'];
    mysql_query( $query, $link );
}

function insert_visit( &$visit_data ) {
    global $link;
    $query  = "INSERT INTO track_visit ( ";
    $query .= implode( ", ", array_keys( $visit_data ) );
    $query .= " ) VALUES( ";
```

```
    $query .= implode(", ", array_values( $visit_data ) );
    $query .= " );";
    $result = mysql_query( $query, $link );
    $visit_data['id'] = mysql_insert_id();
}
?>
```

لیست ۶- ۱۹ برنامه کامل ردیابی بازدیدکنندگان از وب سایت که از یک بانک اطلاعاتی MySQL نیز بهره می‌برد

چنانکه در این لیست مشاهده می‌کنید در خط ۲ از برنامه متغیر سراسری جدیدی را به عنوان $slength مورد استفاده قرار داده‌ایم. از این متغیر جدید جهت ثبت فاصله زمانی کوتاهی بعد از این فرض که یک بازدید جدید از وب سایت در حال انجام است، استفاده به‌عمل می‌آوریم. درصورتی‌که متغیر $visit _ id در برنامه موجود باشد، متوجه می‌شویم که کوکی موردنیاز جهت پیگیری بازدیدکننده موردنظر در بازدید از وب سایت موجود می‌باشد. با این فرض، تابع ( ) olduser را در خط ۱۰ برنامه فراخوانی کرده و متغیر $visit_ id را به‌عنوان آرگومان به آن ارسال می‌کنیم.

همان‌گونه که در تعریف تابع ( ) olduser مشاهده می‌کنید ما ابتدا اطلاعات موردنیاز درمورد بازدید کاربراز وب سایت را از طریق فراخوانی تابع ( ) get_ visit در خط ۳۱ برنامه مورد دستیابی قرار داده‌ایم. تابع ( ) get_ visit در خط ۵۳ از برنامه تعریف شده‌است. این تابع به‌عنوان تنها آرگومان موردنیاز شناسه منحصر به فرد بازدیدکننده که آرگومانی با نام visit_ id ذخیره شده است را دریافت می‌کند. از این آرگومان ورودی تابع موردنظر با بهره‌گیری از تابع ویژه‌ای با نام ( ) mysql_ query در خط ۵۶ برنامه اقدام به استخراج سطر مربوطه از جدول track_ visit از بانک اطلاعاتی می‌کند. با این فرض که توانسته باشیم سطر موردنظر از جدول track_ visit را که با کوکی visit_ id متناظر است پیدا کنیم، در خط ۶۰ از برنامه تابع ( ) mysql_ fetch_ assoc را فرا می‌خوانیم. این تابع آرایه‌ای با نام $visit_ data را که یک آرایه انجمنی است با اسامی و مقادیر فیلدهای موجود در سطر یاد شده از جدول مذکور مقداردهی می‌کند. تابع ( ) olduser بدین ترتیب اکنون باید به آرایه مقداردهی شده $visit_ data دسترسی داشته باشد. درغیر این‌صورت فرض وجود اطلاعات مربوط به بازدیدکننده در جدول نادیده گرفته شده و همان‌گونه که در خط ۳۳ از برنامه مشاهده می‌کنید به‌واسطه فراخوانی تابع ( ) newuser، اقدامی جهت اضافه‌کردن یک سطر جدید از اطلاعات به جدول track_ visit از بانک اطلاعاتی صورت می‌پذیرد.

اما در خط ۳۵ برنامه یک بررسی در مورد زمان صورت می‌گیرد. طی این بررسی مشخص می‌شود که آیا مجموع مقدار ذخیره شده در عنصر [' last_ visit '] data _ $visit از آرایه با زمان ذخیره شده در متغیر $slength از زمان جاری بزرگ‌تر است یا خیر. درصورتی که مجموع این دو زمان از زمان جاری بزرگ‌تر باشد بدین معنی خواهد بود که مدت زمان سپری شده از زمانی که بازدیدکننده

آخرین درخواست خود از طریق کلیک فراپیوند مربوطه را به سرور ارسال کرده است، نسبت به مقدار زمانی ذخیره شده در متغیر $slength کوچک‌تر می‌باشد. بدین ترتیب می‌توان چنین نتیجه‌گیری کرد که این درخواست تنها یکی از درخواستهای ارسالی به سرور طی این جلسه کاری می‌باشد. به همین منظور در خط ۳۶ برنامه مدت زمان سپری شده از زمان آخرین درخواست بازدیدکننده به مقدار موجود در عنصر [' total _ duration '] $visit_ datd از آرایه اضافه می‌شود.

درصورتی‌که کاربر اولین بازدید خود را از وب سایت طی این جلسه کاری انجام می‌دهد با بهره‌گیری از عبارت موجود در خط ۳۹ برنامه مقدار موجود در عنصر[' num_ visit '] $visit_ datd از آرایه دچار افزایش می‌شود.

در نهایت آرایه $visit _ data به‌عنوان آرگومان تابع ( ) update_ visit در خط ۴۲ برنامه مورد استفاده قرار می‌گیرد. چنانکه ملاحظه می‌کنید تعریف تابع ( ) update_ visit در خط ۶۳ از این برنامه انجام شده است. از این تعریف پیداست که تابع مورد بحث با بهره‌گیری از یک ساختار تکرار برمبنای مقادیر تغییر یافته از آرایه $visit_ data اقدام به ایجاد عبارت UPDATE می‌کند. این عبارت UPDATE در خط ۷۲ برنامه به‌عنوان آرگومان تابع ( ) mysql_ query جهت به‌روز رسانی اطلاعات مربوطه به بازدیدکننده (که در جدول track _ visit از بانک اطلاعاتی نگهداری می‌شود) مورد استفاده قرار می‌گیرد. تابع ( ) olduser به‌عنوان نتیجه عملیات خود، آرایه $visit_ data را که اکنون دستخوش تغییر شده است به برنامه فراخواننده بازمی‌گرداند.

اکنون که کار توسعه کد مورد نظر جهت ردیابی فعالیتهای بازدیدکنندگان در وب سایت به اتمام رسیده است، وقت آن است که با بهره‌گرفتن از یک تابع عملکرد برنامه خود را در عمل مورد بررسی و ارزیابی قرار دهیم. برای این منظور تابعی با نام ( ) outputstats را ایجاد خواهیم کرد. این تابع به‌سادگی مقادیر میانگین اطلاعات بازدیدکننده جاری را مورد محاسبه قرارداده و نتیجه حاصل از این فرآیند را به‌عنوان خروجی بر روی پنجره مرورگر اینترنت ارسال خواهد کرد. در دنیای واقع هنگام توسعه برنامه‌های واقعی به‌احتمال قوی مایل خواهید بود تا اطلاعاتی را در این مورد پیش روی کاربران خود قرار دهید. برنامه موجود در لیست ۷- ۱۹ کد مربوط به تابع ( ) outputstats را نشان می‌دهد. ضمن اینکه کد برنامه مثال قبل با بهره‌گیری از تابع ( ) include در این برنامه شامل شده است.

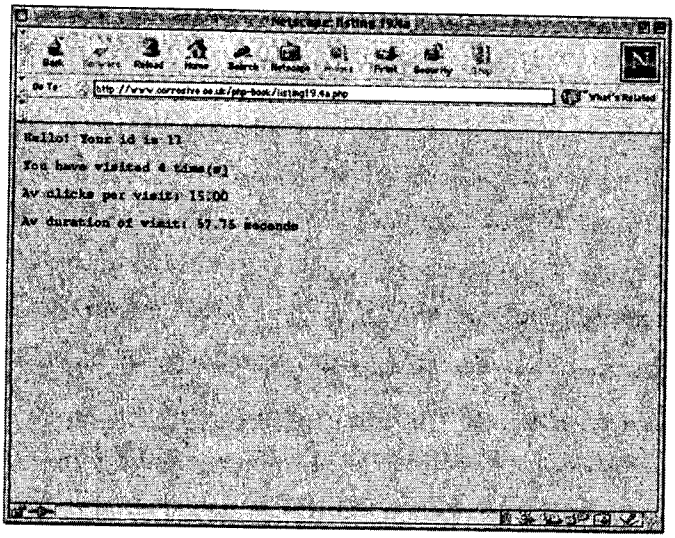
```
1:<?php
2:include("listing19.7.php");
3:outputStats();
4:function outputStats() {
5:    global $user_stats;
6:    $clicks = sprintf( "%.2f",

7:($user_stats['total_clicks']/$user_stats['num_visits']) );
8:    $duration =  sprintf( "%.2f",
9:($user_stats['total_duration']/$user_stats['num_visits']));
```

```
10:   print "<p>Hello! Your id is
".$user_stats['id']."</p>\n\n";
11:   print "<p>You have visited
12:         ".$user_stats['num_visits']." time(s)</p>\n\n";
13:   print "<p>Av clicks per visit: $clicks</p>\n\n";
14:    print "<p>Av duration of visit: $duration
seconds</p>\n\n";
15:}
16:?>
```

لیست ۷- ۱۹ برنامه‌ای جهت نمایش اطلاعات آماری جمع‌آوری شده توسط برنامه لیست ۶- ۱۹

چنانکه در لیست فوق مشاهده می‌کنید، در خط ۲ از برنامه تابع ( ) include را فراخوانی کرده‌ایم. این تابع باعث می‌شود تا کد مربوط به ردیابی عملکردهای بازدیدکنندگان در سایت وب که پیشتر در برنامه‌ای تحت عنوان listing19. 3. php آن را توسعه دادیم، مورد فراخوانی واقع شود. جهت جمع‌آوری اطلاعات به‌صورت کامل از فراخوانی‌های مشابهی به‌ازای هریک از صفحات موجود در سایت کارفرمای خود استفاده خواهیم نمود. تابع ( ) outputStats در خط ۳ از برنامه فراخوانی شده است. تعریف این تابع همان‌گونه که مشاهده می‌کنید در خط ۴ از این برنامه ارائه شده است. این تابع جهت انجام عملیات خود از متغیر سراسری $user_ stats که شامل یک آرایه می‌باشد، بهره می‌گیرد. آرایه مذکور به‌واسطه فراخوانی یکی از دو تابع ( ) newuser (برای بازدیدکنندگان جدید) یا ( ) olduser (برای بازدیدکنندگان قدیمی) در دسترس قرار می‌گیرد. این آرایه دقیقاً شامل اطلاعاتی است که در فیلدهای متناظر با عناصر آرایه فوق به‌ازای هر یک از بازدیدکنندگان در جدول visit _track از بانک اطلاعاتی نگهداری می‌شود. خروجی حاصل از اجرای این برنامه به‌ازای بازدیدکننده‌ای با شماره شناسه 17 در شکل ۱- ۱۹ قابل بررسی و مشاهده است.



شکل ۱- ۱۹ گزارش آماری استفاده از وب سایت

در خط ۶ از برنامه لیست ۷- ۱۹ چنانکه مشاهده می‌کنید جهت محاسبه تعداد دفعاتی که به‌طور میانگین بازدیدکننده برروی فراپیوندهای مختلف موجود در صفحات وب سایت کلیک کرده است، مقدار ذخیره شده در عنصر $user_ stats [' total _ clicks '] را به تعداد دفعات بازدید وی تقسیم کرده‌ایم. همچنین در خط ۸ از برنامه مذکور به‌طور مشابه مقدار ذخیره شده در عنصری از آرایه با عنوان $user_ stats [' total _ duration '] را به همان عامل، یعنی تعداد دفعات بازدید کاربر تقسیم کرده‌ایم. همان‌گونه که مشاهده می‌کنید جهت قالب‌بندی نتیجه حاصل از این تقسیم که یک عدد اعشاری است، از تابع ( ) sprint استفاده کرده‌ایم. شاخص % . 2f در هر دو مورد فوق بدین معنی است که عدد اعشاری حاصل از این تقسیم تا دو رقم اعشار گرد می‌شود. آنچه که باقی می‌ماند انعکاس آمار به‌دست آمده بر روی صفحه مرورگر اینترنت است که آن‌هم با بهره‌گیری از چند تابع ( ) print صورت می‌پذیرد.

علاوه بر این اطلاعات آماری که در این مثال مشاهده نمودید درصورت تمایل می‌توانیم اطلاعات دیگری را نیز به‌واسطه عملکرد بازدیدکننده در مواجهه با صفحات به دست آوریم. برای مثال علاقمندی‌های بازدیدکننده (مواردی که کاربر بیشتر به مشاهده آنها تمایل نشان داده است)، همچنین ثبت نوع مرورگر اینترنت مورد استفاده وی و نیز آدرس IP تخصیص یافته به وی از سوی ISP مربوطه از جمله چنین اطلاعاتی هستند. تصور کنید که وب سایت ما بتواند به‌طور هوشمند (با توجه به اطلاعات آماری به دست آمده) گشت و گذار بازدیدکنندگان خود را مورد تحلیل قرارداده و متناسب با علاقه‌مندی‌های هریک از آنها محتویات و اطلاعاتی را که در بخشهای مختلف سایت مدفون شده‌اند در اختیار آنها قرار دهد.

بهره‌گیری از دنباله‌های پرس و جو

یکی از نقاط تاریک قابل توجه در استفاده از کوکی‌ها وابستگی بسیار زیاد آنها به تصمیمات و سیاست‌هایی است که کاربران مختلف به‌گونه‌های متفاوت اتخاذ می‌کنند. علاوه براین وابستگی شدید کوکی‌ها به کاربران، کسانی ممکن است حتی مانع از عملکرد کوکی بر روی کامپیوترشان شوند، موضوع دیگر در رابطه با کوکی‌ها این است که علیرغم پذیرش آنها به‌عنوان ابزاری استاندارد باید به پیاده‌سازی آنها توسط مرورگرهای مختلفی که تعدادشان نیز روز به روز در حال تزاید است، اعتماد کنید. برخی از شرکتهای ارائه کننده محصول فوق، اشکالاتی را که مرورگرهایشان هنگام کار با کوکی‌ها با آنها برخورد می‌کنند، مستند سازی می‌نمایند و این اطلاعات در اختیار کاربرانی که از این نوع مرورگرها استفاده می‌کنند قرار می‌دهند. اگر هدف شما تنها ثبت وضعیت مربوط به یک جلسه تنها باشد به‌احتمال زیاد استفاده از روش قدیمی‌تری را که جهت این کار ابداع شده است، ترجیح خواهید داد. این روش که در این قسمت به شرح آن می‌پردازیم شامل استفاده از دنباله‌ای موسوم به دنباله پرس و جو است.

هنگامی که فرمی را با بهره‌گیری از روش GET به سرور ارسال می‌کنید فیلدها و مقادیر متناظر با هریک از آنها به طریق خاصی کدگذاری شده و به آدرس URL مقصد یا به‌عبارت دیگر به موقعیتی که به آن ارسال می‌شوند، ضمیمه می‌گردند. بدین ترتیب این فیلدها و مقادیر مربوط به آنها جهت انجام پردازشهای گوناگون در اختیار سرور و همچنین برنامه اسکریپت قرار خواهند گرفت. برای روشن شدن مطلب فرض کنید فرمی را که شامل دو فیلد با عناوین user_ id و name است به سرور ارسال کرده‌ایم. در این صورت دنباله پرس و جویی که به‌واسطه این فرآیند به انتهای آدرس URL مقصد ضمیمه می‌شود مشابه با آن چیزی خواهد بود که در اینجا مشاهده می‌کنید:

http: // www. corrosive. co. uk / test5. php? name = 344343 & user _ id = matt + zandstra

چنانکه در URL فوق مشاهده می‌کنید، هریک از اسامی فیلدها و مقدار متناظر با آنها با بهره‌گیری از علامت تساوی (=) از یکدیگر جدا شده‌اند. همچنین توجه کنید که هرجفت نام و مقدار با استفاده از علامت & از جفت نام و مقدار دیگر تفکیک شده است. علاوه براین دقت کنید که چگونه نام و نام خانوادگی matt Zandstra با استفاده از علامت + به یکدیگر الحاق شده‌اند. به‌واسطه وجود این علایم معمولاً چنین گفته می‌شود که دنباله‌های پرس و جو در قالبی " کد گذاری شده" به انتهای آدرس URL ضمیمه می‌شوند. واضح است که پیش از آنکه برنامه اسکریپت موردنظر بتواند از این مقادیر ارسالی جهت انجام پردازشهای پیش‌بینی شده خود استفاده نماید، لازم است تا به‌روشی دنباله پرس و جوی ضمیمه شده را " کد گشایی" نماید. خوشبختانه PHP خود اقدام به یک همچون کد گشایی کرده و حاصل این فرآیند را که شامل جفتهایی از اسامی و مقادیر متناظر با آنهاست دریک آرایه انجمنی ویژه با نام $HTTP_ GET_ VARS که جهت همین کار پیش‌بینی شده است، ذخیره می‌کند. درصورتی که گزینه‌ای با عنوان register_ globals در فایل php. ini تنظیم شده باشد، PHP

علاوه بر کدگشایی‌مذکور یک متغیر سراسری برای هر فیلد ایجاد کرده و آن‌را با مقدار متناظر با آن فیلد مقدار دهی‌می‌کند. از این رو جهت دستیابی به متغیر user_ id می‌توان از هریک از متغیرهای زیر استفاده نمود:

```
$HTTP_ GET_ VARS [ ' user _ id ' ] ;
$user _ id ;
```

با این همه توجه داشته باشید که جهت ارسال دنباله‌های پرس و جو به سرور از فرم تنها روش ممکن نمی‌باشد. درصورتی‌که تمایل داشته باشید می‌توانید با کمی دقت اقدام به ایجاد دنباله‌های پرس و جوی موردنیاز خود نموده و بدین ترتیب اطلاعات مفیدی را طی جلسات مختلف از یک صفحه به صفحه دیگر ارسال نمایید.

چگونگی ایجاد یک دنباله پرس و جو

برای ایجاد درستی دنباله‌های پرس و جو باید رفتاری را که مرورگر اینترنت در ارسال اطلاعات یک فرم به سرور از خود نشان می‌دهد، شبیه سازی کنیم. به‌عبارت دیگر لازم است تا به روشی کلیدها (اسامی فیلدها) و مقادیر متناظر با آنها را کدگذاری نماییم. فرض کنید بخواهیم که یک آدرس URL موجود را به‌عنوان بخشی از یک دنباله پرس و جو به صفحه دیگری ارسال کنیم. از آنجا که علایم / و : موجود در آدرس URL موجب سردرگمی مرورگر در تحلیل دنباله پرس و جوی ارسالی خواهد شد. چاره‌ای نداریم جز اینکه این دو علامت را به نوعی کاملاً مشخص متمایز کنیم. در این راه این دو کاراکتر را به صورت کدهای هگزادسیمال خواهیم نوشت. کدهای هگزادسیمال معادل با علایم / و : به ترتیب عبارتند از 2F و 3A که جهت استفاده از آنها باید از علامت % پیش از هر کد بهره ببریم. چنانکه مشاهده می‌کنید، به‌خاطر سپردن کدهای معادل هگزادسیمال و استفاده از آنها به‌ویژه هنگامی که دنباله پرس و جو اندکی پیچیده و طولانی باشد به‌راحتی می‌تواند موجبات اشتباه برنامه‌نویس را فراهم نماید. خوشبختانه در زبان PHP تابعی پیش‌بینی شده است که فرآیند کدگذاری را دقیقاً به همان شیوه‌ای‌که مرورگرهای اینترنت انجام می‌دهند، شبیه سازی کرد. نام این تابع مفید( ) urlencode است. تابع مذکور ساختار بسیارساده‌ای دارد. این تابع دنباله‌ای از کاراکترها را به‌عنوان تنها آرگومان ورودی می‌پذیرد و به‌عنوان حاصل عملیات دنباله دیگری از کاراکترها را که نسخه کدگذاری شده دنباله کاراکتری ورودی است به برنامه فراخواننده باز می‌گرداند. فراخوانی تابع فوق و نتیجه آن درمورد آدرس URL نمونه‌ای که مشاهده می‌کنید در قطعه کد زیر آورده شده است:

```
Print urlencode ( "http : // www. corrosive. co. uk " ) ;
// prints http%3A%2F%2Fwww. corrosive. co. uk
```

(دقت کنید در این فرآیند تبدیل دنباله کاراکتری اصلی که به‌عنوان آرگومان ورودی تابع ( ) urlencode مورد استفاده قرار می‌گیرد، هیچ‌گونه تغییری نخواهد کرد بلکه یک کپی از آن دستخوش تغییر می‌شود).

اکنون که می‌توانیم دنباله کاراکتری موردنظر را به‌شیوه مورد انتظار URL تبدیل کنیم می‌توانیم به‌راحتی اقدام به ایجاد دنباله‌های پرس و جو کرده و عملکرد مرورگر را در تبدیل اطلاعات فرم به دنباله‌های پرس و جو جهت ارسال به شیوه GET به سرور شبیه سازی نماییم. برای مثال قطعه برنامه زیر را درنظر بگیرید. این قطعه کد با بهره‌گیری از تابع ( ) urlencode مقادیر ذخیره شده در دو متغیر $homepage و $interest را به این شیوه کدگذاری می‌کند:

```
< ? php
$interest = " arts " ;
$homepage = " http : // www. corrosive. co. uk " ;
$query = " homepage = " . urlencode ($homepage) ;
$query ' = " & interest = " . urlencode ($interest) ;
? >
< A HREF = "newpage . php ? <? print$query ? > " > Go < / A >
```

آنچه که پس از کلیک بر روی فراپیوند GO در برنامه فوق مشاهده می‌کنیم، درخواست سند newpage. php به‌همراه دنباله پرس و جوی ایجاد شده از روی آدرس صفحه خانگی و علاقه‌مندی‌هایی است که در قالب متغیرهای $homepage و $interest مشخص شده‌اند. درخواست ارسالی بدین ترتیب چنین خواهد بود:

```
newpage. php ? homepage = http % 3A % 2F % 2F www. corrosive. co. uk &
        interest = arts
```

توجه کنید که به‌واسطه این ارسال (که در پاسخ به فرآیند کلیک روی فراپیوند GO حاصل می‌شود) پارامترهای homepage و interest به‌عنوان دو متغیر سراسری در دسترس برنامه newpage. php قرار خواهند گرفت.

با وجود سادگی و سرراست بودن این روش جهت ارسال اطلاعات، رویکرد فوق را می‌توان دلیلی بر عدم رعایت اصول کدنویسی تلقی کرد. از آنجا که از اسامی دو متغیر $homepage و $interest در فرآیند ایجاد دنباله پرس و جو استفاده کرده‌ایم، نمی‌توان چنین ادعا کرد که بهره‌گیری مجدد از این کد به‌راحتی امکان‌پذیر است (امکان بهره‌گیری و استفاده مجدد از کدها و برنامه‌ها که با عنوان Software Reusability مشهور است یکی از اصول مهم در توسعه برنامه‌ها محسوب شده و عدم رعایت آن علاوه بر اینکه دلیلی بر مهارت کم برنامه‌نویس در کدنویسی می‌باشد می‌تواند توسعه آتی چنین برنامه‌هایی را نیز با مشکلات فراوانی که صرف زمان بسیار تنها یکی از آنهاست، مواجه نماید). جهت اصلاح این نقطه ضعف و ارسال اطلاعات به‌شیوه‌ای قابل اطمینان‌تر مابین صفحات مختلف وب سایت لازم است تا از روشی ساده جهت تعبیه اسامی فیلدها و مقادیر متناظر با آنها در یک فراپیوند و تولید

خودکار دنباله پرس و جو استفاده نماییم. این روش هنگامی از اهمیت زیاد برخوردار می‌شود که کار خود را با توجه با امکاناتی که PHP با درنظرگرفتن برنامه‌نویسان تازه کار یا غیر برنامه‌نویسان در اختیار قرار داده است انجام دهیم.

برنامه موجود در لیست ۸-۱۹ تابعی با نام ( ) qlink ایجاد می‌کند. این تابع ساختار ساده‌ای داشته و تنها از یک آرایه انجمنی به‌عنوان آرگومان ورودی بهره می‌برد. در مقابل آنچه که تابع فوق به‌عنوان نتیجه عملیات خود به برنامه فراخواننده باز می‌گرداند یک دنباله پرس و جو است که از این آرایه انجمنی حاصل می‌شود.

```
 1: <html>
 2: <head>
 3: <title>Listing 19.5 A function to build query strings</title>
 4: </head>
 5: <body>
 6: <?php
 7: function qlink( $q ) {
 8:     if ( ! $q )
 9:     return $GLOBALS['QUERY_STRING'];
10:     $ret = "";
11:     foreach( $q as $key => $val ) {
12:         if ( strlen( $ret ) ) $ret .= "&";
13:         $ret .= urlencode( $key ) . "=" . urlencode( $val );
14:     }
15:     return $ret;
16: }
17: $q = array (
18:          'name' => "Arthur Harold Smith",
19:          'interest' => "Cinema (mainly art house)",
20:          'homepage' => "http://www.corrosive.co.uk/harold/"
21:          );
22: print qlink( $q );
23: // prints name=Arthur+Harold+Smith&interest=Cinema+%28mainly+art+house
24: //         %29&homepage=http%3A%2F%2Fwww.corrosive.co.uk%2Fharold%2F
25: ?>
26: <p>
27: <a href="anotherpage.php?<? print qlink($q) ?>">Go!</a>
28: </p>
29: </body>
30: </html>
```

لیست ۸- ۱۹ تابعی جهت ایجاد یک دنباله پرس و جو از روی یک آرایه انجمنی

همان‌گونه که در این لیست مشاهده می‌کنید تابع ( ) qlink در خط ۷ تعریف شده است. تابع مزبور یک آرایه انجمنی را که شامل مقادیر فیلدها و اسامی آنها است در قالب متغیری با عنوان $q دریافت می‌کند. درصورتی‌که متغیر مذکور فاقد مقدار مورد انتظار تابع مورد بحث باشد، با استفاده از دستورالعمل return موجود در خط ۹ از برنامه دنباله پرس و جوی فعلی که در متغیر $QUERY_STRING ذخیره شده است به برنامه فراخواننده این تابع بازگردانده می‌شود. بدین ترتیب

تابع ( ) qlink می‌تواند به‌سادگی جهت ارسال داده‌های مربوط به درخواست GET که بدون تغییر و دست‌نخورده باقی می‌مانند، مورد استفاده قرار بگیرد.

اما با فرض اینکه متغیر $q با مقدار مورد انتظار تابع (یعنی آرایه انجمنی) مقداردهی شده باشد برنامه در خط ۱۰ ابتدا متغیری با نام $rel را با یک دنباله کاراکتری تهی مقداردهی می‌کند. این متغیر چنانکه به‌زودی خواهید دید شامل دنباله پرس و جوی حاصل از عملیات این تابع خواهد بود.

از آنجا که جهت ایجاد دنباله پرس و جوی موردنظر لازم است تا کلیه عناصر موجود در آرایه انجمنی $q را مورد پردازش قرار دهیم در خط ۱۱ برنامه از یک ساختار تکرار از نوع foreach استفاده کرده و در هربار گذر از حلقه نام کلید دستیابی یکی از عناصر آرایه را در متغیر $key و مقدار متناظر با آن‌را در متغیر $val قرار می‌دهیم.

از آنجا که در دنباله پرس و جوی حاصل اسامی کلیدهای دستیابی و مقادیر متناظر با آنها باید با استفاده از علامت & از یکدیگر جدا شوند، دستورالعمل خط ۱۲ از برنامه به شرط اینکه گذر فعلی از حلقه اولین گذر از حلقه نباشد، کاراکتر & را به‌عنوان خروجی چاپ خواهد کرد.

چنانکه می‌دانیم طول دنباله‌های کاراکتری موجود در متغیر $ret در اولین گذر از حلقه برابر با صفر است، به همین جهت می‌توانیم با ارزیابی طول دنباله کاراکتری مذکور از وجود چنین گذر خاصی مطلع شویم.

تابع تبدیل کننده ( ) urlencode را در خط ۱۳ برنامه و در درون تابع qlink مورد فراخوانی قرارداده‌ایم.همان‌گونه‌که حدس می‌زنید این تابع مقادیر موجود در متغیرهای $key و $val را کدگذاری کرده و آنها را به یکدیگر ضمیمه می‌کند. این نوع ضمیمه کردن با بهره‌گیری از علامت تساوی (=) که در اینجا مابین مقادیر موجود در متغیرهای فوق را بیان می‌کند انجام می‌پذیرد. در نهایت، حاصل این پیوند به متغیر $ret نسبت داده می‌شود.

در انتهای تابع پس از اینکه فرآیند تبدیل با گذر از تمامی حلقه‌ها به ازای مقادیر موجود در آرایه ورودی $q انجام شد، متغیر $ret به‌عنوان نتیجه عملیات تابع ( ) qlink به برنامه فراخواننده بازگردانده می‌شود.

با بهره‌گیری از این تابع همان‌گونه که مشاهده می‌کنید می‌توانیم با به‌کارگیری کمترین کد برنامه PHP در قالب نشانه‌های HTML اقدام به ارسال اطلاعات مابین صفحات مختلف وب نماییم.

جمع‌بندی

در درس این ساعت با دو روش متداولی که برنامه‌نویسان وب جهت ارسال اطلاعات مورد نظرشان بین درخواستهای مختلف مورد استفاده قرار می‌دهند، آشنا شدید. با بهره‌گیری از این روشها می‌توانید اقدام به ایجاد برنامه‌های کاربردی " چند پنجره‌ای " نموده و محیطی را در اختیار کاربر قرار

دهید که از طریق آن به‌راحتی از امکانات وب سایت استفاده کرده و اطلاعات موجود در آن‌را با اطمینان و تسهیلات بیشتری مورد دستیابی قراردهد.

در این ساعت چگونگی استفاده از امکانات تابع ارزشمند ( ) setcookie جهت تنظیم و بهره‌برداری از کوکی‌های موجود بر روی کامپیوتر بازدیدکنندگان از وب سایت را فراگرفتید. طی این فراگیری مشاهده کردید که چگونه می‌توان با بهره‌گیری از یک بانک اطلاعاتی به‌همراه کوکی‌های موجود اطلاعات بسیار مفیدی را درمورد بازدیدکنندگان و چگونگی بهره‌برداری آنها از محتویات وب سایت مابین جلسات مختلف ثبت و نگهداری کرد. همچنین متوجه شدید که نگهداری و استفاده از یک چنین اطلاعات مفید آماری کاملاً در پیگیری علاقه‌مندی‌های بازدیدکنندگان و تنظیم محتوای صفحات وب سایت با توجه به علایق و نیازهای آنان تأثیر قابل توجهی دارد. علاوه براین در این ساعت با چگونگی ایجاد دنباله‌های پرس و جو و استفاده از آنها جهت ثبت و ارسال اطلاعات مختلف بین صفحات وب موجود در سایت آشنا شدید و طی آن چگونگی کدگذاری اطلاعات مذکور را به شیوه URL فراگرفتید. در انتهای درس نیز تابعی را با نام ( ) qlink جهت مکانیزه نمودن فرآیند کدگذاری توسعه دادید.

PHP4 قابلیتهای فراوانی را در اختیار برنامه‌نویسان این زبان قرار داده است. در درس ساعت آینده شما را با برخی از توابع سیستمی آشنا می‌کنیم، این توابع امکانات بسیار جالب توجهی را در رابطه با مکانیزه کردن بسیاری از فرآیندهایی که در این درس آنها را تحت بررسی قرار دادیم در اختیار برنامه‌نویسان قرار می‌دهند.

پرسش و پاسخ

پرسش: آیا در مورد استفاده از کوکی‌ها نکته قابل ذکری درباره ضعف امنیتی وجود دارد؟

پاسخ: نکته‌ای که باید متوجه آن باشید این است که کوکی‌ها همواره تنها دراختیار سروری قرار می‌گیرند که پیشتر آنها را بر روی مرورگر بازدید کننده ارسال کرده است. با وجودی که کوکی‌ها برروی هارددیسک کامپیوتر بازدیدکنندگان سایتهای مربوطه ثبت و ذخیره می‌شوند، این اطمینان وجود دارد که هیچ بخش دیگری از سیستم فایل کامپیوتر این کاربران به‌واسطه این فرآیند مورد تاخت و تاز قرار نمی‌گیرد. با این حال این احتمال وجود دارد که کوکی به‌واسطه درخواستی که برای دریافت یک تصویر ارسال شده است، تنظیم شده باشد. بنابراین اگر سایتهای مورد دستیابی کاربر شامل تصاویری باشند که توسط یک سرور خاص که به منظور در دسترس قرار دادن تصاویر طراحی شده است تامین شده باشد، این احتمال وجود دارد که سرور مزبور بتواند رد پای بازدیدکنندگان را در میان چندین حوزه مختلف پیگیری نماید.

پرسش: ظاهر دنباله پرس و جو به‌ویژه هنگامی که در فیلد آدرس پنجره مرورگر اینترنت قرار می‌گیرد بسیار زننده و نامطلوب است. بهتر نیست همواره از کوکی‌ها جهت ثبت وضعیت جلسات استفاده کنیم؟ آیا کوکی بهترین روش برای انجام این کار نیست؟

پاسخ: با کمال تاسف باید اعتراف کنیم که فرآیند مذکور به این سادگی که ظاهر آن نشان می‌دهد، نیست. کاملاً صحیح است که استفاده از کوکی بهترین روش برای حفظ وضعیت جلساتی است که طی آنها کاربر وب سایت را مورد بازدید قرار می‌دهد اما یک واقعیت تلخ همواره در بین است و آن این است که برخی از کاربران با بهره‌گیری از تنظیماتی که مرورگرهای وب در اختیار آنها قرار می‌دهد، ترتیبی می‌دهند تا به‌محض ارسال کوکی از جانب سرور آنها را با نمایش پیغامی آگاه می‌کنند. بدین ترتیب احتمال زیادی وجود دارد که با اجتناب از پذیرش کوکی‌ها منجر به عدم سرویس‌دهی وب سایت به ایشان گردند. این‌گونه کاربران از آن‌دسته افرادی هستند که به احتمال زیاد سایتهایی را در اولویت قرار می‌دهند که از روشی غیر از کوکی‌ها جهت ردیابی و ثبت جلسات استفاده می‌کنند.

تمرینها

هدف از این بخش ارائه تمرینهایی است که در قالب آزمون طراحی شده‌اند. پاسخ بخش آزمون بلافاصله بعد از آن آمده است. بخش فعالیتها شامل تمرینهایی است که به قصد افزایش مهارت و قابلیت برنامه‌نویسان طراحی شده است. این بخش فاقد پاسخ است.

آزمون

۱- از کدام تابع می‌توان جهت تنظیم کوکی برروی مرورگر اینترنت کاربر استفاده کرد؟
۲- چگونه می‌توان یک کوکی را حذف کرد؟
۳- با استفاده از کدام تابع می‌توان یک دنباله کاراکتری را جهت استفاده در دنباله پرس و جو از یک درخواست در قالب خاصی که به کدگذاری URL شهرت دارد، رمزگذاری کرد؟
۴- کدام متغیر سیستمی شامل فرم خام دنباله پرس و جو (فرم رمز گذاری نشده) می‌باشد؟
۵- چنانکه در متن درس نیز عنوان شد، هر زوج شامل نام و مقدار که به عنوان بخشی از یک دنباله پرس و جو به سرور ارسال می‌شوند در قالب متغیرهای سراسری در اختیار برنامه‌نویس قرار می‌گیرند. ضمناً این زوجها در قالب یک آرایه انجمنی سیستمی نیز ذخیره می‌شوند. نام این متغیر خاص و آرایه انجمنی مذکور را بیان کنید.

پاسخ آزمون

۱- با بهره‌گیری از تابع ( ) setcookie می‌توان کوکی‌های موردنظر را بر روی مرورگر اینترنت کاربران بازدیدکننده از وب سایتها تنظیم نمود (روش دیگر این کاراستفاده از تابع ( ) header است که هدر set _ cookie از پاسخ را که مربوط به تنظیم کوکی است، تعریف می‌کند).

۲- برای حذف یک کوکی کافی است تابع ( ) setcookie را با تاریخی که پیش از تاریخ فعلی مشخص می‌کند، فراخوانی کنیم.

۳- تابع ( ) urlencode قادر است یک دنباله کاراکتری را دریافت کرده و آن‌را به شیوه URL کدگذاری نماید. در این فرآیند آنچه دستخوش تغییر می‌شود کپی دنباله کاراکتری فوق بوده و دنباله کاراکتری اصلی دست‌نخورده باقی می‌ماند.

۴- کل دنباله پرس و جو را می‌توان به صورت خام (کد گذاری نشده) از طریق متغیر سیستمی $QUERY_ STRING مورد دستیابی قرار داد.

۵- پس از ارسال درخواست به سرور، متغیر سیستمی $HTTP_ GET_ VARS شامل زوجهایی از اسامی و مقادیر متناظر با آنها خواهد بود.

فعالیتها

۱- یک فرم نمونه ایجاد کنید که بازدیدکننده بتواند علاقه‌مندی‌هایش را از طریق آن مشخص کند. در این فرم کاربر باید بتواند رنگ دلخواه خود و همچنین نام دلخواهی را به‌عنوان نام کاربری خود انتخاب نماید. با بهره‌گیری از کوکی‌ها ترتیبی دهید تا دفعه بعدی که کاربر اقدام به بازدید از سایت می‌کند رنگ صفحه با رنگ دلخواه او تنظیم شده و پیغام خوش‌آمدگویی دوستانه‌ای با استفاده از نام تعیین شده توسط وی بر روی پنجره مرورگر نمایش پیدا کند.

۲- فعالیت موجود در تمرین قبل را به‌گونه‌ای انجام دهید که به‌جای استفاده از کوکی از یک دنباله پرس و جو برای انجام این کار استفاده نماید. نتیجه در هر حال باید معادل قبل باشد.

ساعت بیستم

ثبت وضعیت با بهره‌گیری از توابع ثبت جلسات

در درس ساعت قبل چنانکه ملاحظه کردید چگونگی ثبت وضعیت را از صفحه‌ای به صفحه دیگر، ضمن بازدید کاربر از وب سایت، مورد بحث و بررسی قراردادیم. دو روشی را که در آن درس عنوان کردیم عبارت بودند از کوکی و دنباله پرس و جو. بار دیگر اعتراف می‌کنیم که PHP4 یک گام جلوتر از ماست. با انتشار PHP4 ، توابع مورد نیاز جهت مدیریت و ثبت وضعیت جلسات به عنوان بخشی از این زبان برنامه نویسی در اختیار برنامه‌نویس قرارگرفت. این توابع روشی را جهت ثبت وضعیت مورد استفاده قرار می‌دهند که ما در درس ساعت قبل بررسی کردیم. اما تفاوت این است که کل روش در قالب تابعی در اختیار ما قرار گرفته است که این خود باعث می‌شود تا فرآیند ثبت وضعیت در حد فراخوانی یک تابع ساده شود.

در درس این ساعت موارد زیر را بررسی خواهیم کرد:

- **بررسی متغیرهای جلسه و نحوه کاربرد آنها**
- **چگونگی ایجاد یک جلسه جدید و از سرگیری یک جلسه قدیمی**
- **نحوه ثبت متغیرها در رابطه با یک جلسه**
- **چگونگی تخریب یک جلسه**
- **چگونگی unset کردن متغیرهای جلسه**

در ادامه به بررسی هریک از این موارد می‌پردازیم:

توابع ثبت جلسات

توابع ثبت جلسات مفهومی را پیاده‌سازی می‌کنند که پیشتر در درس ساعت قبل آن‌را مورد بحث قرار دادیم و هم اینک با آن آشنا هستید. این مفهوم بسیار ساده بوده و عبارت است از تخصیص شناسه‌ای منحصر به فرد به هر یک از کاربران به‌گونه‌ای که بتوان طی دستیابی‌های مختلف آنها به صفحات موجود در یک وب سایت اطلاعات دیگری را که در رابطه با آن شناسه‌ها در جایی (مثلاً یک بانک اطلاعاتی) ذخیره شده است، مورد بهره‌برداری و استفاده قرار دارد. این رویکرد همان روشی بود که در درس ساعت گذشته در قالب دو روش کوکی و دنباله‌های پرس و جو مورد بحث قرار گرفت. با این همه تفاوتی که این توابع با آن روشها دارند این است که کل عملیات مورد نظر تحت قالب یک تابع پیاده‌سازی شده است. از آنجا که این توابع توسط برنامه‌نویسان قهار توسعه پیدا کرده و اشکال‌زدایی شده‌اند، می‌توان هنگام استفاده از آنها اطمینان خاطر فراوانی داشت. توابع ویژگی بسیار قابل توجه دیگری نیز دارند و آن این است که پس از توسعه یک تابع بارها و بارها می‌توان آن‌را در یک برنامه و حتی در برنامه‌های مختلف مورد بهره‌برداری قرارداد. هنگامی که کاربر صفحه‌ای از وب سایت را که به ابزار ردیابی بازدیدکنندگان مجهز شده مورد دستیابی قرار می‌دهد، دو امکان پیش رو قرار می‌گیرد. درصورتی‌که کاربر برای اولین بار چنین سایتی را مورد دستیابی قرار داده باشد، شناسه منحصر به فرد به او اختصاص داده می‌شود. اما چنانچه کاربر پیشتر این سایت را مورد دستیابی قرار داده باشد، شناسه منحصر به فردی که پیشتر به او اختصاص داده شده بود، مورد استفاده قرار خواهد گرفت. نکته قابل ذکر در این میان این است که هر متغیری که قبلاً در مورد این جلسه تعریف شده بود کاملاً در دسترس کدبرنامه قرار می‌گیرد. درصورتی‌که گزینه register_ globals از فایل php. ini تنظیم شده باشد، داده‌های جلسه نیز در دسترس برنامه قرار خواهند داشت. درغیر این‌صورت جهت دستیابی به این داده‌ها می‌توان از آرایه انجمنی $HTTP_ SESSION_ VARS که یک آرایه سیستمی است، استفاده نمود.

هر دو روشی را که در درس ساعت قبل جهت ارسال اطلاعات از یک صفحه به صفحه دیگر به‌منظور ثبت وضعیت جلسه مورد بررسی قرار دادیم به‌طور خودکار توسط توابع ثبت جلسات که به‌عنوان بخشی از زبان برنامه‌نویسی PHP محسوب می‌شوند، به‌همراه PHP4 منتشر شده و در اختیار برنامه‌نویس قرار می‌گیرند. بنا به پیش‌فرض استفاده از کوکی‌ها نسبت به دنباله‌های پرس و جو در اولویت قرار می‌گیرند، اما با امکانات موجود که توابع مورد بحث را در اختیارمان قرار می‌دهند می‌توان اطمینان حاصل کرد که به‌واسطه ضمیمه کردن شناسه منحصر به فردی که به‌ازای هر جلسه به کاربر تخصیص داده شده و در تمامی پیوندهای موجود در صفحات سایت مورد استفاده قرار می‌گیرند، فرآیند موردنظر کاملاً با موفقیت و بدون هیچ مشکلی صورت خواهد پذیرفت.

وضعیت جلسه معمولاً در قالب یک فایل موقت ذخیره می‌شود با این حال درصورت تمایل می‌توان جهت انجام این کار از یک بانک اطلاعاتی استفاده کرد. این فرآیند با بهره‌گیری از تابع ویژه‌ای که ( ) session_ set_ save_ handler نام دارد قابل پیاده‌سازی است. تابع session_ set_ save_ handler ( ) و جزئیات مربوط به چگونگی عملکرد و همچنین نحوه استفاده از آن چیزی نیست که در قالب این کتاب قابل بررسی باشد. با این حال جهت کسب اطلاع بیشتر درباره جزئیات به‌کارگیری این تابع درصورت تمایل می‌توانید به آدرس زیر مراجعه کنید:

http: // www. Php. net / manual / en / function . session_ set_ save_ handler. php

آغاز یک جلسه با استفاده از تابع ( ) session_ start

چنانچه فایل تنظیمات php. ini را به‌گونه‌ای مناسب پیکربندی نکرده باشید، جهت آغاز یک جلسه جدید و یا از سرگیری یک جلسه قدیمی لازم است تا به‌طور صریح فرآیند مورد نظرتان را مشخص کنید. بنا به پیش‌فرض جلسات به‌طور خودکار با دستیابی کاربر به صفحات وب سایت آغاز نمی‌شوند. جهت خودکار سازی این فرآیند لازم است تا فایل تنظیمات php. ini را با تغییر گزینه خاصی از آن مجدداً پیکربندی نمایید. اگر هم‌اکنون فایل مذکور را باز کنید خطی را مشاهده می‌کنید که به صورت زیر مقدار گزینه مورد نظر ما یعنی session . auto_ start را تنظیم کرده است:

```
session . auto_ start = 0
```

با تغییر مقدار متغیر session . auto_ start از صفر به یک می‌توانید مطمئن شوید که به‌ازای دستیابی کاربر به هریک از اسناد PHP موجود در وب سایت یک جلسه کاری تشکیل خواهد شد. درصورتی که متغیر session . auto_ start را با مقدار یک تنظیم نکنید، لازم است تا درصورت تمایل به تشکیل جلسه در ازای دستیابی کاربر به صفحهPHP مورد نظرتان هر بار تابعی با نام ( ) session_ start را فراخوانی نمایید.

پس از آغاز یک جلسه (به‌طور خودکار یا با فراخوانی تابع ( ) session_ start) بلافاصله می‌توانید با بهره‌گیری از تابعی با عنوان ( ) session_ id به شناسه جلسه مربوط به این کار دسترسی داشته باشید. با بهره‌گیری از این تابع علاوه بر دستیابی به این شناسه قادر خواهید بود تا درصورت تمایل مقدار آن‌را نیز تنظیم نمایید. برنامه‌ای که در لیست ۱- ۲۰ آرایه می‌کنیم با بهره‌گیری از توابع فوق اقدام به آغاز یک جلسه نموده و شناسه مربوطه را بر روی صفحه مرورگر اینترنت نمایش دهد.

```
1: <?php
2: session_start();
3: ?>
4: <html>
5: <head>
6: <title>Listing 20.1 Starting or resuming a session</title>
7: </head>
8: <body>
9: <?php
10: print "<p>Welcome, your session ID is ".session_id()."</p>\n\n";
11: ?>
12: </body>
13: </html>
```

لیست ۱- ۲۰ چگونگی تشکیل یک جلسه

هنگامی که این برنامه برای اولین مرتبه از یک مرورگر اینترنت اجرا می‌شود به‌واسطه فراخوانی تابع( ) session_ start در خط ۲ یک شناسه منحصر به فرد برای جلسه‌ای که به‌تازگی تشکیل شده است، تولید می‌گردد. درصورتی‌که این برنامه باردیگر از طریق دستیابی به سند PHP مربوطه اجرا شود همین شناسه منحصر به‌فرد به کاربری که اقدام به بازدید نموده است، تخصیص داده خواهد شد. کل این فرآیند البته با این پیش‌فرض انجام می‌شود که کاربر مورد بحث گزینه مربوط به پذیرش کوکی‌ها بر روی کامپیوتر خود را تغییر نداده باشد و بدون هیچ مانعی کوکی ارسالی از سرور بر روی کامپیوتر کاربر ذخیره شود. درصورتی که هدرهای HTTP حاصل از این ارسال را که در لیست ۱- ۲۰ انجام شده است مورد بررسی قرار دهید، خواهید دید که این برنامه از طریق هدر cookie _set اقدام به تنظیم کوکی نموده است.

```
4: <html>
5: <head>
6: <title>Listing 20.1 Starting or resuming a session</title>
7: </head>
8: <body>
9: <?php
10: print "<p>Welcome, your session ID is ".session_id()."</p>\n\n";
11: ?>
12: </body>
13: </html>
```

برنامه لیست ۱- ۲۰

به‌دلیل اینکه تابع ( ) session _ start پس از آغاز یک جلسه (به‌ازای اولین دستیابی کاربر به سند PHP مربوطه) اقدام به تنظیم یک کوکی بر روی کامپیوتر بازدیدکننده می‌کند، لازم است تا این تابع پیش از آنکه برنامه محتوایی را به‌عنوان خروجی بر روی پنجره مرورگر اینترنت نمایش دهد، مورد فراخوانی قرار گیرد. نکته قابل توجه دیگر در لیست ۲- ۲۰ این است که در تنظیم کوکی که به ازای جلسه کاری موجود انجام شده است هیچ اثری از تاریخ انقضای کوکی (که در درس ساعت گذشته در مورد آن صحبت کردیم) به‌چشم نمی‌خورد. عدم تنظیم تاریخ انقضای کوکی همان‌گونه که به احتمال

قوی متوجه شده‌اید، بدان مفهوم است که عمر مفید کوکی موردنظر به اندازه عمر پنجره مرورگری است که برنامه از طریق آن اجرا شده است. به‌عبارت دیگر، کوکی مورد بحث با بسته شدن پنجره‌ای که به‌واسطه دستیابی کاربر از طریق آن به برنامه ایجاد شده است از بین خواهد رفت و هنگامی که کاربر پنجره مرورگر دیگری را باز کند اثری از کوکی مذکور به‌چشم نخواهد خورد. البته درصورت تمایل می‌توانید این رفتار پیش‌فرض را تغییر دهید. برای انجام این کار لازم است تا گزینه ویژه‌ای با عنوان session . cookie_ lifetime را از فایل تنظیمات php. ini مجدداً تنظیم نمایید. مقدار پیش‌فرض این گزینه برابر با صفر است. درصورتی‌که قصد تنظیم تاریخ انقضای کوکی موردنظر را داشته باشید، لازم است تا مقدار این گزینه را بر مبنای تعداد ثانیه‌های مورد نظرتان مشخص نمایید. عددی که در این مورد به‌کار می‌برید بیانگر طول عمر کوکی مورد نظر برحسب ثانیه خواهد بود. این فرآیند باعث می‌شود تا هر کوکی ارسالی به مرورگر اینترنت کاربر، طول عمری برابر با مقدار فوق داشته باشد.

بهره‌گیری از متغیرهای جلسه

دستیابی به یک شناسه منحصر به فرد به‌ازای دستیابی به هر یک از اسناد PHP موجود در وب سایت تنها یکی از قابلیتهای چشمگیر PHP4 در رابطه با امکان پی‌گیری و ثبت جلسات است. حقیقت این است که در صورت لزوم می‌توان به هر تعداد مورد نیاز از متغیرهای سراسری به‌همراه جلساتی که ایجاد می‌شوند، استفاده نمود. بدین ترتیب می‌توان در حین دسترسی به هر سندی از نوع PHP که شامل پشتیبانی جلسات باشد این متغیرها را مورد دستیابی قرار داد.

به منظور ثبت یک متغیر به‌همراه جلسه جاری کافی است تابعی را که به‌همین منظور در زبان PHP پیش‌بینی شده است و ( ) session_ register نام دارد، فراخوانی کنیم. تابع (session_ register ساختار ساده‌ای داشته و یک دنباله کاراکتری را که نماینده نام یک یا چند متغیر است به‌عنوان آرگومان ورودی دریافت می‌کند. تابع مورد بحث درصورتی که فرآیند ثبت متغیر به‌همراه جلسه با موفقیت همراه باشد، مقدار true را به برنامه فراخواننده باز می‌گرداند. نکته جالب توجهی که در رابطه با آرگومان ارسالی به تابع ( ) session_ register وجود دارد، نحوه غیر متداولی است که در انجام این کار مورد استفاده قرار می‌گیرد. به بیان دیگر هنگام تامین آرگومان این تابع تنها باید نام متغیر را به‌عنوان آرگومان ارسال کرده و از ارسال خود متغیر به این تابع خودداری نمایید. بررسی یک برنامه نمونه فرآیند مذکور را روشن‌تر می‌سازد.

برنامه موجود در لیست ۲-۲۰ با بهره‌گیری از تابع ( ) session_ register در خطوط ۱۰ و ۱۱ اقدام به ثبت دو متغیر product 1 و product 2 به‌همراه جلسه جاری می‌کند.

```
1: <?php
2: session_start();
3: ?>
4: <html>
5: <head>
6: <title>Listing 20.2 Registering variables with a session</title>
7: </head>
8: <body>
9: <?php
10: session_register( "product1" );
11: session_register( "product2" );
12: $product1 = "Sonic Screwdriver";
13: $product2 = "HAL 2000";
14: print "The products have been registered";
15: ?>
16: </body>
17: </html>
```

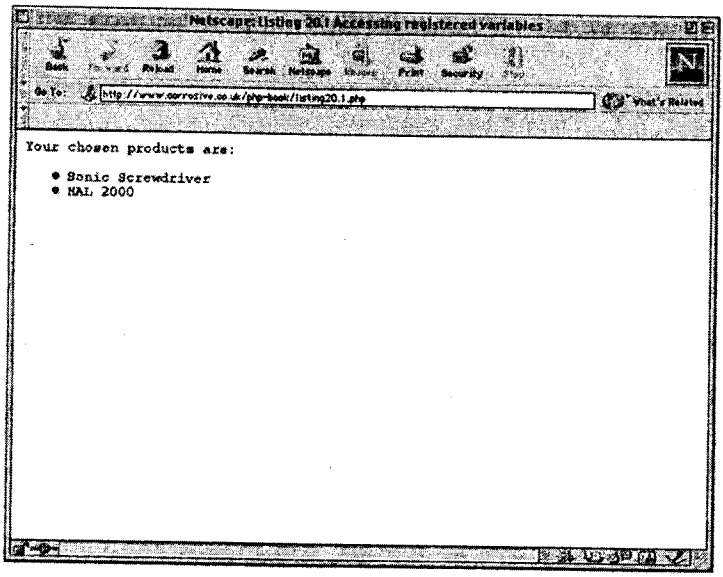
لیست ۲- ۲۰ ثبت متغیرهای جلسه

```
1: <?php
2: session_start();
3: ?>
4: <html>
5: <head>
6: <title>Listing 20.3 Accessing registered variables</title>
7: </head>
8: <body>
9: <?php
10: print "Your chosen products are:\n\n";
11: print "<ul><li>$product1\n<li>$product2\n</ul>\n";
12: ?>
13: </body>
14: </html>
```

لیست ۳- ۲۰ دستیابی به متغیرهای جلسه

تأثیر جادویی برنامه ارائه شده در این لیست تا مادامی که کاربر صفحه جدیدی از وب سایت مربوطه را مورد بازدید خود قرار نداده است آشکار نخواهد شد. برنامه موجود در لیست ۳-۲۰ یک اسکریپت PHP دیگری را نشان می‌دهد که از متغیرهای ثبت شده توسط برنامه لیست ۲-۲۰ استفاده می‌کند.

چنانکه ملاحظه می‌کنید در خط ۱۱ از این برنامه متغیرهای جلسه‌ای با نام $product 1 و $product 2 که پیشتر توسط لیست ۲-۲۰ در پاسخ به دستیابی کاربر به صفحه PHP مربوطه ایجاد شده بودند با بهره‌گیری از تابع ( ) print بر روی صفحه مرورگر اینترنت به نمایش درآمده‌اند. دسترسی به این‌گونه متغیرها در هر صفحه جدید دیگری به همین سادگی انجام‌پذیر است. خروجی حاصل از این لیست در شکل ۱- ۲۰ قابل مشاهده است.



شکل ۱- ۲۰ نتیجه دستیابی به متغیرهای جلسه

اجازه دهید تا در این قسمت چگونگی عملکرد این فرآیند جادویی را از نزدیک بررسی کرده و مکانیزم درونی مورد استفاده آن‌را بیشتر بشناسیم. حقیقت این است که در پشت پرده PHP4 مشغول نوشتن اطلاعاتی در یک فایل موقت است. اگر مایل باشید تا به محل دقیق این فایل موقت بر روی سیستم فایل موجود بر روی کامپیوترتان پی ببرید، می‌توانید تابع ( ) session_ save_ path را فراخوانی نمایید. تابع( ) session_ save_ path بسیار ساده بوده و جهت فراخوانی لزوماً به هیچ آرگومانی نیاز ندارد، اما درصورت تمایل می‌توانید از یک آرگومان اختیاری که نشانگر مسیر فهرست موردنظرتان است استفاده کرده و کلیه فایل‌های مربوط به اطلاعات جلسات مختلف را در آن بنویسید. درصورتی که هنگام فراخوانی تابع ( ) session_ save_ path از هیچ آرگومانی جهت ارسال به آن استفاده نکنید، تابع نامبرده به‌عنوان نتیجه عملیات خود، در یک دنباله کاراکتری به برنامه فراخواننده باز می‌گرداند. این دنباله کاراکتری نماینده فهرست جاری خواهد بود که فایل‌های شامل اطلاعات جلسات (فایل‌های موقت) در آنجا نگهداری می‌شوند. فراخوانی زیر را به‌عنوان یک مثال درنظر بگیرید:

```
Print session _ save_ path ( ) ;
```

این فراخوانی (بر روی سیستم از نوع UNIX) احتمالاً فهرستی با عنوان tmp / را مشخص خواهد کرد. بر روی یک سیستم نمونه UNIX فهرست فوق شامل فایل‌هایی با اسامی زیر می‌باشند:

```
sess_ 26388864e9216fee10fcb8a61db382909
sess _ 76cae8ac1231b11afa2c69935c11dd 95
sess _ bb55771a769c605ab77424d59c784ea0
```

در یک سیستم نمونه که به‌واسطه اجرای برنامه لیست ۱- ۲۰ برای اولین مرتبه یک شناسه منحصر به فرد برای جلسه اختصاص داده شده است، می‌توان چگونگی ذخیره متغیرهای ثبت شده را

در این فایل‌های موقت مشاهده کرد. نمونه زیر چگونگی این ذخیره‌سازی را بر روی کامپیوتر مؤلف نشان می‌دهد:

Product1 | S : 17 : "Sonic Screwdriver" ; product2 | s : 8: "HAL 2000" ;

به محض فراخوانی تابع session _ register ( ) نام متغیرها به‌همراه مقادیر مربوطه توسط PHP در یک فایل موقت جهت دستیابی‌های بعدی که به‌واسطه درخواست کاربر جهت بازدید صفحات صورت می‌پذیرد، ذخیره می‌شود. طی دستیابی‌های آتی به این ترتیب این مقادیر را می‌توان مورد بازخوانی قرار داد.

چنانکه ملاحظه کردید برنامه موجود در لیست ۲-۲۰ چگونگی فرآیند ثبت متغیرها را به‌همراه یک جلسه کاری (بازدید) نشان داد. با این همه باید اظهار کرد که روش مذکور از کارآیی و قابلیت انعطاف مطلوبی برخوردار نمی‌باشد. در حالت مطلوب باید بتوان چندین متغیر را به‌همراه یک جلسه بازدید ثبت نمود با این توضیح که این تعداد متغیرها در حین بازدید، مثلاً به‌واسطه تصمیمی که بازدیدکننده درمورد تعداد اقلامی که اسامی آنها را از یک لیست انتخاب می‌کند، باید تعیین شوند. برای نمونه، ممکن است به‌واسطه فرمی به بازدیدکننده این اجازه را بدهید که از میان محصولات مختلف موجود نام چند محصول را از لیستی که به همین منظور تدارک دیده شده است، انتخاب کند. خوشبختانه ساختار تابع( ) session _ register برای ثبت متغیرهای متعدد تنها با یک فراخوانی بسیار مناسب و کارآمد است. برای این منظور کافی است تا نام آرایه‌ای را به‌عنوان آرگومان این تابع ارسال کنید. آنچه که در این حالت اتفاق می‌افتد این است که تابع فوق کلیه داده‌ها را پس از کدگذاری (به روش URL) در این آرایه ذخیره خواهد کرد.

ارائه یک برنامه نمونه وضعیت را به‌خوبی روشن می‌کند. برنامه موجود در لیست ۴-۲۰ فرمی را ایجاد می‌کند که امکان انتخاب چندین محصول مختلف را به طور هم‌زمان در اختیار کاربر قرار می‌دهد. با در دست داشتن این برنامه می‌توان برنامه دیگری را توسعه داد که مانند یک کارت خرید اسامی محصولات منتخب را به‌همراه جزئیات مربوطه مورد دستیابی و نمایش قرار دهد.

```
1: <?php
2: session_start();
3: ?>
4: <html>
5: <head>
6: <title>Listing 20.4 Registering an array variable with a session</title>
7: </head>
8: <body>
9: <h1>Product Choice Page</h1>
10: <?php
11: if ( isset( $form_products ) ) {
12:     if ( empty( $products ) )
13:         $products=$form_products;
14:     else
15:     $products = array_unique(
```

```
16:                     array_merge( $products, $form_products ) );
17:     session_register( "products" );
18:     print "<p>Your products have been registered!</p>";
19: }
20: ?><p>
21: <form method="POST">
22: <select name="form_products[]" multiple size=3>
23: <option> Sonic Screwdriver
24: <option> Hal 2000
25: <option> Tardis
26: <option> ORAC
27: <option> Transporter bracelet
28: </select>
29: </p><p>
30: <input type="submit" value="choose">
31: </form>
32: </p>
33: <a href="listing20.5.php">A content page</a>
34: </body>
35: </html>
```

لیست ۴- ۲۰ ثبت یک متغیر آرایه به همراه جلسه بازدید

چنانکه در این لیست مشاهده می‌کنید با فراخوانی تابع ( ) session_ start در خط ۲ از برنامه جلسه بازدید جدیدی آغاز شده است. به‌واسطه این فرآیند قادر خواهیم بود تا به تمامی متغیرهای جلسه‌ای که پیش از این تعریف شده‌اند، دسترسی داشته باشیم. فرم HTML موردنظر در خط ۲۱ با بهره‌گیری از نشانه < form > تعریف شده است. در خط ۲۲ با استفاده از نشانه < select > لیستی از محصولات را ایجاد کرده‌ایم (نام این لیست چنانکه در خط ۲۲ مشخص شده [ ] form_ products است). این لیست شامل چندین نشانه < option > است که هریک نام یک محصول را مشخص می‌کند. به‌خاطرداشته‌باشیدکه‌عناصری از فرمهای HTML که امکان چندین انتخاب را در اختیار بازدیدکنندگان قرار می‌دهند باید نامی شبیه به نام یک آرایه داشته باشند. به‌عبارت دیگر نام لیست موردنظر که با نشانه < select > مشخص می‌شود باید با جفت علامت [ ] پایان یابد. بدین ترتیب می‌توان اقلام منتخب کاربر را از طریق آرایه‌ای با نام فوق در برنامه مورد دستیابی قرار داد.

در داخل کد PHP موجود در این لیست مابین خطوط ۱۰ تا ۲۰ تعریف شده است ابتدا در خط ۱۱ وجود آرایه‌ای با عنوان $form _ products مورد بررسی قرار گرفته است (چنانکه متوجه هستید نام موردنظر همانی است که در مورد لیست انتخاب محصولات در خط ۲۲ برنامه استفاده شده است). درصورتی که متغیری با نام فوق موجود باشد، می‌توان چنین فرض کرد که فرم شامل موارد منتخب توسط کاربر به سرور ارسال شده است. همچنین در خط ۱۲ از این برنامه یک بررسی جهت ارزیابی آرایه دیگری با نام $products به‌منظور تعیین اینکه آیا این آرایه تهی است یا خیر انجام می‌شود. درصورتی‌که اصلاً یک چنین آرایه‌ای موجود نبوده و یا درصورت وجود تهی باشد؛ فرآیند معرفی و مقداردهی عناصر آن با استفاده از آرایه شامل محصولات، یعنی $form_ products انجام می‌پذیرد

(خط ۱۳ از برنامه را ببینید). اما درصورتی که آرایه $products موجود باشد به‌واسطه دستیابی قبلی کاربر به سند PHP حاوی این برنامه مقداردهی شده است. به‌هرحال درصورت این دو آرایه، یعنی $products و $form_ products با بهره‌گیری از تابع ( ) array_ merg در خط ۱۶ از این برنامه با یکدیگر تلفیق می‌شوند. ضمناً با استفاده از تابع ( ) array_ unique عناصر منحصر به فرد آرایه حاصل استخراج شده و نتیجه این عملیات مجدداً در قالب $products ذخیره می‌شوند (خط ۱۵ را ببینید). در گام بعدی آرایه $products به‌عنوان یک متغیر جلسه با بهره‌گیری از تابع ( ) session _ register به‌همراه جلسه جاری ثبت می‌گردد (خط ۱۷ را ببینید). همان‌گونه که مشاهده می‌کنید به‌طور مستقیم آرایه $form_ products را به‌همراه جلسه جاری ثبت نکرده و به‌جای این متغیر از $products استفاده کرده‌ایم. این عمل بدان دلیل صورت می‌گیرد که درصورت استفاده از آرایه $form_ products در این فرآیند، به‌واسطه ارسال مجدد فرم مربوطه به سرور برخوردی مابین اسامی متغیرهای همنام پیش‌می‌آمد (از این وضعیت چنانکه ملاحظه کردید با استفاده از فراگیری تابع ( ) array_ uniqe جلوگیری به‌عمل آمده است). در انتهای این برنامه در خط ۳۳ فراپیوندی به یک سند دیگر PHP ایجاد شده است که کاربر به‌واسطه دستیابی به آن می‌تواند امکان بهره‌گیری از متغیرهای جلسه را حین دستیابی به اسناد مختلف موجود در یک وب سایت در عمل مشاهده نماید. عنوان این سند listing20.5.php است که ما در اینجا آن‌را در قالب لیست شماره ۵-۲۰ جهت بررسی بیشتر ارائه دهیم.

```
1: <?php
2: session_start();
3: ?>
4: <html>
5: <head>
6: <title>Listing 20.5 Accessing session variables</title>
7: </head>
8: <body>
9: <h1>A Content Page</h1>
10: <?php
11: if ( isset( $products ) ) {
12:     print "<b>Your cart:</b><ol>\n";
13:     foreach ( $products as $p )
14:         print "<li>$p";
15:     print "</ol>";
16: }
17: ?>
18: <a href="listing20.4.php">Back to product choice page</a>
19: </body>
20: </html>
```

لیست ۵- ۲۰ دسترسی به متغیرهای جلسه

بار دیگر، آن‌چنانکه در این لیست مشاهده می‌کنید با بهره‌گیری از تابع ( ) session_ start از این برنامه جلسه‌ای را که پیشتر در لیست ۵-۲۰ با فراخوانی همین تابع تشکیل داده بودیم از سر

گرفته‌ایم (جهت درک بهتر این وضعیت را می‌توان به جلسات متعددی که در یک شرکت نمونه مابین اعضای هیئت مدیره در مورد یک مسأله بخصوص و در روزهای مختلف تشکیل می‌شود، تشبیه نمود. واضح است که هیچ یک از نتایج برگرفته از جلسات قبل از میان نرفته و کاملاً در اختیار اعضای جلسه قرار می‌گیرد). در خط ۱۱ این برنامه احتمال وجود متغیری با نام $products مورد بررسی قرار گرفته است. در صورتی که یک چنین متغیری موجود باشد با بهره‌گیری از یک ساختار تکرار از نوع foreach در خط ۱۳ برنامه طی هربار گذر از حلقه، کلیه اقلام منتخب توسط هریک از بازدیدکنندگان سند مورد بحث از وب سایت بر روی پنجره مرورگر اینترنت به نمایش در می‌آید.

با این همه در برنامه‌های واقعی مناسب‌تر آن‌است که از ابزارهای منسجم‌تری جهت توسعه استفاده کنیم. در مورد برنامه کارت خرید (shopping cart) که در لیستهای قبل مشاهده کردید، به احتمال زیاد جهت نگهداری جزئیات مربوط به هریک از محصولات از بانک اطلاعاتی که ابزار بسیار قابل اعتمادتری از یک فایل یا بدتر از آن یک یا چند متغیر است استفاده کرده و به‌جای پذیرش کورکورانه مقادیر ورودی و نمایش نتایج، بررسی مناسبی را متناسب با نوع هر ورودی انجام می‌دهیم. اما در کل آنچه که از برنامه‌های ارائه شده در دو لیست ۴-۲۰ و ۵-۲۰ دستگیرمان شد این است که با بهره‌گیری از توابع مربوط به ثبت جلسات می‌توانیم به متغیرها و عناصر آرایه‌هایی که طی جلسات گذشته مقداردهی شده‌اند، دسترسی داشته باشیم.

تخریب جلسات و متغیرهای مربوطه با مقادیر اولیه

تخریب جلسات فرآیند بسیار ساده‌ای بوده و با بهره‌گیری از تابع ( ) session_ destroy قابل انجام است. فراخوانی این تابع علاوه بر تخریب یک جلسه باعث می‌شود تا کلیه متغیرهای مربوط به آن جلسه حذف شود. تابع ( ) session_ destroy دارای ساختار بسیار ساده‌ای بوده به‌گونه‌ای که جهت فراخوانی به آرگومان نیاز ندارد. با این حال برای اینکه نتیجه موردنظر و مطلوب از این فراخوانی حاصل شود لازم است تا در موعد فراخوانی آن قبلاً یک جلسه بازدید تشکیل شده باشد؛ چراکه این تابع در تخریب جلساتی که اصلاً وجود خارجی نداشته باشد منجر به بروز مشکلاتی خواهد شد. قطعه کد زیر چگونگی حذف جلسه را نشان داده است (در این قطعه کد به‌منظور حصول اطمینان از صحت عملیات تخریب ابتدا جلسه‌ای تشکیل شده است) :

```
session_ start ( ) ;
session_ destroy ( );
```

پس از اجرای تابع ( ) session_ destroy چنانچه صفحه دیگری از وب سایت را که جهت کار با جلسات طراحی شده است مورد دستیابی قرار دهیم، متوجه می‌شویم که جلسه تخریب شده دیگر در دسترس نخواهد بود. به‌عبارت دیگر درصورت نیاز به جلسه باید با استفاده از تابع ( ) session_ start یک جلسه جدید ایجاد کنیم. تخریب جلسه توسط تابع ( ) session_ destroy شامل کلیه متغیرهای

ثبت شده با آن جلسه نیز می‌باشد بدین ترتیب نمی‌توانیم از متغیرهای یک جلسه تخریب شده در جلسه جدیدی که ایجاد می‌کنیم، بهره ببریم.

نکته بسیار مهمی که در رابطه با تخریب جلسه با استفاده از تابع ( ) session_ destroy وجود دارد این است که فراخوانی تابع نامبرده موجب از بین رفتن متغیرهای جلسه در همان لحظه نمی‌شود. به‌عبارت دیگر، این‌گونه متغیرها در درون همان برنامه‌ای که جلسه موردنظر با فراخوانی تابع session_ ( ) destroy تخریب شده است کماکان به حیات خویش ادامه می‌دهند (البته این وضعیت تا زمانی که این برنامه مجدداً در قالب سند PHP مربوطه مورد دستیابی مجدد قرار نگرفته است به قوت خود باقی خواهد بود). برای نشان دادن این واقعیت نمونه‌ای را در اینجا ارائه می‌دهیم. قطعه کد زیر با بهره‌گیری از تابع ( ) session_ start اقدام به تشکیل جلسه جدید (یا از سرگیری یک جلسه قدیمی) نموده و متغیری با نام $test را که با عدد صحیح 5 مقداردهی می‌شود با بهره‌گیری از تابع ثبت متغیر جلسه یعنی ( ) session_ register ثبت می‌کند. همان‌گونه که مشاهده می‌کنید فراخوانی تابع ( ) print در مورد متغیر مذکور عدد صحیح 5 را بر روی صفحه نمایش می‌دهد:

```
session_ start ( ) ;
session _ register ( " test " ) ;
$test = 5 ;
session _ destroy ( ) ;
print $test ;   // prints 5
```

آنچه از این قطعه برنامه برداشت می‌کنیم این است که فراخوانی تابع تخریب session_ ( ) destroy موجب از دست رفتن آنی متغیر $test نمی‌شود. با این همه شاید در برخی موارد لازم است تا این رفتار تغییر کرده و تخریب جلسه بلافاصله موجب حذف متغیرهای مربوط به آن نیز شود. جهت انجام این کار کافی است تابعی با نام ( ) session _ unset را در برنامه خود فرابخوانیم. این فراخوانی باعث حذف کلیه متغیرهایی خواهد شد که به‌همراه این جلسه تعریف شده‌اند. توجه کنید که تاثیر این فرآیند علاوه بر اینکه در برنامه مشهود است در فایلی که شامل اطلاعات جلسه است نیز مشخص خواهد بود. بدین ترتیب باید اعتراف کردکه تابع( ) session _ unset موجب تغییرات اساسی می‌شود. لذا توصیه می‌کنیم که هنگام استفاده از آن نهایت دقت را به‌خرج دهید. قطعه کد زیر مشابه قطعه کد قبلی است با این تفاوت که این بار با بهره‌گیری از این تابع اقدامی جهت حذف متغیر جلسه $test صورت گرفته است. فراخوانی تابع ( ) print این گفته را تأیید می‌کند:

```
session _ start ( ) ;
session _ register ( " test " ) ;
$test = 5 ;
session _ unset ( );
session _ destroy ( );
print $test ;   // prints nothing. The $test variable is no more
```

چنانکه در این قطعه برنامه ملاحظه می‌کنید پیش از تخریب، جلسه session _ unset ( ) فراخوانی شده است. همین عمل کافی است تا کلیه متغیرهایی که پیشتر به همراه این جلسه ثبت شده بودند (در این مورد تنها متغیر $test) از صحنه برنامه حذف شوند. عبارت print ( ) بدین ترتیب هیچ خروجی دربر نخواهد داشت.

بهره‌گیری از شناسه جلسات در دنباله‌های پرس و جو

تا بدین جا همان‌گونه که مشاهده کردید جهت حفظ و انتقال شناسه جلسات از یک سند به سند دیگر از کوکی‌ها استفاده کردیم. کوکی‌ها را به‌خودی خود نمی‌توان روش قابل اعتمادی برای ثبت وضعیت و اطلاعات مربوطه دانست؛ چراکه همواره این امکان وجود دارد که بازدیدکننده با تنظیم گزینه‌هایی از مرورگر اینترنت مورد استفاده خود آنها را غیر فعال نماید و بدین ترتیب از پذیرش و ذخیره آنها بر روی کامپیوترش جلوگیری به‌عمل آورد. با این حال همواره می‌توان شناسه جلسات را در قالب دنباله‌های پرس و جو مابین برنامه‌های اسکریپت مختلف موجود در وب سایت ارسال نمود. درصورتی که کوکی موردنظر به‌ازای شناسه جلسه مشخصی یافت نشود، PHP یک زوج متشکل از نام و مقدار مربوطه را در قالب ثابتی با عنوان SID در دسترس قرار می‌دهد. از این مقدار ثابت می‌توان در هر فراپیوندی که مقصد آن سندی با پشتیبانی جلسات است، استفاده کرد. به نمونه زیر توجه کنید:

```
<a href=anotherpage.html?PHP Print SID; ?>">Another page </a>
```

بهره‌گیری از فرا پیوند فوق موجب خواهد شد تا مرورگر آن را به صورت زیر تلقی نماید.

```
< a href = "anotherpage . html? PHPSESSID = 08ecedf 79 fe34561 fa
    82591401a01da1" > Another Page < / a >
```

شناسه جلسه‌ای که بدین روش ارسال می‌شود به‌طور خودکار توسط سند مقصد و با بهره‌گیری از تابع session _ start ( ) قابل دستیابی خواهد بود. بدین ترتیب می‌توان متغیرهای جلسه را به ترتیبی که در حالت عادی دسترسی داشتیم، مورد دستیابی و استفاده قرار دهیم.

در صورتی که PHP4 را با استفاده از گزینه -enable - trans - sid - نصب کرده باشید، متوجه می‌شوید که این دنباله پرس و جو به‌طور خودکار به کلیه فراپیوندهای مربوط به صفحات وب سایت ضمیمه خواهد شد. بنا به پیش‌فرض این گزینه در حالت عادی غیر فعال است. با این حال استفاده از ثابت SID به‌طور صریح با فراپیوندهای موردنظر موجب افزایش قابلیت حمل برنامه‌ها خواهد شد.

کدگذاری و رمزگشایی متغیرهای جلسه

تاکنون متوجه شده‌اید که PHP پیش‌از آنکه اقدام به ذخیره متغیرهای جلسه نماید آنها را به شیوه خاصی کدگذاری می‌کند. این فرآیند هر مرتبه‌ای که بازدیدکننده اقدام به دستیابی به سندی که

جهت بهره‌گیری از جلسات به تابع session _ start ( ) مجهز شده باشد، روی می‌دهد. با این وجود درصورتی‌که برنامه نویس مایل باشد می‌تواند با فراخوانی تابع ویژه‌ای که جهت این کار طراحی شده و session _ encode ( ) نام دارد به دنباله کاراکتری کدگذاری شده دست پیدا کند. استفاده از این تابع معمولاً در مواردی چون اشکال‌زدایی برنامه‌هایی که از جلسات بازدید پشتیبانی به‌عمل می‌آورند، مفید واقع می‌شود. با بهره‌گیری از تابع( ) session _ encode می‌توان وضعیت کلیه متغیرهای جلسه موجود را مورد مشاهده و بررسی قرار داد. در قطعه کد زیر از تابع فوق به این منظور استفاده شده است. به‌خروجی حاصل از اجرای این قطعه کد توجه کنید:

```
session _ start ( ) ;
print session _ encode ( ) . "< br >" ;
// sample output : products | a : 2 : { i : 0 ; s : 8 : "Hall 2000" ;
//   i : 1 ; s : 6 : "Tradis" ;}
```

با بررسی خروجی حاصل از این قطعه کد می‌توان متوجه این نکته شد که چه متغیرهای جلسه‌ای هم‌اکنون به‌همراه جلسه موردنظر ثبت شده‌اند. از این اطلاعات مفید می‌توان جهت بررسی این مطلب که آیا متغیرهای جلسه موردنظر به همان ترتیبی که مورد نظر ما بوده ثبت و به‌روز رسانی شده‌اند یا خیر. تابع session _ encode ( ) استفاده مفید دیگری نیز دارد و آن‌اینکه از این تابع می‌توان در فرآیند ذخیره متغیرهای جلسه در یک بانک اطلاعاتی یا یک فایل نیز استفاده کرد.

پس از استخراج دنباله کاراکتری کدگذاری شده موردنظر با استفاده از تابع دیگری با عنوان session _ decode ( ) می‌توان فرآیند معکوس، یعنی فرآیند رمزگشایی را بر روی آن انجام داد. به این ترتیب می‌توان مقادیر اصلی را در قالب پیش از رمزگذاری بازیابی نمود. در قطعه کد زیر این فرآیند را نشان می‌دهد:

```
session _ start ( ) ;
session _ unset ( ) ;  // there should now be no session variables
session _ decode ( "products | a : 2 : { i : 0 ; s : 8 : \ "Hal 2000 \"  ; i : 1 ;
      s : 6 : \ "Tradis \" ;}" ) ;
foreach ($products as  $p) {
     print "$p < br > \ n" ;
}
// output :
// Hall 2000
// Tradis
```

چنانکه در این قطعه کد مشاهده می‌کنید ابتدا یک جلسه بازدید با بهره‌گیری از تابع session_ start ( ) تشکیل شده است. سپس تابع session _ unset ( ) را به منظور reset کردن متغیرهای جلسه موجود فراخوانی کرده‌ایم. این فراخوانی شاید به‌نظر بی‌مورد بیاید اما درحقیقت جهت اطمینان از اینکه هیچ‌گونه متغیر جلسه‌ای به‌همراه جلسه جاری موجود نیست، اقدام مفیدی محسوب می‌شود. درگام بعدی یک دنباله کاراکتری کدگذاری شده را به‌عنوان آرگومان ورودی تابع رمزگشایی session _

( ) decode به این تابع ارسال کرده‌ایم. تابع( ) session _ decode به‌جای اینکه مقداری را به برنامه فراخواننده بازگرداند اقدامی را جهت تبدیل دنباله کاراکتری کدگذاری شده صورت داده و آن‌را به فرم اصلی تبدیل می‌کند. فراخوانی تابع ( ) print در درون ساختار تکرار foreach به‌ازای هریک از عناصر آرایه $products که اکنون شامل مقادیر کدگشایی شده است، نشان می‌دهد که تابع session _ ( ) decode در فرآیند رمزگشایی خود موفق بوده است.

بررسی متغیرهای جلسه

همان‌گونه که تا بدین جا مشاهده کردید هنگام نیاز به اطلاع از این مطلب که آیا متغیر جلسه خاصی موجود می‌باشد یا خیر، یا به‌عبارت دیگر اطلاع از اینکه آیا متغیری به همراه جلسه خاص ثبت شده است یا خیر، از تابع ( ) isset استفاده نمودیم. با این حال روش دیگری نیز برای این کار وجود دارد و آن استفاده از تابعی با نام ( ) session _ is _ registered است. تابع فوق را نیز می‌توان جهت بررسی وضعیت ثبت متغیرها با یک جلسه مورد بهره‌برداری قرارداد. این تابع ساختار بسیار ساده‌ای از نظر فراخوانی دارد به‌گونه‌ای که تنها از یک آرگومان ورودی جهت انجام این بررسی استفاده می‌کند. آرگومان این تابع دنباله کاراکتری است که نماینده متغیر مورد بررسی می‌باشد. این تابع چنانچه متغیر مذکور به‌همراه جلسه فعلی ثبت شده باشد مقدار true و در غیر این‌صورت مقدار false را به برنامه فراخواننده باز می‌گرداند. به‌نمونه‌ای از چگونگی استفاده از این تابع توجه نمایید:

```
If (session _ is_ registered ( "products " ))
    Print " 'producte' is registered! " ;
```

تابع ( ) session _ is _ registered جهت اطمینان از منبع متغیر موردنظر بسیار مفید است. در بسیاری از موارد لازم است تا مطمئن شویم که متغیر مورد بررسی به‌عنوان یک متغیر جلسه در دسترس ما قرار دارد. این وضعیت نقطه مقابل داده‌های ارسالی به سرور به عنوان بخشی از یک درخواست GET است.

بهره‌مندی از آرایه $HTTP _ SESSION _ VARS

در رابطه با استفاده از متغیرهای جلسه یک نکته ناگفته باقی‌مانده است و آن اینکه چنان چه گزینه‌ای با عنوان register _ globals را از فایل تنظیمات php. ini با مقدار off یا صفر تنظیم کرده باشید قادر نخواهید بود تا مستقیماً از متغیرهای جلسه بهره ببرید. در چنین حالتی مجبور خواهید بود تا از طریق آرایه $HTTP _ SESSION _ VARS اقدام به تنظیم و دستیابی متغیرهای جلسه موردنیاز خود نمایید. در مورد پروژه‌های برنامه‌نویسی بزرگ، این وضعیت می‌تواند مصلحت آمیز باشد؛ چراکه استفاده از این آرایه در کاربرد مذکور می‌تواند خطرناشی از هم‌نام بودن متغیرهای سراسری را با

متغیرهای جلسه از میان بردارد. از این‌رو جهت تنظیم یک متغیر جلسه با نام test به‌گونه‌ای که شامل مقدار عدد صحیح 5 باشد مانند قبل از فراخوانی تابع ( ) session _ register استفاده می‌کنیم اما این تابع را تنها به‌منظور ثبت متغیر و نه مقداردهی آن به‌کار می‌گیریم. جهت تنظیم مقدار چنین متغیری از آرایه $HTTP _ SESSION _ VARS به‌صورتی که در قطعه کد زیر مشاهده می‌کنید، استفاده می‌کنیم:

```
session _ start ( ) ;
session _ register ( " test " ) ;
$HTTP _ SESSION _ VARS [ ' test ' ] = 5 ;
```

جهت دستیابی به متغیر جلسه در سایر برنامه‌های اسکریپت مربوط به این وب سایت بار دیگر می‌توان از این آرایه استفاده کرد. قطعه کد زیر چگونگی انجام این فرآیند را نشان می دهد:

```
session _ start ( ) ;
print  " test is . .  " ;
print $HTTP _ SESSION _ VARS [ ' test ' ] ;
```

جمع‌بندی

در درس این ساعت و ساعت قبل چنانکه دیدید روشهای مختلفی را که برنامه‌نویسان PHP در عرصه وب جهت ثبت وضعیت بازدیدکنندگان از وب سایت (به‌واسطه قرارداد بدون وضعیت HTTP) مورد بهره‌برداری قرار می‌دهند، معرفی کرده و مزایا و معایب هر یک را تحت بررسی و گفتگو قرار دادیم. در تمامی این روشها کم و بیش استفاده از کوکی و همچنین دنباله‌های پرس وجو به‌چشم می‌خوردند. برخی از روشها نیز علاوه بر این از فایل‌ها و بانکهای اطلاعاتی جهت ثبت و نگهداری اطلاعات وضعیت استفاده می‌کردند. هرکدام از این روشها نقاط ضعف و قوتی داشتند که به‌موقع مورد بررسی قرار گرفتند.

اکنون با دانشی که درمورد کوکی‌ها به دست آورده‌اید، تصدیق می‌کنید که نمی‌توان به این ابزار متداول جهت ثبت وضعیت به‌خودی خود اعتماد نمود. همچنین نمی‌توان با استفاده از آن حجم زیادی از اطلاعات مربوط به وضعیت را ذخیره کرد. اما در عین حال می‌توان آن‌را ابزار مناسبی جهت ذخیره اطلاعات به مدت طولانی به‌حساب آورد.

سایر رویکردها در رابطه با ثبت و نگهداری اطلاعات وضعیت استفاده از فایل‌ها و بانکهای اطلاعاتی است. بهره‌گیری از فایل یا بانک اطلاعاتی جهت ثبت اطلاعات آن هم به‌صورت دائمی، مدت‌دار و البته قابل اطمینان بسیار ایده‌آل است؛ اما به‌دلیل صرف زمان مورد نیاز جهت برقراری اتصال با بانک اطلاعاتی یا بازکردن فایل موجود در سیستم فایل (و یا ایجاد آن در صورت عدم وجود) ممکن است نارضایتی‌هایی را در رابطه با کارایی وب سایت در پی داشته باشد. با این وجود همواره می‌توان

به‌واسطه بخش کوچکی از اطلاعات، مثل شناسه مربوط به یک جلسه خاص اطلاعات بسیاری را درمورد آن به دست آورد.

رویکرد آخری که طی این دو درس با هم تحت بررسی قرار دادیم بهره‌گیری از دنباله‌های پرس و جو بود. همان‌گونه که تا به حال متوجه شدید دنباله‌های پرس و جو برخلاف روش کوکی و البته بانک‌اطلاعاتی و فایل توانایی ذخیره دائمی اطلاعات را ندارند. معمولاً کاربران (و برخی از برنامه‌نویسان) به واسطه ظاهر نامانوس و بدترکیب آن، که به نحو خاصی در فیلد آدرس مرورگر اینترنت جلوه نمایی می‌کند از به‌کارگیری آن صرف‌نظر می‌کنند. با این همه به کمک دنباله‌های پرس و جو می‌توان حجم بزرگی از اطلاعات را مابین اسنادی از یک وب سایت که بر اساس جلسات کار می‌کنند، رد و بدل کرد. تصمیم‌گیری درباره اینکه کدام روش بر دیگری ارجحیت دارد بستگی به شرایط مسأله و نیز عواملی که برنامه نهایی باید قادر به تامین آن باشد، دارد.

در درس این ساعت چگونگی تشکیل یک جلسه جدید را با بهره‌گیری از تابع session_ start( ) مشاهده کردید. همچنین متوجه شدید که تابع مذکور علاوه بر تشکیل جلسات توانایی از سرگیری جلسات قدیمی را نیز دارد. ملاحظه کردید که می‌توان با استفاده از تابع session_ register ( ) متغیرهایی را (که به متغیرهای جلسه شهرت دارند) با جلسه موردنظر ثبت نمود. پس از ثبت یک چنین متغیرهایی می‌توان در صفحاتی از وب سایت که جهت کار با جلسات مجهز شده‌اند به این‌گونه متغیرها دسترسی پیدا کرد. همچنین مشاهده کردید که به‌منظور اطلاع از اینکه متغیر خاصی به‌همراه جلسه جاری ثبت شده است یا خیر، می‌توان همیشه از تابع session _ is _ registered ( ) استفاده کرد. تابع مذکور در صورت ثبت متغیر مورد نظر مقدار true را به برنامه فراخواننده بازمی‌گرداند. در رابطه با جلسات دو تابع دیگر نیز مورد بررسی قرار گرفتند. تابع اول با عنوان session _ unset ( ) به‌منظور حذف متغیرهای موردنظر مورد استفاده قرار می‌گیرد. همچنین تابع session _ destroy ( ) نیز جهت تخریب جلسه جاری به کار گرفته می‌شود. چنانکه ملاحظه کردید تخریب جلسه با استفاده از این تابع موجب حذف آنی متغیرهای موجود در آن جلسه نمی‌شود بلکه تأثیر حذف آنها به‌واسطه دستیابی مجدد به سند مربوطه مشخص می‌گردد.

نکته آخری که در درس این ساعت بررسی شد این بود که جهت اطمینان از اینکه کاربران بیشتری در حد امکان از قابلیتهای پشتیبانی از جلسات در وب سایت شما بهره‌مند می‌شوند، می‌توانید از ثابت سیستمی خاصی با عنوان SID جهت ارسال یک شناسه منحصر به فرد جلسه به سرور در قالب یک دنباله پرس و جو استفاده نمایید.

در درس ساعت آینده با عنوان " بهره‌گیری از محیط سرور" به بررسی روشهایی خواهیم پرداخت که با استفاده از آنها از طریق برنامه‌های PHP می‌توانید سایر امکانات و ابزارهای موجود بر روی سرور را مورد دستیابی قرار دهید.

پرسش و پاسخ

پرسش: آیا استفاده از توابع مربوط به ثبت جلسات مستلزم توجه نکته خاص و قابل ذکری می‌باشد؟

پاسخ: توابع مربوط به ثبت جلسات که در درس این ساعت مورد بررسی قرار دادیم عموماً از ضریب اطمینان بالایی برخوردارند. با این وجود لازم است به‌خاطر داشته باشید که کوکی‌ها را نمی‌توان از طریق اسامی حوزه‌های مختلف مورد دستیابی و بازخوانی قرار داد. از این جهت درصورتی‌که پروژه شما مستلزم دستیابی به اطلاعات جلسات از طریق چند حوزه مختلف باشد (چنین وضعیتی را می‌توان در وب سایت‌هایی که قالب e _ commerce دارند، مشاهده نمود) به احتمال قوی لازم است تا بهره‌گیری از کوکی‌ها را غیر فعال نمایید. برای انجام یک چنین کاری لازم است تا گزینه ویژه‌ای از فایل تنظیمات php. ini با عنوان session . use _ cookie را با مقدار عددی صفر تنظیم نمایید.

تمرینها

هدف‌ازاین‌بخش ارائه تمرینهایی در قالب آزمون می‌باشد. پاسخ بخش آزمون هر درس بلافاصله پس‌ازآن‌آمده است. بخش فعالیتها شامل تمرینهایی است که جهت تقویت مهارت و قابلیت برنامه‌نویسی خواننده طراحی شده است و فاقد پاسخ لازم می‌باشد.

آزمون

۱- از کدام تابع می‌توان جهت تشکیل یک جلسه جدید یا از سرگیری یک جلسه قدیمی استفاده کرد؟

۲- با بهره‌گیری از کدام تابع می‌توان به شناسه جلسه فعلی پی‌برد؟

۳- چگونه می‌توان متغیری را به‌همراه یک جلسه ثبت نمود؟

۴- چگونه می‌توان جلسه‌ای را تخریب کرده و ردپای آن‌را در دستیابی‌های آتی از بین برد؟

۵- چگونه می‌توان متغیرهای جلسه‌ای را حذف کرد به‌گونه‌ای که حتی در برنامه فعلی نیز نتوان تأثیری از متغیرهای همراه شده با آن جلسه را مشاهده کرد؟

۶- ثابت SID نمایانگر چیست؟

۷- چگونه می‌توان در مورد ثبت شدن متغیری با نام $test به‌همراه جلسه جاری اطلاع حاصل نمود؟

پاسخ آزمون

۱- با بهره‌گیری از تابع ( ) session _ start می‌توان جلسه‌ای را تشکیل داد.
۲- به کمک تابع ( ) session _ id می‌توان شناسه یک جلسه را مورد دستیابی قرار داد.
۳- با استفاده از تابع ( ) session _ register می‌توان متغیر خاصی را که نام آن به‌عنوان آرگومان به این تابع ارسال می‌شود به‌همراه جلسه جاری ثبت نمود.
۴- با بهره‌گیری از تابع ( ) session _ register می‌توان ترتیبی داد که جلسه فعلی تخریب شده و در دستیابی‌های بعدی به سند PHP مربوطه اثری از آن دیده نشود.
۵- با استفاده از تابع ( ) session _ unset می‌توان متغیرهای جلسه جاری را به‌گونه‌ای حذف کرد که حتی برنامه در حال اجرا نیز اثری از آنها را مشاهده نکند.
۶- درصورتی‌که کوکی‌ها در دسترس نباشند ثابت سیستمی SID شامل زوجی متشکل از نام و مقدار متناظر آن خواهد بود. از این زوج می‌توان جهت دستیابی به اطلاعات بیشتر در مورد جلسه در دنباله پرس و جو استفاده نمود. بدین ترتیب می‌توان شناسه جلسه را از یک درخواست به دیگری ارسال کرد.
۷- جهت اطلاع از اینکه آیا متغیر خاصی به‌همراه یک جلسه ثبت شده است یا خیر همواره می‌توان تابع ( ) session _ is_ registered را با آرگومان متغیر موردنظر به صورتی که در زیر مشاهده می‌کنید، فراخوانی نمود:

```
Session _ is _ registered ( "test " ) ;
```

فعالیتها

۱- چنانکه خاطرتان است در بخش فعالیتهای درس گذشته برنامه‌ای نوشتید که با بهره‌گرفتن از کوکی یا دنباله پرس و جو اقدامی را جهت ثبت و نگهداری علایق و سلیقه‌های شخصی بازدیدکنندگان وب سایت حین بازدید آنها از صفحات مختلف سایت صورت دادید. درصورتی که فعالیت مزبور را به‌درستی انجام داده باشید اکنون بازدیدکنندگان باید رنگ دلخواهی را که به‌عنوان پس‌زمینه موردنظرشان انتخاب کرده‌اند به همراه یک پیغام خوش‌آمدگویی که شامل نام کاربری آنهاست، مشاهده کنند. به‌عنوان اولین فعالیت این درس تمرین فوق را تکرار کنید با این تفاوت که این‌بار به‌جای بهره‌گیری از کوکی یا دنباله پرس و جو از توابع معرفی شده در این درس استفاده نمایید.

۲- برنامه‌ای ایجاد کنید که قادر باشد تا بهره‌گیری ازتوابع مربوط به ثبت جلسات صفحاتی از وب سایت را که هر کاربر مورد بازدید قرار داده‌است، به‌خاطر بسپارید.

این برنامه باید اسامی (یا عناوین) اسنادی را که هر کاربر مورد بازدید قرار داده است در قالب فراپیوندهایی همواره در مقابل دیدگان وی قرار داده و بدین ترتیب امکان بازدید مجدد وی از صفحات مذکور را به‌راحتی فراهم نماید.

ساعت بیست و یکم

بهره گیری از محیط سرور

در ساعات گذشته چنانکه شاهد بودید در مورد روشها و تکنیکهای مؤثری جهت ارتباط با ماشینهای راه دور و همچنین دستیابی به مقادیری که کاربر به‌عنوان ورودی مشخص کرده است، بحث و گفتگو کردیم. در درس این ساعت بحث ویژه‌ای را در مورد اجرای برنامه‌ها باز خواهیم کرد. آنچه در درس این ساعت راجع به آن گفتگو می‌کنیم، تکنیکهایی است که به ما اجازه می‌دهند تا با بهره‌گیری از آنها برنامه‌های خارجی موجود بر روی کامپیوتر را اجرا نماییم. برنامه‌های نمونه موجود در این ساعت به‌منظور اجرا بر روی سیستم عامل Linux توسعه پیدا کرده‌اند. با این حال اصول به کاررفته را درمورد سیستم عامل windows نیز می‌توان مورد استفاده قرار داد.

در درس این ساعت مطالب زیر را مورد بحث و بررسی قرار خواهیم داد:

- **چگونگی ارسال داده‌ها یا دریافت آنها از برنامه‌های خارجی**
- **بررسی سایر روشهای ممکن برای ارسال دستورالعملها و فرمانهای shell و نمایش حاصل از اجرای آنها بر روی صفحه مرورگر اینترنت**
- **بررسی‌مسائل‌امنیتی‌موجوددررابطه‌باارتباطات بین فرآیندی Interprocess Communication یا به اختصار IPC در برنامه‌های PHP**

در ادامه به بررسی این موارد می‌پردازیم.

بهره‌گیری از تکنیک خط لوله یا پایپ جهت برقراری ارتباط با فرآیندهای موجود درحال اجرا بر روی سیستم با تابع ( ) popen

همان‌گونه که با بهره‌گیری از تابع ( ) Popen که در درسهای گذشته بررسی کردیم، می‌توانیم فایل‌های مورد نظرمان را جهت نوشتن یا خواندن باز کنیم؛ با بهره‌گیری از تابع دیگری با عنوان ( ) popen قادریم تا خط لوله‌ای را جهت برقراری ارتباط با یک فرآیند موجود ایجاد نماییم.

مبحث ارتباطات بین فرآیندی یا IPC از جمله مباحث پیشرفته هر زبان برنامه‌نویسی محسوب می‌شود و جهت درک هر چه بهتر و دقیق‌تر آن ابتدا لازم است تا خواننده مفهوم اساسی این مبحث را که همان فرآیند یا process است، درک نماید. در این میان شاید مناسب‌ترین تعبیری که از این مفهوم اساسی در کامپیوتر و به‌ویژه سیستم عامل شده است، همان تعبیری است که در صفحه ۷۳ از کتاب مشهور آقای پروفسور Andrew S . Tanenbaum با عنوان "سیستمهای عامل مدرن" به چشم می‌خورد و ما جهت سادگی مراجعه دراینجا آن‌را بیان می‌کنیم:

" ... تفاوت مابین دو مفهوم فرآیند و برنامه (به ترتیب process و program) بسیار ظریف و در عین حال بسیار حیاتی و با اهمیت است و شاید یک تشبیه در این مورد بتواند نکات تاریک و مبهم را تا اندازه زیادی روشن کند. درنظر بگیرید که شخصی درحال طبخ یک کیک برای مراسم جشن تولد دخترش باشد. آشپزخانه و لوازم آشپزی کاملاً مهیا بوده و مواد لازم جهت طبخ در دسترس است. این مواد عبارتند: از آرد، تخم مرغ، شکر، وانیل، شیر و سایر مواد. این شخص با دنباله کردن دستورالعملهای موجود در یک کتاب آشپزی در نهایت هدف پختن کیک را دنبال می‌کند. در این تشبیه دستورالعملهای آشپزی مشابه برنامه یا همان الگوریتمی است که‌به‌طریقی‌دقیق بیان می‌شوند؛ همچنین خود شخص موردنظر را می‌توان به پردازنده کامپیوتر (CPU) و مواد لازم برای طبخ کیک را نیز به داده‌های ورودی به کامپیوتر تشبیه نمود. بنابراین آنچه که می‌توان در مورد این تشبیه به‌صراحت بیان کرد، این است که فرآیند مورد نظر در اینجا عبارت است از فعالیتی که طی آن شخص مورد بحث ما دستورالعملهای موردنیاز جهت طبخ کیک را از کتاب آشپزی خوانده و برمبنای این دستورالعملها مواد مورد نیاز را با یکدیگر به‌صورتی که در کتاب ذکر شده با یکدیگر ترکیب نموده و درنهایت کیک را آماده می‌کند. " آنچه که از این تشبیه دستگیرمان می‌شود این است که "برنامه" معادل دقیقی برای "فرآیند" نبوده و تنها بخشی از آن را تشکیل می‌دهد ـ مترجم.

استفاده از تابع ( ) popen مستلزم تأمین دو آرگومان برای آن است. آرگومان اول یک دنباله کاراکتری است که مسیر فرمان مورد نظر بر روی سیستم را مشخص می‌کند (از آنجا که هر فرمان متناظر با یک برنامه اجرایی همنام با آن است، شاید بهتر باشد بگوییم این آرگومان مسیر برنامه موردنظر بر روی سیستم فایل را مشخص می‌کند. این طرز بیان با تشبیه ما درمورد مفهوم برنامه فرآیند که در کادر فوق بیان کردیم، همخوانی کامل دارد). آرگومان دوم که آن‌هم از نوع دنباله کاراکتری است نماینده یکی از دو حالت خواندن یا نوشتن است. این تابع به‌عنوان نتیجه عملیات خود مرجعی به یک فایل را بازمی‌گرداند. از این مرجع بازگشتی می‌توان دقیقاً به همان صورتی‌که از مرجع بازگشتی به یک فایل که در اثر فراخوانی تابع ( ) Popen ایجاد می‌شود، استفاده کردیم و از نمونه‌هایی از آن که در درسهای گذشته مشاهده نمودید، استفاده می‌کنیم. به‌عنوان آرگومان دوم تابع ( ) popen می‌توانیم یکی از دو مقدار " w " یا " r " را به‌ترتیب به نشانه نوشتن یا خواندن در قالب یک دنباله کاراکتری مورد بهره‌برداری قرار دهیم. نکته مهم در این مورد این است که نمی‌توان عمل خواندن و نوشتن را به‌طور توأم در یک فراخوانی این تابع انجام داد.

پس از پایان انجام کار موردنظر با فایلی که مرجع آن با فراخوانی تابع ( ) popen در اختیار قرار گرفته است، لازم است تا به‌روشی ارتباط خود را با آن قطع کنیم. در این مورد می‌توانیم این عمل را با فراخوانی تابع دیگری با نام ( ) pclose انجام دهیم. ساختار این تابع بسیار ساده بوده و جهت فراخوانی آن کافی است تا مرجع فایل موردنظر را به‌عنوان آرگومان ورودی ارسال کنیم.

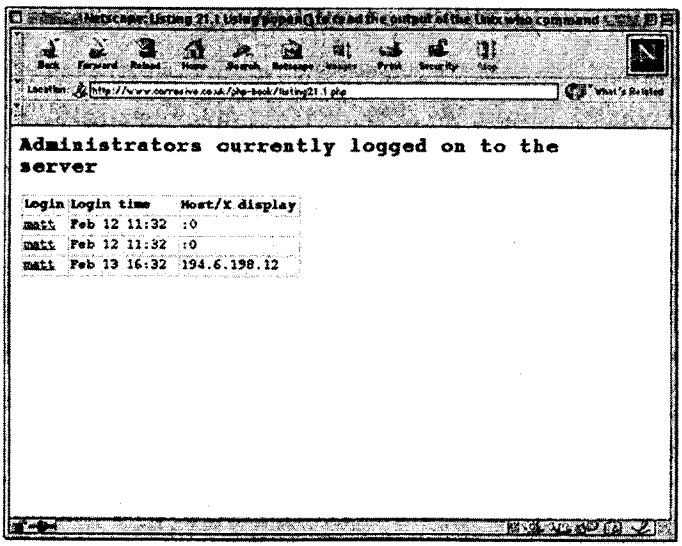
عمل خواندن از طریق فراخوانی تابع ( ) popen که با تعیین آرگومان دوم این تابع به‌صورت "r" مشخص می‌شود، معمولاً در مواردی مفید واقع می‌شود که بخواهیم خروجی حاصل از یک فرآیند را به‌صورت خط به خط مورد بررسی قرار دهیم. برنامه‌ای را که در لیست ۱- ۲۱ مشاهده می‌کنید با استفاده از تابع ( ) popen ارتباطی با نسخه GNU از برنامه who برقرار کرده و پیوندی از نوع mailto را به اسامی هر یک از کاربرانی که به‌واسطه اجرای این برنامه در دسترس قرار می‌گیرند، ضمیمه می‌کند.

```
 1: <html>
 2: <head>
 3: <title>Listing 21.1 Using popen() to read the
 4:        output of the Unix who command</title>
 5: </head>
 6: <body>
 7: <h2>Administrators currently logged on to the server</h1>
 8: <?php
 9: $ph = popen( "who", "r" )
10:         or die( "Couldn't open connection to 'who' command" );
11: $host="corrosive.co.uk";
12: while ( ! feof( $ph )  ) {
13:     $line = fgets( $ph, 1024 );
14:     if ( strlen( $line ) <= 1 )
15:         continue;
16:     $line = ereg_replace(     "^([a-zA-Z0-9_\-]+).*",
```

```
17:                 "<a href=\"mailto:\\1@$host\">\\1</a><BR>\n",
18:                 $line );
19:     print "$line";
20: }
21: pclose( $ph );
22: ?>
23: </table>
24: </body>
25: </html>
```

لیست ۱- ۲۱ بهره‌گیری از تابع ( ) popen جهت بازخوانی خروجی حاصل از فرمان who یک سیستم UNIX

چنانکه در این لیست مشاهده می‌کنید ما با بهره‌گیری از تابع ( ) popen در خط ۹ مرجعی به فایل حاوی برنامه who ایجاد نموده و سپس با استفاده از یک ساختار تکرار از نوع while در خط ۱۲ از برنامه اقدامی را جهت بازخوانی هریک از خطوط خروجی حاصل از فرآیند صورت داده‌ایم. درصورتی‌که خروجی تنها متشکل از یک کاراکتر تنها باشد از ادامه کار حلقه جاری از این ساختار تکرار چشم‌پوشی کرده و کار خود را به‌ازای حلقه بعدی از سر می‌گیریم (خطوط ۱۴ و ۱۵ از برنامه را ببینید). اما چنان‌چه طول دنباله کاراکتری خروجی بیش از یک کاراکتر باشد با استفاده از تابع ( ) ereg _ replace در خط ۱۶ برنامه فراپیوندی از نوع HTML را پیش از نمایش خروجی در خط ۱۹ به دنباله کاراکتری مربوطه اضافه می‌کنیم و درنهایت پس از اتمام عملیات موردنظر در ساختار تکرار while با فراخوانی تابع ( ) pclose ارتباط مابین برنامه جاری و برنامه who را خاتمه می‌دهیم. خروجی حاصل از این لیست را می‌توانید در شکل ۱- ۲۱ مشاهده می‌کنید.



شکل ۱- ۲۱ بازخوانی خروجی حاصل از اجرای فرمان who در یک سیستم UNIX

علاوه بر خواندن با استفاده از تابع ( ) popen می‌توانید امکانات نوشتن در یک فرآیند نمونه را نیز فراهم نمایید. عمل نوشتن در یک فرآیند که با ارسال دنباله کاراکتری " w " به‌عنوان دومین آرگومان تابع ( ) popen انجام می‌شود معمولاً در مورد برنامه‌ها یا فرمانهایی مفید است که مستلزم دریافت داده‌ها از طریق ورودی استاندارد می‌باشند (چنانکه ملاحظه کردید برخی از فرمانها مانند who فاقد ورودی هستند اما برخی دیگر از فرمانها علاوه بر آرگومان‌های ویژه‌ای که جهت تعیین نحوه عملکرد فرمان لازم است، مستلزم دریافت ورودی هستند. به‌عنوان نمونه در سیستم عامل windows فرمان Dir فاقد ورودی ولی فرمان Del مستلزم دریافت ورودی می‌باشند). برنامه موجود در لیست ۲- ۲۱ با بهره‌گیری از تابع ( ) popen اقدام به برقراری ارتباطی با برنامه column که نیازمند دریافت ورودی است، می‌نماید.

```
 1: <html>
 2: <head>
 3: <title>Listing 21.2 Using popen() to pass
 4:        data to the column command</title>
 5: </head>
 6: <body>
 7: <?php
 8: $products = array(
 9:         array( "HAL 2000", 2, "red" ),
10:         array( "Tricorder",  3, "blue" ),
11:         array( "ORAC AI", 1, "pink"  ),
12:         array( "Sonic Screwdriver", 1, "orange"   )
13:         );
14: $ph = popen( "column -tc 3 -s / > purchases/user3.txt", "w" )
15:     or die( "Couldn't open connection to 'column' command" );
16: foreach ( $products as $prod )
17:     fputs( $ph, join('/', $prod)."\n");
18: pclose( $ph );
19: ?>
20: </table>
21: </body>
22: </html>
```

لیست ۲- ۲۱ بهره گیری از تابع ( ) popen جهت ارسال داده‌های ورودی موردنیاز برنامه column در یک سیستم UNIX

چنانکه ملاحظه می‌کنید هدف برنامه موجود در لیست فوق این است که عناصر یک آرایه چند بعدی را (که در خط ۸ تعریف شده است) دریافت کرده و آنها را به‌عنوان خروجی در قالب یک جدول ASCII به یک فایل ارسال نماید. برای این منظور برنامه ابتدا در خط ۱۴ با استفاده از تابع ( ) popen که این‌بار به‌منظور نوشتن در فرآیند فراخوانی می‌شود (به دومین آرگومان این تابع توجه کنید) ارتباطی را با فرمان column برقرار می‌کند. همان‌گونه که در اولین آرگومان تابع ( ) popen مشاهده می‌کنید، فرمان column به چندین ابزار مختلف (آرگومان‌های سطر فرمان) جهت اجرا مجهز شده است. آرگومان t - باعث می‌شود که خروجی حاصل از اجرای فرمان column در قالب یک جدول

نمایش داده شود. همچنین آرگومان سطر فرمان c - که با مقدار عددی 3 به‌صورت c 3- مشخص شده است، باعث خواهد شد تا جدول خروجی داده‌های موردنظر را در سه ستون قالب‌بندی نماید و بالاخره آرگومان سطر فرمان s- با تعیین کاراکتر / به شکل / s - باعث می‌شود تا کاراکتر فوق مبنای جداسازی فیلدها در قالب‌بندی موردنظر قلمداد شوند. بخش " purchases / user 3 . txt > " از اولین آرگومان تابع ( ) popen که به تکنیک redirect شهرت دارد، باعث می‌شود تا خروجی حاصل از اجرای برنامه column به‌جای نمایش در خروجی استاندارد (که اغلب صفحه نمایش است) در فایلی با نام user3 . txt در مسیر تعیین شده واقع شود. توجه به این نکته ضروری است که وجود فهرستی با عنوان purchases در سیستم فایل کامپیوتر ضروری بوده و علاوه بر این برنامه باید مجوز نوشتن آن‌را داشته باشد (اگر بخواهیم دقیق‌تر باشیم باید چنین اظهار کنیم که کاربری که این برنامه را اجرا می‌کند، باید مجوز نوشتن در فهرست purchases را داشته باشد).

همچنین توجه کنید که با اجرای این فرمان چیزی بیش از یک عمل ساده را انجام می‌دهیم، بدین ترتیب که علاوه بر اجرای فرمان column خروجی حاصل از آن‌را نیز در یک فایل می‌نویسیم. درحقیقت کاری که ما انجام می‌دهیم این است که فرمان موردنظر را در یک shell غیر محاوره‌ای صادر می‌کنیم و این خود بدان معنی است که علاوه بر تغذیه فرآیند موردنظر با محتویات مطلوب می‌توانیم سایر فرآیندها را نیز به‌طور توام مورد استفاده قرار دهیم. حتی می‌توانیم خروجی حاصل از اجرای فرمان column را به‌صورتی که در زیر مشاهده می‌کنید از طریق پست الکترونیکی به فرد مورد نظرمان ارسال نماییم:

```
Popen (" cloumn - tc 3 - s / | mail matt@corrosive . co . uk ", " w ") ;
```

(با استفاده از تکنیکی موسوم به خط لوله یا piping که با بهره‌گیری از کاراکتر | قابل پیاده‌سازی است در سیستم عاملهای نوع UNIX می‌توانیم خروجی حاصل از یک برنامه را به‌عنوان ورودی برنامه دیگر مورد استفاده قرار دهیم).

این درجه عالی از قابلیت انعطاف البته می‌تواند عامل بسیار تهدیدکننده‌ای نیز برای سیستمی که یک چنین فرآیندی را اجرا می‌کند، تلقی گردد و آن درصورتی است که ورودی تعیین شده توسط کاربر را به یک تابع PHP که فرمانی از shell را صادر می‌کند، ارسال نماییم. به‌عبارت دیگر راهی را جهت اجرای فرمانهای shell بر روی کامپیوتر سرور در اختیار کاربران قرار دهیم. به‌زودی در همین ساعت جوانبی از احتیاط را که رعایت آنها در این مورد ضروری است، بررسی خواهیم کرد.

در ادامه برنامه لیست ۲- ۲۱ با در دست داشتن مرجع ph$ در خط ۱۶ با استفاده از ساختار تکرار foreach بر روی آرایه products$ برنامه را پی می‌گیریم. هر یک از عناصر این آرایه خود آرایه دیگری است که ما با بهره‌گیری از تابع ( ) join در خط ۱۷ برنامه آن‌را به یک دنباله کاراکتری تبدیل کرده‌ایم. توجه کنید که به‌جای بهره‌گیری از فضای خالی به‌عنوان بخشی از یک آرگومان سطر فرمان یعنی / s - مشخص شده است، استفاده کرده‌ایم. این کاراکتر چنانکه مشاهده می‌کنید کاراکتر ساده /

است. استفاده از کاراکتری مذکور جهت انجام این کار امری ضروری است زیرا استفاده از فضای خالی جهت جداسازی عناصر آرایه مورد بحث به‌طور حتم موجب سردرگمی برنامه column خواهد شد. با ترکیب عناصر این آرایه در قالب دنباله کاراکتری و جداسازی آنها با کاراکتر مناسب / اکنون می‌توان آن‌را به‌همراه یک علامت خط جدید به تابع ( ) fputs ارسال نمود (خط ۱۷ را ببینید).

گام نهایی بستن ارتباط مابین برنامه اسکریپت ما با برنامه column است. این عمل در خط ۱۸ و با ارسال مرجع فرآیند $ph به‌عنوان آرگومان به تابع ( ) pclose انجام می‌شود.

اکنون با اجرای برنامه موجود در لیست ۲- ۲۱ خروجی حاصل از برنامه column در فایلی با عنوان user3 . txt از فهرست purchases قرار می‌گیرد. نمونه‌ای از این خروجی که در جدولی به‌صورت زیر قالب‌بندی شده است، ارائه می‌گردد:

HAL 2000	2 blue
Tricorder	3 red
ORAC AI	1 pink
Sonic Screwdriver	1 orange

توجه کنید که با بهره‌گیری از تابع ( ) sprintf جهت قالب‌بندی متن می‌توانستیم به قابلیت حمل بیشتری در مورد برنامه دست پیدا کنیم که البته این خود به رویکردی که مورد نظرتان است، بستگی دارد.

اجرای فرمانها با بهره‌گیری از تابع ( ) exec

تابع ( ) exec یکی از توابع متعددی است که به ما امکان می‌دهد تا فرمانهای موردنظرمان را جهت اجرا به shell ارسال نماییم (shell در سیستمهای UNIX محلی برای اجرای فرمانها و برنامه‌ها است و پنجره فرمان در سیستم عامل windows را می‌توان معادلی برای shell در سیستم عامل UNIX فرض کرد. هرچند که قابلیتهای shell بسیار بیشتر از معادل خود در سیستم عامل windows است). این تنها جهت اجرای عملیات پیش‌بینی شده خود نیازمند دریافت یک آرگومان ورودی است. مسیر فرمان موردنظر در سیستم فایل محیط عامل تنها آرگومان ضروری این تابع را تشکیل می‌دهد. با این همه می‌توان درصورت لزوم از دو آرگومان اختیاری نیز جهت توسعه قابلیتهای تابع مورد بحث استفاده نمود. اولین آرگومان اختیاری این تابع آرگومانی است که مقادیر عناصر آن با خروجی حاصل از فرمانی که توسط آرگومان ضروری مشخص گردیده تعیین می‌شود. دومین آرگومان اختیاری نیز یک متغیر اسکالر است که مقدار آن با توجه به مقدار بازگشتی از فرمان موردنظر مشخص می‌شود.

برای روشن شدن مطلب در اینجا برنامه‌ای را ارائه می‌کنیم که از این تابع جهت اجرای فرمان موردنظر استفاده می‌کند. در این برنامه که کد آن در لیست ۳- ۲۱ آمده است قصد ما این است که اسامی فایل‌ها و زیرفهرستهای موجود در فهرست جاری را به‌همراه مشخصات مربوطه نمایش دهیم

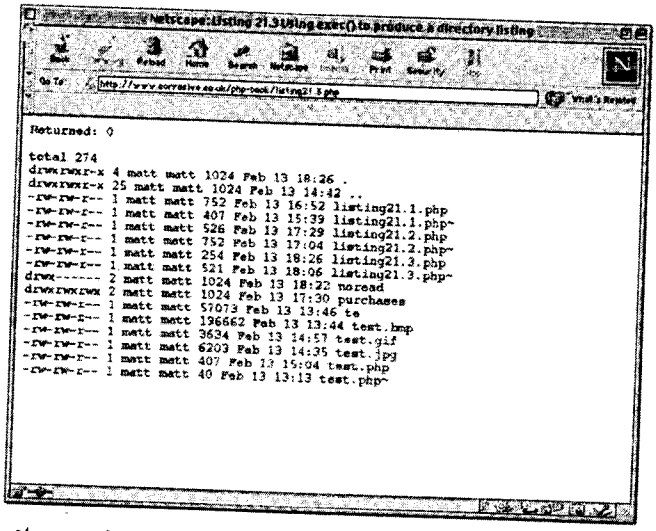
(منظور از فهرست جاری فهرستی است که برنامه لیست ۳- ۲۱ در آنجا واقع است). برنامه‌ای که در سیستم عامل UNIX اسامی فایل‌ها و زیرفهرست را در اختیار قرار می‌دهد برنامه‌ای با عنوان Ls است (این برنامه مشابه برنامه Dir در سیستم عامل windows است). از این‌رو جهت حصول نتیجه موردنظر ما دنباله کاراکتری " Ls -al " را به‌عنوان اولین آرگومان تابع ( ) exec مورد استفاده قرار می‌دهیم (خط ۷ را ببینید) و در نهایت خروجی حاصل از اجرای این برنامه را بر روی پنجره مرورگر اینترنت نمایش می‌دهیم. (به آرگومان‌های فرمان ls یعنی a- و l- توجه کنید. نکته اول اینکه در سیستم عامل UNIX می‌توان آرگومان‌ها را ترکیب کرده و آنها را به صورت al - نوشت. نکته بعدی درمورد تأثیر آرگومان‌ها است. فرمان Ls در حالت عادی فایل‌ها و زیرفهرستهای مخفی و سیستمی یک فهرست را نمایش نمی‌دهد و بهره‌گیری از آرگومان a - موجب نمایش این‌گونه فایل‌ها و فهرستها می‌شود. همچنین فرمان ls در حالت معمولی تنها اسامی فایل‌ها و فهرستها را نمایش می‌دهد. استفاده از آرگومان l - باعث می‌شود تا مشخصات دقیق‌تری نیز درمورد آنها در خروجی ظاهر شوند).

```
1: <html>
2: <head>
3: <title>Listing 21.3 Using exec() to produce a directory listing</title>
4: </head>
5: <body>
6: <?php
7: exec( "ls -al .", $output, $return );
8: print "<p>Returned: $return</p>";
9: foreach ( $output as $file )
10:     print "$file<br>";
11: ?>
12: </table>
13: </body>
14: </html>
```

لیست ۳- ۲۱ بهره‌گیری از تابع ( ) exec جهت اجرای برنامه Ls

خروجی حاصل از اجرای این برنامه را می‌توانید در شکل ۲- ۲۱ ملاحظه نمایید.

توجه کنید که موفقیت در اجرای برنامه Ls موجب می‌شود تا این برنامه مقدار عددی صفر را به برنامه فراخواننده بازگرداند. در صورتی‌که برنامه نامبرده قادر به بازخوانی فهرست موردنظر نبوده (برای مثال مجوز لازم در دسترس نباشد) و یا قادر به یافتن فهرست مذکور در مسیر مشخص شده نباشد مقدار عددی 1 را بازخواهد گرداند.



شکل ۲- ۲۱ بهره‌گیری از تابع ( ) exec جهت نمایش محتوای یک فهرست از سیستم فایل

باردیگر، چنان‌چه به‌دقت توجه کرده باشید، متوجه خواهید شد که ما برای حصول نتیجه موردنظرمان در این برنامه مشمول یک اصطلاح قدیمی شده‌ایم که به " اختراع مجدد" شهرت دارد (اشاره این اصطلاح به دوباره کاری است). برای انجام کار موردنظر ما می‌توانستیم به‌راحتی از توابع ( ) opendir و ( ) readdir که در درسهای گذشته بررسی کردیم، استفاده نماییم. با این حال می‌دانیم مواقعی وجود دارد که بهره‌گیری از یک فرمان آماده به اجرا نسبت به پیاده‌سازی همان قابلیت و عملکرد با بهره‌گیری از امکانات برنامه نویسی در PHP مستلزم صرف زمان کمتری است. این حقیقتی است که با استفاده از این امکانات برنامه نویسی در زبان PHP می‌توانید عملکرد بسیاری از فرمانهای سیستم عامل را حتی در مواقعی که کار پیچیده‌ای مورد نظر است، پیاده‌سازی کنید اما درصورتی که سرعت توسعه ضریب بسیار مهم و حیاتی برای شما محسوب می‌شود به‌احتمال زیاد تصمیم‌گیری در مورد بهره‌گرفتن از برنامه‌های خارجی به‌جای پیاده‌سازی مجدد آنها توسط کد PHP ارزش بسیاری خواهد داشت. با این حال به‌خاطرداشتن این موضع خالی از لطف نیست که فراخوانی و استفاده از فرآیندهای خارجی همواره هم از لحاظ زمانی و هم از لحاظ استفاده از منبع ارزشمند حافظه هزینه سرباری را به برنامه‌های شما تحمیل خواهد کرد.

اجرای فرمانهای خارجی با استفاده از تابع ( ) system یا عملگر Backtick

تابع ( ) system از این جهت که قادر به اجرای برنامه‌های خارجی است شباهت زیادی به تابع ( ) exec دارد. تابع مذکور نیز جهت اجرای عملیات پیش‌بینی شده نیاز به یک آرگومان ضروری دارد.

این آرگومان مشابه آنچه که در مورد تابع ( ) exec ملاحظه کردید مسیر فرمان موردنظر را جهت اجرا مشخص می‌کند. تابع ( ) system آرگومانی را نیز به‌عنوان یک آرگومان اختیاری می‌پذیرد. این آرگومان متغیری‌است که مقدار آن با توجه به مقدارباز‌گشتی از فرمان موردنظر (اولین آرگومان تابع( )system) تعیین می‌شود. تابع ( ) system خروجی حاصل از اجرای فرمان shell را مستقیماً بر روی مرورگر اینترنت ارسال می‌کند. قطعه برنامه‌ای که در ادامه مشاهده می‌کنید مستندات (توضیحات مربوط به چگونگی استفاده) فرمان man را بر روی پنجره مرورگر نشان می‌دهد:

```
< ? php
print "< pre >" ;
system ( "man man | col - b " , $return) ;
print "< / pre >" ;
? >
```

استفاده از نشانه < pre > در این قطعه کد چنانکه می‌دانید موجب حفظ قالب‌بندی صفحه می‌شود. از فراخوانی تابع ( ) system در اینجا برای اجرای فرمان man استفاده شده است (استفاده از فرمان man به‌همراه نام فرمان موردنظر در سیستم عامل UNIX موجب نمایش مستندات آن فرمان خواهد شد. برای مثال صدور فرمان man به‌همراه فرمان ls به‌صورت man ls موجب می‌شود تا سیستم مستندات مربوط به فرمان ls را که شامل چگونگی استفاده از آنست، بر روی صفحه نمایش دهد). چنانکه مشاهده می‌کنید خروجی حاصل از فرمان man با بهره‌گیری از تکنیک piping به برنامه دیگری با نام col ارسال شده است. برنامه col قادر است تا خروجی حاصل از فرمان man را به‌گونه‌ای که بتوان آن‌را در قالب ASCII مشاهده نمود مجدداً قالب‌بندی می‌کند. نتیجه بازگشتی از اجرای فرمان shell در متغیری که با نام $return به عنوان دومین آرگومان تابع ( ) system مشخص شده است ذخیره شده و تابع نامبرده خروجی خود را بر روی مرورگر نمایش می‌دهد.

به‌جای استفاده از تابع ( ) system درصورت تمایل می‌توان نتیجه مشابهی را با بهره‌گیری از عملگر backtick که با علامت (`) بر روی صفحه کلید مشخص شده است، به دست آورد (کلید مربوطه بر روی صفحه کلیدهای استاندارد درست در زیرکلید Esc واقع شده است). این روش مستلزم آن‌است که فرمان shell موردنظرتان را در درون یک جفت علامت(`) قرار دهید. بدین ترتیب فرمان واقع شده در بین جفت علامت فوق اجرا شده و خروجی حاصل از اجرای آن به‌عنوان نتیجه عملیات به برنامه مادر بازگردانده می‌شود. در این حالت می‌توان خروجی فوق را با بهره‌گیری از تابع ( ) print بر روی صفحه نمایش چاپ کرده و یا آن‌را در قالب یک متغیر برای استفاده‌های بعدی ذخیره نمود.

برای روشن شدن مطلب در اینجا قطعه برنامه اخیر خود را با استفاده از همین روش مجدداً بازنویسی می‌کنیم:

```
print "< pre >" ;
print ` man man | col - b ` ;
print "< / pre >" ;
```

دقت کنید که نتیجه حاصل از عملیاتی که به‌واسطه استفاده از عملگر backtick اجرا می‌شود به‌طور خودکار به‌خروجی ارسال نشده و برای انجام این‌کار لازم است تا خودمان به‌طور صریح از یک روش (مثلاً تابع ( ) print) جهت نمایش آن استفاده کنیم.

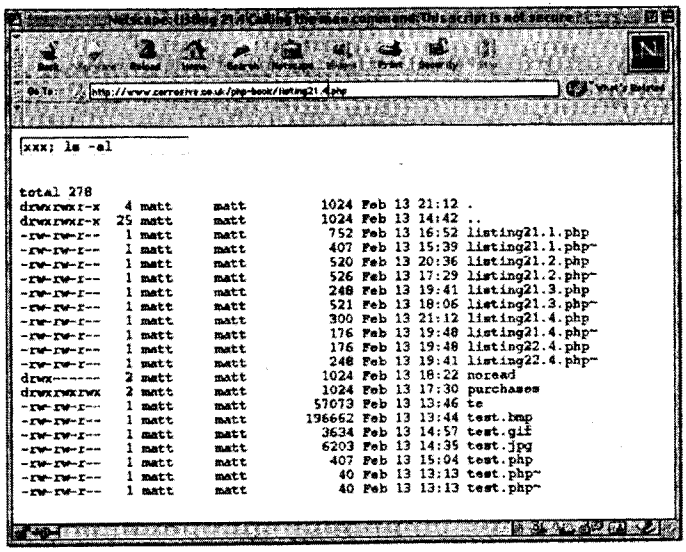
پوشش شکافهای امنیتی با بهره‌گیری از تابع escapeshellcmd( )

پیش از پرداختن به تابع ( ) escapeshellcmd و بررسی نحوه عملکرد آن ابتدا اجازه دهید خطری را که این تابع می‌تواند دفع کند، تشریح کنیم. فرض کنید مایلیم تا به‌شیوه‌ای امکان تایپ فرمان man را به‌همراه دستور موردنظر کاربر (جهت مشاهده مستندات مربوط به آن دستور) در اختیار وی قرار دهیم. اکنون که می‌توانیم یک صفحه مستندات را به‌عنوان خروجی ایجاد کنیم، ارسال هر صفحه دیگری به خروجی نیز فرآیند امکان‌پذیری به‌نظر می‌رسد. مراقب برنامه موجود در لیست ۴- ۲۱ باشید چراکه ما به‌طور یک اشکال امنیتی در آن کار گذاشته‌ایم.

```
 1: <html>
 2: <head>
 3: <title>Listing 21.4 Calling the man command.
 4:        This script is not secure</title>
 5: </head>
 6: <body>
 7: <form>
 8: <input type="text" value="<?php print  $manpage; ?>" name="manpage">
 9: </form>
10: <pre>
11: <?php
12: if ( isset($manpage) )
13:     system(  "man $manpage | col -b" );
14: ?>
15: </pre>
16: </table>
17: </body>
18: </html>
```

لیست ۴- ۲۱ فراخوانی فرمان man

چنانکه ملاحظه می‌کنید ما در این برنامه مثال قبلی خود درمورد فراخوانی فرمان man را اندکی با اضافه کردن یک فیلد متن در خط ۸ و بهره‌گیری از مقدار حاصل از ارسال فرم در فرمان shell که در خط ۱۳ برنامه در قالب فراخوانی تابع ( ) system انجام شده است، تغییر داده‌ایم. با این همه ما به این برنامه اطمینان می‌کنیم. بر روی سیستمی از نوع UNIX یک کاربر بداندیش به راحتی قادر خواهد بود تا با اضافه کردن فرمانهای موردنظر خود به فیلد manpage از دسترسی محدود خود به سرور لذت کافی ببرد. شکل ۳- ۲۱ نتیجه یک همچنین دستیابی را نشان می‌دهد.



شکل ۳- ۲۱ فراخوانی فرمان man ـ خطر امنیتی

همان گونه که در این شکل مشاهده می‌کنید کاربر بداندیش ما در اینجا فرمان xxx; ls -al را از طریق فیلدی که به همین منظور پیش‌بینی شده است برای اجرا به سرور تحویل می‌دهد. برنامه چنان‌که در لیست ۴- ۲۱ ملاحظه شد این فرمان را در قالب متغیری با نام $manpage ثبت می‌کند و پس از ترکیب این متن با فرمان shell آن‌را به‌عنوان آرگومان اول به تابع ( ) system ارسال می‌نماید. در این حالت اولین آرگومان تابع نامبرده به‌صورت زیر خواهد بود:

" man xxx; ls -al | col - b "

کاملاً واضح است که این فرمان به سیستم دستور می‌دهد تا مستندات مربوط به فرمانی با عنوان xxx را که البته فرمان معتبری در سیستم عامل UNIX محسوب نمی‌شود، مورد بازیابی قرار دهد. در ادامه، پس از بازیابی مستندات این فرمان اقدامی را جهت بازیابی اسامی مشخصات فهرستها و فایل‌های موجود در فهرست جاری صورت داده و با بهره‌گیری از فرمان col خروجی به‌صورت قالب بندی‌شده‌ای نمایش‌می‌یابد.در مورد فاجعه آمیز بودن این وضعیت و اینکه چه خطرات امنیتی می‌تواند در پی داشته باشد، خوب فکر کنید. یک بازدیدکننده بی‌مسئولیت بدین ترتیب به‌راحتی می‌تواند به مشخصه‌های کلیه زیرفهرستها و فایل‌های موجود در فهرست جاری دست پیدا کند. وضعیت حتی بدتر از این است؛ چراکه کاربر موردنظر با کمی هوشیاری می‌تواند فایل etc / passwd / را با بهره‌گیری از دنباله کاراکتری زیر مورد بازخوانی قرار دهد:

xxx ; cat / etc / passwd

خوشبختانه سیستم امنیتی UNIX به‌گونه‌ای است که لیست کلمات عبور کاربران سیستم در فایل etc / shadow / نگهداری می‌شود و تنها توسط کاربر اصلی سیستم که با عنوان ریشه یا root

شناخته می‌شود، قابل بازخوانی و دسترسی است. با این همه وضعیت فوق سوراخ امنیتی بسیار هولناکی را برای ما آشکار می‌کند. واضح است که اعطای مجوز انجام چنین کارهایی به کاربران هیچ‌گونه معنی دیگری غیر از دعوت آنها به تاخت و تاز در درون سیستم در پی ندارد. مطمئن‌ترین روش مقابله با یک چنین وضعیت خطرناکی این است که سیستم را به واسطه جلوگیری از ارسال ورودی مستقیم به فرمانهای shell و در کل خود shell محافظت کنیم. یکی از مکانیزم‌های مفید و قابل اعتماد در این راه استفاده از تابع ( ) escapeshellcmd است، بدین ترتیب که به‌منظور محافظت از سیستم می‌توانیم از تابع ( ) escapeshellcmd جهت اضافه کردن علامت \ به هرگونه کاراکترهای اضافی است که کاربر به‌منظور انجام عملیات خود تنها از یک آرگومان ورودی استفاده می‌کند. این آرگومان ورودی همان دنباله کاراکتری موردنظر ما است که کاربر جهت اجرا به برنامه ما ارسال می‌کند. آنچه که تابع مورد بحث به‌عنوان نتیجه عملیات به برنامه فراخواننده باز می‌گرداند، نسخه دست‌کاری شده‌ای از دنباله‌های کاراکتری ورودی است (البته این دستکاری به‌گونه‌ای است که امکان شکستن حریم امنیتی را توسط کاربران بداندیش از میان بر می‌دارد). برنامه موجود در لیست ۵- ۲۱ با بهره‌گیری از تابع ( ) escapeshellcmd سعی دارد تا از امنیت سیستم دفاع کند.

```
 1: <html>
 2: <head>
 3: <title>Listing 21.5 Escaping user input with
 4:        the escapeshellcmd() function</title>
 5: </head>
 6: <body>
 7: <form>
 8: <input type="text" value="<?php print  $manpage; ?>" name="manpage">
 9: </form>
10: <pre>
11: <?php
12: if ( isset($manpage) ) {
13:     $manpage = escapeshellcmd( $manpage );
14:     system(  "man $manpage | col -b" );
15: }
16: ?>
17: </pre>
18: </table>
19: </body>
20: </html>
```

لیست ۵- ۲۱ جلوگیری از ارسال فرمان کاربر به shell با استفاده از تابع ( ) escapeshellcmd

تنها تفاوتی که این برنامه ارائه شده در لیست قبل دارد، این است که در خط ۱۳ از تابع ( ) escapeshellcmd استفاده شده است. در این حالت چنانکه کاربر موردنظر ما دنباله کاراکتری "xxx; cat / etc / passwd" را در فیلدی که در اختیار دارد وارد نموده و آن‌را جهت اجرا به این برنامه ارسال نماید، تابع ( ) escapeshellcmd با تغییر آن به‌صورت "xxx \ ; cat / etc / passwd" مانع از اجرای آن توسط shell خواهد شد. درحقیقت آنچه که کاربر مذکور در خروجی مشاهده می‌کند محتوای فایل

شامل کلمات عبور کاربران سیستم نبوده بلکه به‌سادگی مستندات فرمان cat را به‌عنوان نتیجه ارسال فرمان ناشایست خود بر روی صفحه مشاهده خواهد کرد.

در اینجا نکته‌ای موجود است که تذکر آن می‌تواند جهت تأمین امنیت سرور مفید واقع شود. باوجودی که ظاهراً بهره‌گیری از تابع ( ) escapeshellcmd می‌تواند میزان امنیت را بهبود بخشد اما در مجموع بهتر آن است که با شیوه‌ای سفت و سخت‌تر از ارسال فرمان کاربر به shell سیستم جهت اجرا جلوگیری به‌عمل آوریم. یک روش بسیار مطمئن‌تر برای انجام این کار این است که لیستی از تمامی مستندات فرمانهای مجاز را از پیش بر روی سیستم خود نگه داریم. بدین ترتیب می‌توانیم آنچه را که کاربر به‌عنوان ورودی موردنظر خود به این برنامه ارسال می‌کند با اقلام موجود در این لیست مقایسه کرده و برمبنای نتیجه این مقایسه اقدام به فراخوانی تابع ( ) system نماییم. یک چنین مکانیزمی را در قسمت بعدی بررسی خواهیم نمود.

اجرای برنامه‌های خارجی با استفاده از فراخوانی تابع ( )passthru

تابع ( ) passthru نیز به‌مانند تابع ( ) system جهت اجرای برنامه‌های موردنظر بر روی shell سیستم مورد استفاده قرار می‌گیرد. با این حال تفاوتی که این تابع با تابع ( ) system دارد این است که خروجی حاصل از اجرای فرمان بر روی shell پیش از نمایش در خروجی بافر نمی‌شود. یک چنین وضعیتی باعث می‌شود که این تابع بیشتر در مورد فرمانهایی مورد بهره‌برداری قرار بگیرد که خروجی آنها به‌جای داده‌های متنی ساده داده‌های باینری است (مانند یک تصویر گرافیکی)، ساختار فراخوانی این تابع نیز مانند تابع مشابه خود ساده است به‌گونه‌ای که تنها از یک آرگومان ضروری برای انجام عملیات پیش‌بینی خود استفاده می‌کند. این آرگومان ضروری همان فرمان موردنظر است که تابع ( ) passthru باید اجرا کند. تابع مورد بحث ضمناً آرگومانی را به‌عنوان آرگومان اختیاری دریافت می‌کند. این آرگومان اختیاری متغیری است که مقدار بازگشتی حاصل از اجرای فرمان موردنظر (آرگومان اول) را ذخیره می‌کند.

جهت روشن شدن مطلب اجازه دهید تا در این قسمت برنامه کوتاهی را ارائه کنیم که از این تابع جهت اجرای فرمان در shell سیستم استفاده می‌کند. فرض کنید قصد ما این است که برنامه‌ای را به زبان PHP جهت ارسال تصاویر به‌خروجی توسعه دهیم. به این ترتیب می‌توانیم برنامه موردنظر را از طریق یک صفحه HTML یا یک سند PHP دیگر مورد فراخوانی و استفاده قرار دهیم. به‌عبارت دیگر می‌توانیم بخش بزرگی از کار موردنظر را به یک برنامه کاربردی خارجی سپرده و کد برنامه اصلی خود را در حد معقول کوچک نگه داریم. ضمن اینکه می‌توانیم از این برنامه کاربردی بارها و بارها در سایر کاربردها نیز بهره‌برداری کنیم. برنامه موجود در لیست ۶- ۲۱ کد مورد نیاز برای تعیین موقعیت تصویر بر روی سیستم فایل و ارسال آن به‌عنوان خروجی به صفحه مرورگر اینترنت را نشان می‌دهد.

```
1: <?php
2: if ( isset($image) && file_exists( $image ) ) {
3:     header( "Content-type: image/gif" );
4:     passthru( "giftopnm $image | pnmscale -xscale .5 -yscale .5 | ppmtogif"
);
5: } else
6:     print "The image $image could not be found";
7: ?>
```

لیست ۶- ۲۱ استفاده از تابع ( ) passthru جهت تولید داده‌های باینری

همان‌گونه که مشاهده می‌کنید، در این برنامه از تابع( ) escapeshellcmd استفاده نکرده‌ایم. در حقیقت استراتژی امنیتی ما در این برنامه به‌گونه‌ای متفاوت تأمین می‌شود. چنانکه می‌بینید، در این برنامه با فراخوانی تابع دیگری با عنوان ( ) file _ exists در خط ۲ مقداری را که کاربر به‌عنوان ورودی مشخص کرده است با فایل‌های موجود در سیستم فایل مقایسه می‌کنیم. در صورتی که تصویر موردنظر کاربر در سیستم فایل موجود نباشد، از ارسال متغیر $image به shell و اجرای فرمان لازم جهت قالب‌بندی و نمایش تصویر خودداری می‌کنیم. توجه کنید که به‌منظور افزایش ضریب امنیت حتی می‌توانستیم فایل‌های ورودی را به پسوندهای خاصی مانند png . یا jpg . یا gif . و یا به هرچیز دیگری محدود کرده و یا دستیابی به فایل‌ها را تنها به فهرستهای مشخصی از سیستم فایل محدود کنیم.

چنانکه در این برنامه مشاهده می‌کنید در خط ۴، تابع ( ) passthru را فراخوانی کرده‌ایم. این تابع در فراخوانی فوق کار نسبتاً پیچیده‌ای را انجام می‌دهد. به‌گونه‌ای که فرمانی را جهت اجرای سه برنامه مختلف که با بهره‌گیری از تکنیک piping هریک خروجی دیگری را مصرف می‌کند، صادر کرده‌ایم. درصورتی‌که مایل به اجرای برنامه لیست ۶- ۲۱ بر روی سیستم خود هستید دقت کنید که جهت حصول نتیجه موردنظر لازم است تا ابتدا هر سه برنامه giftopnm، pnmscale و بالاخره ppmtogif را بر روی سیستم خود نصب کنید. همچنین توجه کنید که نام هر سه برنامه باید از طریق متغیر path سیستم در دسترس باشد به‌طوری که تایپ نام هر یک از این برنامه‌ها از هر موقعیتی در سیستم منجر به اجرای آن برنامه شود. اولین برنامه فراخوانی شده در این میان giftopnm است. همان‌گونه که مشاهده می‌کنید متغیر $image به‌عنوان آرگومان این برنامه مورد استفاده قرار گرفته است. این برنامه به‌سادگی یک تصویر گرافیکی از نوع gif را مورد بازیابی قرارداده و آن‌را به قالب دیگری که به قالب pnm شهرت دارد، تبدیل می‌کند. خروجی حاصل از این برنامه با استفاده از تکنیک piping به برنامه دیگری که عنوان آن pnmscale است، ارسال می‌گردد. برنامه اخیر ترتیبی می‌دهد تا تصویر حاصل از عملیات اندازه‌ای برابر با ۵۰ درصد ابعاد اصلی خود را پیدا کند (آرگومان 5 . xscale - موجب کوچک‌نمایی ۵۰ درصدی در راستای افق و آرگومان 5 . yscale - موجب کوچک‌نمایی ۵۰ درصدی در راستای عمود می‌شود. بدین ترتیب کل تصویر در مجموع شامل ۵۰ درصد کوچک‌نمایی می‌شود). درنهایت خروجی حاصل از برنامه pnmscale بار دیگر با بهره‌گیری از تکنیک piping به برنامه

دیگری با عنوان ppmtogif ارسال می‌شود. برنامه ppmtogif عکس عملکرد برنامه giftopnm را پیاده‌سازی کرده و تصویر را به قالب gif، یعنی همان قالب اولیه خود تبدیل می‌نماید و تصویر نهایی به‌عنوان خروجی بر روی مرورگر اینترنت به‌نمایش درمی‌آید.

اکنون می‌توانیم این برنامه اسکریپت را از درون هر صفحه یا سندی از وب مورد استفاده قرار دهیم:

```
< img src = "listing 21 . 6 . php? image = < ? php print urlencode
        ( " / path / to / image . gif ") ? > " >
```

فراخوانی یک برنامه CGI خارجی با استفاده از فراخوانی تابع virtual ( )

در اغلب مواقع هنگامی‌که وب سایتی را از قالب ساده HTML به قالب PHP تبدیل می‌کنید متوجه خواهید شد که SSI شامل در صفحات HTML کارایی خود را از دست داده و به صورت عادی کار نمی‌کند (SSI یا Server _ side Include به مجموعه‌ای از فرمانهای خاص اطلاق می‌شود که جهت تولید محتوای پویا در درون اسناد HTML تعبیه می‌شوند. برخی از این فرمانها عبارتند از: ECHO و INCLUDE که امکان بهره‌گیری از محتویاتی که دائماً در حال تغییر است، مانند زمان و اطلاعات ذخیره شده در بانک اطلاعاتی خاصی را در صفحات وب در اختیارمان قرار می‌دهند. همچنین فرمان EXEC که می‌توان از آن جهت اجرای برنامه‌های CGI و تعبیه نتایج حاصل در صفحات وب بهره گرفت). اگر PHP را به‌عنوان ماجولی از وب سرور Apache مورد استفاده قرار می‌دهید (نام این ماجول php _ mod است)، می‌توانید از تابع ( ) virtual برای فراخوانی برنامه‌های CGI، مانند شمارنده‌هایی که با زبان برنامه‌نویسی Perl یا C پیاده‌سازی شده‌اند، و منظور کردن خروجی حاصل از اجرای آن‌ها در صفحات وب استفاده نمایید. توجه کنید که هریک از برنامه‌های CGI مورد استفاده باید هدرهای HTTP را پیش از ارسال هرگونه خروجی به مرورگر اینترنت ارسال نماید.

اجازه دهید تا برای روشن شدن این مطلب مثال کوتاهی را که شامل یک برنامه CGI است، ارائه دهیم. این برنامه CGI با استفاده از زبان برنامه‌نویسی perl توسعه یافته است. درصورتی‌که با زبان برنامه‌نویسی perl آشنایی ندارید هیچ جای نگرانی نیست؛ چراکه درک این برنامه بیش از اندازه ساده است؛ به‌گونه‌ای که ابتدا هدر HTTP مورد نیاز و سپس کلیه متغیرهای سیستمی مربوطه را جهت نمایش به خروجی ارسال می کند:

```
# ! / usr / bin / perl  - w
print "Content -type : text /html  n \ n" ;
foreach (keys %ENV) {
   print "$ _ : $ENV { $ _ } < br > \ n" ;
}
```

با فرض اینکه این برنامه در قالب یک برنامه اجرایی با نام test . pl در درون فهرستی با عنوان cgi _ bin ذخیره شده باشد، می‌توانیم به این ترتیب با فراخوانی تابع ( ) virtual خروجی حاصل از آن‌را در سند PHP موردنظرمان درج کنیم. قطعه کد زیر چگونگی انجام این کار را نشان می‌دهد:

```
< ? php
virtual ( " / cgi -matt / test . pl " ) ;
? >
```

جمع‌بندی

در درس این ساعت چگونگی ارتباط با shell سیستم عامل و روشهای مختلف جهت به‌کارگیری برنامه‌های خارجی یا فرمانهای موجود در آن آشنا هستید. همان‌گونه که اکنون تصدیق می‌کنید، PHP زبان برنامه‌نویسی توانمندی است و این امکان وجود دارد که عملیات پیچیده مختلفی را با بهره‌گیری از امکاناتی که در این زبان تعبیه شده است، انجام دهد؛ اما در برخی موارد لازم است تا به‌دلیل افزایش در سرعت اجرای برنامه بخشی از عملیات موردنظرمان را به فرمانهایی از سیستم عامل (یا به عبارت دقیق‌تر، برنامه‌های موجود بر روی سیستم عامل که از این‌رو به آنها برنامه‌های خارجی می‌گوییم) واگذار کنیم.

بهره‌گیری از تکنیک piping (علامت) جهت انتقال داده‌ها از یک فرمان یا به یک فرمان با استفاده از تابع ( ) popen موضوع مهم دیگری بود که در درس این ساعت مورد بحث و بررسی قرار گرفت. این روش در مورد برنامه‌های کاربردی که داده‌هایی را از طریق ورودی استاندارد می‌پذیرند، مفید است همچنین روش مذکور در مواقعی مورد استفاده قرار می‌گیرد که خواسته باشیم داده‌هایی را که از یک برنامه خارجی وارد برنامه می‌شود، مورد پردازش قرار دهیم.

در این ساعت با چگونگی استفاده از تابع ( ) exec و ( ) system و همچنین نحوه بهره‌برداری از علامت backtick جهت ارسال فرمانهای مورد نظر به shell سیستم عامل (محل اجرای فرمانها) و دریافت ورودی برنامه از کاربر را به‌خوبی فراگرفتید. ملاحظه کردید که ارسال ورودی کاربر به shell سیستم‌عامل‌چه‌خطرات‌امنیتی‌می‌تواندد‌رپی داشته باشد. درهمین‌رابطه‌تابعی با نام ( ) escapeshellcmd را معرفی کردیم که امکان محدود کردن ورود داده‌ها توسط کاربران را به نحو مطلوبی محدود می‌کرد. علاوه بر این چگونگی استفاده از تابع ( ) passthru را جهت دریافت داده‌های باینری حاصل از اجرای فرمان موردنظر در shell سیستم فراگرفتید. درنهایت نحوه شبیه‌سازی فرمانهای SSI را با استفاده از تابع ( ) virtual مورد بحث و بررسی قرار دادیم.

در درس ساعت آینده قابلیتهایی از زبان برنامه‌نویسی PHP را که جهت پشتیبانی از XML در این زبان پیش‌بینی شده است، بررسی خواهیم کرد. طی این بررسی علاوه بر توابع استاندارد PHP که

از نسخه‌های پیشین در PHP4 باقی‌مانده، به بررسی توابع بسیار جدیدی خواهیم پرداخت که در زمان انتشار این کتاب به‌تازگی در دسترس برنامه‌نویسان قرار گرفته است.

پرسش و پاسخ

پرسش: در درس این ساعت مبحث مربوط به امنیت به‌طور تقریباً مفصل بررسی شد. چگونه می‌توان اطلاعات بیشتری در مورد امنیت سایت‌های وب به دست آورد؟

پاسخ: به احتمال قوی یکی از جامع‌ترین منابعی که مسأله امنیت وب را به‌طور مقدماتی در قالب پرسشهای متداول یا FAQ (Frequently Asked Questions) مورد بررسی قرار داده، سندی است که توسط آقای Lincoln Stein (کسی که ماجول معروف CGI . pm را برای زبان برنامه‌نویسی perl توسعه داده) تدوین شده است.

برای دستیابی به این سند کافی است تا آدرس زیر را با استفاده از مرورگر اینترنت خود مورد دستیابی قرار دهید:

http: // www. w3 . org / Security / Faq

پرسش: در چه مواقعی باید استفاده از برنامه‌های خارجی آماده (از پیش نوشته شده) را به ایجاد مجدد آنها جهت پیاده‌سازی عملکرد موردنظر خود در برنامه‌های اسکریپت ترجیح داد؟

پاسخ: مسأله‌سرعت در توسعه و کارآیی در دنیای برنامه‌نویسی اهمیت خاصی دارد. به‌طور کلی هرجا که این دو عامل به‌همراه قابلیت حمل برنامه مد نظر باشد، باید از برنامه‌های خارجی بهره‌برداری کرد.

درصورتی‌که خودتان عملکرد مورد نیاز خود را به‌جای بهره‌گیری از فرآیندهای خارجی موجود بر روی سیستم ایجاد می‌کنید، برنامه اسکریپت باید به‌آسانی بر روی محیطهای زیربنایی (سیستم عامل) مختلف و یا بر روی سیستمهایی که شامل برنامه‌های کاربردی که عملکردهای موردنیاز را تأمین می‌کنند نمی‌باشند، به‌خوبی اجرا شود. در مورد عملکردهای ساده (مانند نمایش اسامی فهرستها و زیرفهرستهای موجود) به احتمال قوی روش کد کردن عملکردهای موردنیاز در برنامه نسبت به بهره‌گیری از برنامه‌های خارجی مانند ls جهت حصول نتیجه مطلوب روش بهتر و مؤثرتری می‌باشد. این بدان دلیل است که استفاده از برنامه‌های خارجی در هر صورت سربار را به‌واسطه تعویض فرآیندها به‌هنگام فراخوانی به سیستم مربوطه تحمیل می‌کند.

تمرینها

هدف از این بخش ارائه تمرینهای مختلف در قالب آزمون است. پاسخ آزمون بلافاصله بعد از آن

آمده است. بخش فعالیتها شامل تمرینهایی است که جهت افزایش مهارت و قابلیت برنامه‌نویسی خواننده طراحی شده و فاقد پاسخ است.

آزمون

۱- از کدام تابع PHP می‌توان با بهره‌گیری از تکنیک piping با یک فرآیند خارجی ارتباط برقرار کرد؟

۲- با استفاده از کدام تابع داده‌ها را از یک فرآیندی که برنامه در حال برقراری ارتباط با آن‌است بازخوانی نمود؟

۳- با استفاده از کدام تابع می‌توان داده‌ها را در یک فرآیندی که برنامه در حال برقراری ارتباط با آن‌است، نوشت؟

۴- آیا با بهره‌گیری از تابع ( ) exec می‌توان خروجی حاصل از اجرای یک فرمان در shell سیستم عامل را مستقیماً جهت نمایش به مرورگر اینترنت ارسال کرد؟

۵- تابع ( ) system با داده‌های خروجی حاصل از اجرای یک فرمان خارجی که آن‌را اجرا کرده است، چه می‌کند؟

۶- عملگر backtick چه چیزی را به برنامه‌ای که از آن استفاده کرده است، باز می‌گرداند؟

۷- جهت تأمین امنیت سیستم چه اقدامی را می‌توان جهت جلوگیری از فرمانهایی که کاربر جهت اجرا به shell سیستم عامل ارسال می‌کند، انجام داد؟

۸- چگونه می‌توان یک برنامه CGI خارجی را از درون یک برنامه PHP اجرا نمود؟

پاسخ آزمون

۱- با استفاده از تابع ( ) popen می‌توان از درون یک برنامه PHP ارتباطی را با یک فرآیند خارجی برقرار کرد.

۲- عمل خواندن از فرآیندی که برنامه PHP موجود از طریق فراخوانی تابع ( ) popen با آن ارتباط برقرار کرده است، مشابه خواندن از یک فایل است که در درسهای قبل مورد بررسی قرار دادیم. بنابراین با بهره‌گیری از توابعی مانند ( ) feof و ( ) fgets می‌توان این‌گونه فرآیندها را مورد بازخوانی قرار داد.

۳- عمل نوشتن در یک فرآیند خارجی نیز مانند عمل خواندن از آن، مشابه عملیات نوشتن در فایل است. بنابراین برای انجام این کار می‌توان از تابع ( ) fputs استفاده کرد.

۴- تابع ( ) exec آرایه‌ای را در قالب یک متغیر به‌عنوان آرگومان ورودی دریافت می‌کند. این تابع آرایه مذکور را با خروجی حاصل از اجرای فرمان موردنظر در shell سیستم عامل مقداردهی می‌نماید.خروجی‌حاصل‌از تابع مذکور مستقیماً بر روی پنجره مرورگر اینترنت ارسال نمی‌شود.

۵- تابع ( ) system خروجی حاصل از اجرای فرمان خارجی مربوطه را مستقیماً بر روی پنجره مرورگر اینترنت ارسال می‌کند.

۶- عملگر backtick خروجی حاصل از اجرای فرمان خارجی موردنظر را به برنامه‌ای که آن‌را مورد استفاده قرار داده است، باز می‌گرداند. این خروجی را بسته به نیاز می‌توان ذخیره کرده یا مورد پردازش قرار داد و یا به‌عنوان نتیجه عملیات بر روی صفحه نمایش داد.

۷- با بهره‌گیری از تابع ( ) escapeshellcmd می‌توان از ورودی که کاربر جهت اجرا جهت اجرای فرمانهای shell به برنامه ارسال کرده است، صرف‌نظر نمود. با این وجود، مطمئن‌ترین روش جهت اجرای فرمانهای shell از طریق برنامه این است که ترتیبی دهیم تا کاربر در مجموع قادر به ارسال فرمانهای خود به برنامه و بدین ترتیب اجرای آنها در shell سیستم نباشد.

۸- با استفاده از تابع ( ) virtual می‌توان یک برنامه CGI خارجی را در برنامه PHP فراخوانی کرد.

فعالیتها

۱- برنامه اسکریپتی ایجاد کنید که امکان اجرای فرمان ps از سیستم عامل UNIX را در اختیار قرار دهد. فرمان ps در این سیستم عامل موجب نمایش اطلاعات مربوط به کلیه فرآیندهای درحال اجرا بر روی کامپیوتر می‌شود. خروجی این برنامه باید بر روی پنجره مرورگر اینترنت به‌نمایش درآید. یک مثل معروف قدیمی چنین می‌گوید " دانش قدرت است" بنابراین از دراختیار گذاشتن چنین برنامه‌ای در دسترس کاربران خود جداً خودداری کنید.

۲- مستندات فرمان ps را با صدور فرمان man page در shell سیستم عامل خود مورد مطالعه قرار دهید. به برنامه ایجاد شده در تمرین قبل فرمی را اضافه کنید که کاربران از طریق آن بتوانند آرگومان‌های این فرمان را نیز مورد استفاده قرار داده و بدین ترتیب بتوانند خروجی حاصل از این فرمان را تغییر دهند. دقت داشته باشید که به هیچ وجه فرمانهای صادر شده از طریق کاربران را مستقیماً جهت اجرا به shell سیستم عامل تحویل ندهید.

ساعت بیست و دوم

PHP4 و XML

عدم توجه به تکنولوژی که این روزها موضوع صحبت هر محفل برنامه‌نویسی شده است، کاری بس دشوار می‌باشد. این تکنولوژی با عنوان XML که کوتاه شده Extensible Markup Language است موضوع بحث این ساعت را تشکیل می‌دهد. XML با سرعتی باورنکردنی در حال تبدیل شدن به یک ابزار بسیار با اهمیت برای به اشتراک گذاری داده‌ها مابین برنامه‌های کاربردی مختلف و جداسازی منطق برنامه‌های کاربردی از لایه نمایش آنها در پروژه‌های برنامه‌نویسی بزرگ است. از زمان انتشار اولین ویرایش کتاب حاضر زبان برنامه‌نویسی PHP به‌منظور پشتیبانی از XML و تکنولوژی‌های مربوطه دستخوش تغییراتی شده و بهبود یافته است. در حال حاضر ترکیب PHP و Zend با یکدیگر ابزار بسیار توانمندی جهت ایجاد برنامه‌های کاربردی مختلف خصوصاً در صحن e_business یا تجارت الکترونیکی محسوب می‌شود؛ ضمن اینکه از پشتیبانی XML نیز در این دو ابزار مهم نباید غافل شد. برای برنامه‌نویسان وب استفاده و درک مفاهیم XML دیگر یک گزینه نبوده بلکه در حال تبدیل شدن به یک ضرورت اجتناب‌ناپذیر است.

در این درس مطالب زیر را مورد بررسی قرار خواهیم داد:

- **مبانی XML و مفاهیم اولیه**
- **چگونگی پردازش اسناد XML با بهره‌گیری از توابع XML Parser**
- **چگونگی ایجاد اسناد XML با بهره‌گیری از توابع DOM**
- **چگونگی پیمایش یک ساختار داده‌ای XML**
- **چگونگی استفاده از یک سند XSL جهت تبدیل اسناد XML**

در ادامه به بررسی این موارد می‌پردازیم.

مفاهیم XML

XML که کوتاه شده عبارت Extensible Markup Language می‌باشد، ابزاری بسیار قابل انعطاف بوده و همین موضع باعث می‌شود که تعریف آن بسیار مشکل شود. پرداختن به این تکنولوژی و جزئیات مربوط به آن خارج از حوصله این کتاب بوده و خود کتاب دیگری را می‌طلبد. با این حال می‌توان مبانی این ابزار مهم در توسعه برنامه‌های کاربردی وب را به‌خوبی تولید کرد. درصورتی‌که به این ابزار و نحوه استفاده از قابلیتهای آن علاقه‌مندید، می‌توانید کتابی را که در همین رابطه انتشارات SAMS منتشر کرده است، مطالعه کنید. نام این کتاب " خودآموز استفاده از XML در ۲۴ ساعت" بوده و کد ISBN آن عبارت است از 7 - 32213 - 672 - 0 . با این حال جهت مشاهده یک تعریف رسمی و قابل استفاده در محافل علمی می‌توانید از محتویات وب سایت مربوط به آدرس زیر استفاده نمایید:

http: // www . w3 . org / XML

XML یک زبان نشانه‌گذاری است که به شما اجازه می‌دهد تا به ابداع زبانهای نشانه‌گذاری جدید مطابق نیازهایتان بپردازید. درحقیقت این تکنولوژی را بهتر است به‌جای دسته‌بندی به‌عنوان یک زبان به‌صورت مجموعه‌ای از قوانین فرض نماییم. این مجموعه قوانین روشهایی را که طی آن می‌توانید عناصر و نشانه‌های زبان موردنظرتان را تعریف کنید، مشخص می‌نماید (در حقیقت زبان نشانه‌گذاری HTML را نیز می‌توان در این قالب گنجاند). مادامی‌که این مجموعه قوانین را زیر پا نگذارید، این آزادی را دارید که زبان مورد نظرتان را به‌همراه قابلیتهای مطلوب ایجاد نمایید. به‌دلیل سفت و سخت بودن این قوانین پردازشگرهای XML مشکل چندانی در رابطه با تفسیر و بازخوانی اسناد XML نخواهند داشت. بدین ترتیب محتوای این اسناد را می‌توان به‌راحتی در دسترس برنامه‌های اسکریپتی که دستورالعملهای موجود در آنها را مورد پردازش قرار می‌دهند، گذاشت.

هر سندی از نوع XML معمولاً با معرفی زیر آغاز می‌شود:

< ? XML version = "1 . 0" ? >

همچنین ممکن است سند XML موردنظر از یک سند دیگر از نوع DTD یا Document Type Declarationبهره‌بگیرد.اسناد DTD و جزئیات مربوطه خارج از بحث این کتاب و درس حاضر می‌باشند. با این حال در مورد این اسناد همین کافی است که بگوییم اسناد DTD نشانه‌هایی را که یک سند XML می‌تواند مورد استفاده قرار دهد و همچنین ترتیب استفاده از آن نشانه‌ها را مشخص می‌کند (در واقع اسناد DTD نقش مانیفست اسناد XML مربوطه را ایفا می‌کنند). چگونگی مراجعه به یک DTD موجود به‌صورت زیر می‌باشد:

< ! DOCTYPE < rootel > SYSTEM "http : // www . corrosive . co . uk / sample . dtd " >

مابقی یک سند XML از نشانه‌های مختلف که بعضاً شامل صفات متعددی نیز می‌باشند، تشکیل می‌شود. نشانه‌های XML بسیار شبیه به نشانه‌های HTML هستند؛ هریک از نشانه‌های XML از یک بخش آغازین و یک بخش پایانی تشکیل می‌شود. این نشانه‌ها می‌توانند دربرگیرنده متن و یا سایر نشانه‌ها باشند.

نشانه آغازین همواره با علامت کوچک‌تر (<) آغاز شده و به‌دنباله آن نام نشانه مورد نظر قرار می‌گیرد و در نهایت با علامت بزرگ‌تر (>) بسته می‌شود. نشانه‌های آغازین می‌توانند شامل صفاتی چند نیز باشند. هریک از این صفتها از یک نام و مقدار متناظر با آن تشکیل شده که این مقدار باید در درون جفت علامت (" ") واقع شود و با استفاده از علامت نسبت‌دهی (=) به نام مذکور نسبت داده شود (دقت کنید که وجود جفت علامت " " برای تعیین مقدار صفت امری ضروری است و این خود یکی از مواردی است که اسناد XML را از اسناد HTML تفکیک می‌کند). قطعه سند زیر نمایشی از یک نشانه XML با نام newsitem می‌باشد که شامل صفتی با عنوان type است .

```
< newsitem type = " world " >
```

دقت کنید که هم نام صفت و هم نام نشانه باید مشمول قوانین نام‌گذاری خاصی باشند. طبق این قوانین نام‌گذاری هر دو باید با یک حرف یا علامت underscor آغاز شده و به‌دنبال آن از ترکیبی از حروف و اعداد (کاراکترهای الفبا عددی) استفاده شود. در مورد نام‌گذاری یک قانون سفت و سخت دیگر نیز موجود می‌باشد. طبق این قانون نام هیچ نشانه‌ای از سند XML نباید با دنباله کاراکتری "XML" آغاز شود.

نشانه پایانی نیز ترکیبی است از علامت کوچک‌تر (<) که به‌دنبال آن از علامت / و به‌دنبال آن نشانه موردنظر و درنهایت از علامت بزرگ‌تر به‌صورت زیر استفاده می‌شود:

```
< / newsitem >
```

همان‌گونه که مشاهده می‌کنید شکل ظاهری نشانه‌های XML (و بنابراین کل یک سند XML) کاملاً شکلی مأنوس و آشناست. تنها یک مورد وجود دارد که ممکن است کدنویسان HTML با آن آشنایی نداشته باشند و آن نشانه تهی یا empty element است. چنین نشانه‌هایی حاوی هیچ‌گونه محتوایی نیستند و در زبان نشانه‌گذاری XML از شکل خاصی برای نشان دادن این وضعیت استفاده می‌شود. شکل عمومی یک نشانه تهی به‌صورت زیر است:

```
< nothinghere > < / nothinghere >
```

با این حال در XML می‌توان آن‌را به این صورت نوشت:

```
< nothinghere / >
```

سند موجود در لیست ۱- ۲۲ یک سند ساده XML است. این سند یک نمونه اولیه است که در طول مثالهای این درس از آن بهره خواهیم گرفت.

```
1: <?xml version="1.0"?>
2: <banana-news>
3:         <newsitem type="world">
4:                 <headline>Banana sales reach all time high</headline>
5:                 <image>/res/high.gif</image>
6:                 <byline>William Curvey</byline>
7:                 <article>Research published today by the World Banana
8:                         Tribunal suggests that we have never had it so
9:                         good banana-wise...</article>
10:        </newsitem>
11:
12:        <newsitem type="home">
13:                <headline>Domestic banana use beggars belief</headline>
14:                <image>/res/use.gif</image>
15:                <byline>Charles Split</byline>
16:                <article>Bananas are for more than eating it seems. Local
17:                        Innovation Centers have been showcasing some
18:                        exciting banana related technologies...</article>
19:        </newsitem>
20: </banana-news>
```

لیست ۱- ۲۲ یک سند ساده XML

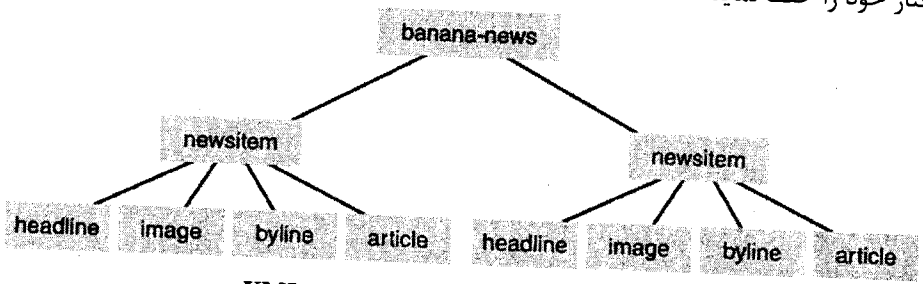
با وجودی که لیست ۱- ۲۲ شباهت زیادی به یک سند HTML دارد، می‌توان کاملاً به این نکته اعتراف کرد که در ساختار این سند از اسامی کاملاً ابتکاری استفاده شده است. این نکته ظریفی است که ضمناً مشخصه بارز اسناد XML نیز می‌باشد. این ویژگی کنترل وصف‌ناپذیری را در دسترس توسعه‌دهنده‌قرار می‌دهد اما از آنجا که قدرت زیاد، مسئولیتهای بزرگ را نیز به‌همراه دارد توسعه‌دهنده مسئول آن چیزی است که در سند درج خواهد کرد. برنامه‌ای موسوم به مفسر XML قواعد دستوری مورد استفاده در سند XML را مورد ارزیابی قرار داده و بدین ترتیب امکان دسترسی به نشانه‌های موجود در سند (و در نتیجه اطلاعات موجود در آن) را برای ما ساده می‌کند؛ اما توجه داشته باشید که توسعه کد موردنیاز جهت دستیابی به اطلاعات موجود به‌عهده توسعه‌دهنده است. قسمت زیادی از این درس به همین موضوع اختصاص یافته است.

چنانکه در لیست ۱- ۲۲ مشاهده می‌کنید سعی ما این است تا ساختاری را برای عناوین خبری یک روزنامه فرضی نمایش دهیم. کل این سند XML توسط نشانه < banana _ news > محصور شده است (از خط ۲ تا ۲۰). چنین نشانه‌ای در یک سند XML به نشانه ریشه یا root element شهرت دارد. یکی از قوانین سفت و سخت XML در رابطه با ساختار اسناد این است که هر سندی از نوع XML باید یک و تنها یک نشانه ریشه داشته باشد به‌گونه‌ای که این نشانه دربرگیرنده تمامی نشانه‌های دیگر سند باشد. قانون سفت و سخت دیگر این است که هر نشانه‌ای از سند XML (به غیر از نشانه ریشه) باید به‌طور کامل در درون نشانه دیگر واقع شود (بدین ترتیب یک رابطه پدر و فرزندی مابین تمامی نشانه‌های موجود در سند XML موجود بوده به‌گونه‌ای که نشانه ریشه پدر تمامی نشانه‌های دیگر می‌باشد). بنابراین همپوشانی نشانه‌هایی به‌صورت زیر:

< A >< B >< / A >< / B >

موجب شکایت پردازنده XML خواهد شد.

رابطه پدر و فرزندی موجود مابین نشانه‌های یک سند XML موجب می‌شود تا بتوانیم برای نمایش آن از یک ساختار درختی استفاده کنیم. ساختار درختی از آن جهت برای ما اهمیت دارد که درک آن برای ما بسیار ساده است. شکل ۱- ۲۲ ساختار درختی مربوط به سند موجود در لیست ۱- ۲۲ را نشان می‌دهد. چنانکه در این ساختار درختی به‌وضوح ملاحظه می‌کنید، نشانه _ banana < نشانه ریشه است که دو نشانه < newsitem > از آن مشتق شده‌اند. به این ترتیب بین این دو نشانه< newsitem > رابطه برادری برقرار است؛ چراکه هر دو از یک نشانه پدر مشتق شده‌اند. هر یک از این دو نشانه به نوبه خود شامل فرزندانی هستند. این درخت می‌تواند تا چندین سطح به همین ترتیب ساختار خود را حفظ نماید.



شکل ۱- ۲۲ ساختار درختی یک سند XML

با وجود مطالبی که تا بدین جای درس عنوان شد این پرسش منطقی به ذهن می‌رسد که XML در مجموع به چه کاری می‌آید؟ پاسخ کوتاه به این پرسش این است که به خود ما بستگی دارد. اما در حوزه عمل از اسناد XML جهت تامین اهداف خاصی استفاده می‌شود که در اینجا به سه نمونه از آنها اشاره می‌کنیم:

۱- سازماندهی‌منطقی‌داده‌ها جهت به اشتراک گذاشتن آنها (این هدف در لیست ۱- ۲۲ دنبال شده است).

۲- قالب‌بندی داده‌ها (HTML نمونه بارز یک چنین هدفی است).

۳- ارسال دستورالعملهای موردنظر به یک مفسر فرمان

دردرس این ساعت قصد ما از بررسی XML دنبال کردن هدف اول است. ساختار عناوین خبری که در لیست ۱- ۲۲ ملاحظه کردید امکان بسیار مناسبی را جهت کار با عناوین خبری هم در اختیار ما و هم در اختیار شرکای ما می‌گذارد. این نمونه بارزی از استفاده اشتراکی از داده‌های موجود است.

توابع مربوط به پردازش اسناد XML

در این قسمت از درس قصد داریم تا مطمئن‌ترین و قابل اعتمادترین ابزارهای موجود در زبان برنامه‌نویسی PHP را به‌منظور کاربر روی اسناد XML مورد بررسی قرار دهیم. توابعی که در این قسمت مورد بحث قرار می‌دهیم به ما این امکان را می‌دهند تا به‌سرعت و با کمترین برنامه‌نویسی اسناد XML را مورد دستیابی قرار دهیم.

توابع موردنظر ما دقیقاً توابعی هستند که آقای Jim Clark در کتابخانه‌ای با نام Expat (یا به‌عبارت‌دیگر Parser Toolkit XML)دراختیار توسعه‌دهندگان علاقه‌مند قرار داده است. جهت دستیابی به این کتابخانه ارزشمند نیازی نیست تا مبلغی را پرداخت نمایید، تنها کافی است تا سری به آدرس زیر بزنید:

http : // www . jclark . com / xml / expat . html

درصورتی‌که از وب سرور Apache 1 . 3 . 7 یا نسخه بالاتر آن استفاده می‌کنید، می‌توانید مطمئن باشید که در حال حاضر کتابخانه Expat بر روی وب سرور مورد استفاده‌تان واقع بوده و آماده بهره‌برداری است. به‌عبارت دیگر می‌توانید بدون اینکه از گزینه‌های نصب PHP استفاده نمایید به توابع موجود در این کتابخانه جهت کار بر روی اسناد XML موردنظرتان بهره‌برداری کنید. در غیر این‌صورت مجبورید تا جهت بهره‌مندی از توابع کتابخانه Expat گزینه مربوطه را هنگام نصب PHP بر روی کامپیوترتان مورد استفاده قراردهید. این گزینه به‌صورت زیر است:

- - with - xml

بدین ترتیب پیکربندی نصب به‌گونه‌ای خواهد بود که کتابخانه Expat در دسترس برنامه‌های اسکریپت‌قرار خواهد گرفت. جهت اطلاع بیشتر درمورد چگونگی نصب PHP به‌همراه گزینه‌های مختلف و همچنین نحوه پیکربندی آن به درس ساعت دوم با عنوان " نصب PHP بر روی کامپیوتر" مراجعه کنید.

دقت کنید مدلی که این پردازشگر بر مبنای آن عمل می‌کند مدلی است که به شیوه رخداد شهرت دارد. این بدان معنی است که به محض مواجهه برنامه با هریک از اجزای سند XML موردنظر، توابعی را که کاربر (برنامه‌نویس) جهت پردازش آنها از پیش آماده کرده است، فراخوانی می‌گردد.

دستیابی به منبع پردازشگر

به‌منظور پردازش یک سند XML پیش از انجام هر اقدامی لازم است تا به یک منبع پردازشگر دسترسی داشته باشیم. جهت دستیابی به یکی از این منابع می‌توانیم تابع ( ) xml_ parser_ create را مورد استفاده قرار دهیم. توجه کنید که تابع( ) xml_ parser_ create جهت انجام عملیات موردنظر به هیچ آرگومان ورودی نیاز ندارد. در مقابل، چنان‌چه همه‌چیز به‌طور دلخواه پیش برود این تابع یک

منبع پردازشگر را به برنامه فراخواننده بازمی‌گرداند. اما چنان‌چه فرآیند با مشکل مواجه شود تابع فوق مقدار false را بازخواهد گرداند. تابع ( ) xml_ parser_ create با این حال به هیچ آرگومانی نیاز ندارد اما درصورت تمایل برنامه‌نویس می‌تواند آرگومانی را از نوع دنباله کاراکتری به‌عنوان یک آرگومان اختیاری بپذیرد. این آرگومان اختیاری چگونگی کدگذاری کاراکترها را مشخص کرده و می‌تواند یکی از سه دنباله کاراکتری "US ASCII "، "ISO _ 8859 _ 1" و یا "UTF _ 8" باشد. مقدار پیش‌فرض این آرگومان اختیاری دنباله کاراکتری "ISO _ 8859 _ 1" است. بنابراین عبارت زیر که شامل فراخوانی تابع موردبحث است از این دنباله کاراکتری به‌عنوان آرگومان پیش‌فرض بهره می‌گیرد:

```
$parser = xml _ parser _ create ( )
```

پس از استفاده مجدد و بهره‌برداری از منبع پردازشگر زمانی که دیگر هیچ استفاده‌ای از آن نمی‌کنیم، بهتر آن‌است که حافظه اشغالی توسط آن‌را مجدداً به سیستم بازگردانیم. آزادسازی منابع کمک می‌کند تا برنامه‌هایی که در حال اجرا هستند با کمبود منابع که مهم‌ترین آنها پردازنده (CPU) و حافظه کامپیوتر هستند، مواجه نشوند. این کار به‌سادگی با فراخوانی تابع ( )xml_ parser_ free امکان‌پذیر است. تابع ( ) xml_ parser_ free جهت انجام عملیات موردنظر خود به یک منبع پردازنده معتبر به‌عنوان آرگومان ورودی نیاز دارد. تابع مذکور مقداری از نوع Boolean را به‌عنوان نتیجه عملیات به برنامه فراخواننده بازمی‌گرداند. درصورتی‌که عملیات آزادسازی حافظه با موفقیت همراه باشد این مقدار برابر با true و در غیر این‌صورت برابر با false خواهد بود عبارت زیر چگونگی استفاده از این تابع را نشان می‌دهد:

```
Xml _ parser _ free ($parser) ;
```

تنظیم کنترل‌کننده‌های XML

زبان نشانه‌گذاری XML شامل هفت رخداد مختلف است و هریک از آنها با یک کنترل‌کننده همراه می‌باشند. در این میان ما تنها دو مورد از این هفت مورد را که بیشترین استفاده را دارند، مورد بحث و بررسی قرار می‌دهیم. این دو مورد عبارتند از آغاز و پایان یک نشانه و داده‌های کاراکتری.

جهت‌همراه‌کردن‌یک‌تابع‌بخصوص‌بارخدادهای‌یک‌نشانه، تابع ( )xml_ set _ element _ handler را مورد استفاده قرار می‌دهیم. این تابع جهت عملیات خود به سه آرگومان ورودی نیاز دارد. اولین آرگومان تابع فوق یک منبع پردازشگر است. دومین آرگومان نام کنترل‌کننده بخش آغازین و سومین آرگومان نیز نام کنترل‌کننده بخش پایانی نشانه می‌باشد.

توابع مورد بحث را خود باید ایجاد کنید. جهت ایجاد کنترل‌کننده نشانه آغازین باید از دریافت سه آرگومان ورودی توسط تابع اطمینان حاصل کنیم. اولین آرگومان این تابع یک منبع پردازشگر می‌باشد. دومین آرگومان یک دنباله کاراکتری است که شامل نام نشانه موردنظر می‌باشد و بالاخره سومین آرگومان نیز یک آرایه انجمنی است که حاوی اسامی صفتهای این نشانه می‌باشد؛ همچنین

کنترل‌کننده نشانه پایانی باید به‌گونه‌ای طراحی شود که دو آرگومان ورودی را دریافت نماید. اولین آرگومان این تابع یک منبع پردازشگر و دومین آرگومان نیز نام نشانه موردنظر است. توابع فوق اسامی نشانه‌ها و صفتهای مربوطه را پیش از به‌کارگیری آنها به حروف بزرگ تبدیل خواهند کرد؛ مگر آنکه به‌طور صریح سیاست دیگری را مشخص کرده باشید. قطعه کد زیر نمونه‌ای از تعریف این توابع را نشان می‌دهد:

```
// ...
xml _ set _ element _ handler ($parser, "start _ handler " , "end _ handler " ) ;
// ...
function start _ handler ($parser , $el _ name , $attribs) {
   print "$el _ name : < br > < ul >" ;
   foreach ($attribs as $at _ name ⇒ $at _ val)
      print "< li > $at _ name ⇒ \ "$at _ val \ " " ;
   print "< / ul > " ;
}
function set _ end _ handler ($parser , $el _ name) {
   print "END : $el _ name > < br >" ;
}
```

قطعه کد فوق دو نمونه بسیار ساده از کنترل کننده‌ها را نشان می‌دهد. کنترل‌کننده نشانه آغازین همان‌گونه که مشاهده می‌کنید نام نشانه و همچنین اسامی صفتهای مربوطه را به‌همراه مقادیر هر یک از آنها در قالب یک لیست بدون ترتیب نمایش می‌دهد. این کنترل‌کننده به‌ازای هر یک از نشانه‌های آغازین موجود در سند XML مورد فراخوانی واقع می‌شود. کنترل کننده نشانه پایانی تنها نام نشانه را بار دیگر نمایش می‌دهد.

اکنون که در مورد نشانه‌های آغازین و پایانی و نحوه ایجاد کنترل‌کننده‌های مربوطه مطالب مفیدی فراگرفتید، وقت آن است تا چگونگی دستیابی به محتوای موجود در هر یک از این نشانه‌ها را مورد بررسی قرار دهیم. انجام چنین فرآیندی با تنظیم یک کنترل‌کننده کاراکتری با استفاده از تابع xml _ set _ character _ data _ handler ( ) امکان‌پذیر است. تابع xml _ set _ character _ data _ handler ( ) جهت انجام عملیات موردنظر مستلزم دریافت دو آرگومان ورودی است. آرگومان اول این تابع نام یک منبع پردازشگر معتبر و آرگومان دوم نیز نام یک تابع کنترل‌کننده را مشخص می‌کند. تابع کنترل‌کننده باید به‌گونه‌ای طراحی شود که یک منبع پردازشگر و دنباله کاراکتری موردنیاز را به‌عنوان آرگومان‌های ورودی دریافت نماید. نمونه‌ای از فراخوانی تابع xml _ set _ character _ data _ handler ( ) به‌قرار زیر است:

```
xml set character_data- handler ($parser , "char _ data" ) ;
function char _ data ($parser , $data) {
print "< i > $data < / i > < br >" ;
}
```

درصورتی‌که به مطالعه در مورد سایر رخدادهای XML که توسط زبان برنامه‌نویسی PHP و کتابخانه Expat مورد پشتیبانی قرار گرفته‌اند علاقه‌مند هستید، توصیه می‌کنیم مستندات مربوطه را در آدرس URL زیر مورد بررسی قرار دهید:

http : // www . php . net / manual / en / ref . xml . php

با این همه ما جهت تسهیل در مراجعه لیست کاملی از این رخدادها را در جدول ۱- ۲۲ به‌همراه توابعی که هنگام وقوع هر یک اجرا می‌شوند، گردآوری کرده‌ایم.

جدول ۱- ۲۲ توابع کنترل‌کننده XML و رخدادهای مربوطه

رخداد	تابع کنترل‌کننده
داده‌های کاراکتری	xml_ set _ character _ data _ handler ( )
رخدادهایی‌که‌توسط‌کنترل‌کننده‌های ویژه‌ای تحت پوشش قرار نگرفته‌اند.	xml _ set _ default _ handler ( )
آغاز و پایان نشانه‌ها	xml _ set _ element _ handler ( )
موجودیتهای خارجی	xml _ set external _ entity _ ref _ handler ( )
معرفی notation ها	xml _ set _ notation _ decl _ handler ( )
دستورالعملهای پردازشی	xml _ set _ processing _ instruction _ handler ( )
موجودیتهای پردازش نشده یا NDATA	xml _ set _ unparsed _ entity _ decl _ handler ( )

تابع xml _ parser _ set _ option ( )

همان‌گونه که پیشتر اشاره‌کردیم اسامی نشانه‌های XML در قالب حروف بزرگ به توابع کنترل کننده مختلف ارسال می‌شوند. نکته مهمی که در اینجا قابل اشاره است این است که اسامی نشانه‌های XML باید نسبت به بزرگی و کوچکی حروف حساسیت داشته باشند. با وجود این وضعیت فوق را به‌گونه دلخواه می‌توان تغییر داد. این فرآیند از طریق فراخوانی تابع xml _ parser _ set _ option ( ) قابل پیاده‌سازی می‌باشد. این تابع جهت انجام عملیات پیش‌بینی شده به سه آرگومان ورودی نیاز دارد. اولین آرگومان این تابع نام یک منبع پردازشگر است. آرگومان دوم تابع مورد بحث یک عدد صحیح است و بیانگر گزینه‌ای می‌باشد که هدف از فراخوانی تابع فوق تنظیم آن‌است. آرگومان آخر این تابع نیز مقدار موردنظر برای گزینه‌ای است که قصدمان تنظیم آن است. به‌منظور غیر فعال کردن گزینه‌ای که منجر می‌شود تا اسامی نشانه‌های XML در قالب حروف بزرگ به توابع کنترل‌کننده ارسال شوند،

لازم است تا ثابت سیستمی ویژه‌ای با نام XML _ OPTION _ CASE _ FOLDING را با مقدار عددی صفر تنظیم نماییم. برای انجام یک چنین کاری تابع ( ) xml_ parser _ set _ option را به‌صورت زیر فراخوانی می‌کنیم:

xml _ parser _ set _ option ($parser , XML _ OPTION _ CASE _ FOLDING , 0) ;

علاوه بر تنظیم اندازه حروف ارسالی به توابع کنترل‌کننده، می‌توان شیوه کدگذاری کاراکتر مقصد را نیز به‌خوبی مشخص کرد. برای انجام این کار باز هم تابع( ) xml_ parser _ set _ option را فرا می‌خوانیم. این بار ضمن استفاده از منبع پردازشگر به‌عنوان اولین آرگومان از ثابت سیستمی XML_ OPTION _ TARGET_ ENCODING به‌عنوان آرگومان دوم استفاده کرده و یکی از سه دنباله کاراکتری "ISO _ 8859_1 "، " US _ ASCII " و یا "UTF _ 8 " را به‌عنوان آرگومان سوم مورد استفاده قرار می‌دهیم. این عمل موجب می‌شود تا پردازشگر XML پیش از ارسال داده‌ها به توابع کنترل کننده موجود در برنامه، شیوه کدگذاری کاراکترها را مطابق این تنظیم تغییر دهد. طبق پیش‌فرض شیوه کدگذاری مورد استفاده در مقصد همان شیوه کدگذاری است که در منبع مورد استفاده قرار می‌گیرد (شیوه پیش‌فرض در کدگذاری منبع شیوه‌ای است که توسط استاندارد ISO _ 8859_1 مشخص شده است مگر آنکه شیوه دیگری را با فراخوانی تابع xml _ parser _ create ( ) اتخاذ کرده باشید).

به غیر از دو ثابت سیستمی که با عناوین XML _ OPTION _ CASE _ FOLDING و XML_ OPTION _ TARGET_ ENCODING در پاراگرافهای فوق مورد بررسی قرار دادیم دو ثابت سیستمی دیگر وجود دارند که می‌توان از آنها به‌همراه تابع ( ) xml_ parser _ set _ option مورد استفاده قرار داد. اسامی این ثابتهای سیستمی عبارتند از XML_ OPTION_ SKIP _ TAGSTART و XML _ OPTION_ SKIP_ WHITE.

پردازش سند

تا بدین جا هیچ اقدامی را در جهت پردازش سند XML صورت نداده و تنها به تنظیم شرایط صحیح جهت فرآیند پردازش پرداختیم. برای اینکه به‌طور واقعی درگیر فرآیند پردازش اسناد XML شویم لازم است تابعی با عنوان ( ) xml_ parse را مورد فراخوانی قراردهیم. تابع( ) xml_ parse جهت انجام عملیات موردنظر (پردازش سند) نیازمند دریافت دو آرگومان می‌باشد. اولین آرگومان از تابع یک منبع پردازشگر معتبر و دومین آرگومان نیز دنباله‌ای کاراکتری است که شامل سند XML موردنظر (سند مورد پردازش) می‌باشد. در صورت تمایل می‌توان تابع ( ) xml _ parse را چندین مرتبه به‌طور متوالی در قالب یک ساختار تکرار مورد فراخوانی قرار داد. بدین ترتیب می‌توان پردازش یک سند XML بزرگ را به چندین مرحله تقسیم نمود. تنها نکته‌ای که باید در این مورد رعایت کرد این است که تابع ( ) xml _ parse باید از این وضعیت مطلع باشد. این بدان معنی است که تابع فوق به‌طور

پیش‌فرض دنباله‌های کاراکتری را که طی فراخوانی‌های متوالی دریافت می‌کند، بخشی از یک سند واحد به‌حساب نمی‌آورد. اما برای اینکه این رفتار پیش‌فرض تابع ( ) xml _ parse به‌صورتی که مد نظر ماست تغییر کند لازم است تا آرگومان سومی را که یک عدد صحیح است به‌عنوان یک آرگومان اختیاری به این تابع ارسال نماییم. ارسال هر عدد صحیح مثبتی به‌عنوان این آرگومان اختیاری به تابع ( ) xml _ parse رفتار پیش‌فرض فوق را به نفع ما تغییر خواهد داد. به‌نمونه‌ای از فراخوانی این تابع توجه نمایید:

```
$xml _ data = "< ? xml version = "1 . 0" ? > < banana- news > < test / >
  < / banana- news > " ;
xml _ parse ($parser , $xml _ data , 1) ;
```

تابع ( ) xml _ parse به‌عنوان نتیجه عملیات خود مقداری از نوع Boolean را به برنامه‌ای که آن‌را مورد فراخوانی قرار داده‌است، باز می‌گرداند. همان‌گونه که حدس می‌زنید مقدار true نشانه موفقیت در عملیات این تابع بوده و مقدار false بیانگر وقوع خطایی در عملیات و بنابراین عدم موفقیت است.

گزارش خطا

هنگامی که سندی از نوع XML را مورد پردازش قرار می‌دهید همواره باید احتمال وقوع خطا را حین پردازش سند XML درنظر داشته باشید. درصورتی‌که حین پردازش یک سند XML خطایی به وقوع بپیوندد پردازشگر XML فرآیند مربوطه را متوقف می‌کند اما توجه کنید که در این حالت هیچ‌گونه پیغامی جهت اطلاع کاربر از روند عملیات بر روی پنجره مرورگر اینترنت به‌نمایش در نمی‌آید. بدین ترتیب وظیفه دیگر برنامه‌نویس در اینجا مشخص می‌شود، بدین معنی که باید در چنین مواقعی با نمایش یک پیغام با معنی و متناسب خطا که شامل توصیف خطا و خطی از سند XML که هنگام پردازش آن خطا به‌وقوع پیوسته است، کاربر را آگاه کند.

توابع کتابخانه Expat تنها خطاهایی را گزارش می‌دهند که مربوط به عدم خوش‌طرح بودن سند XML می‌باشند. ویژگی خوش‌طرح بودن یا well _ formedness در مورد اسنادی از نوع XML صدق می‌کند که دستور زبان (مجموعه قوانین) مربوط به این زبان نشانه‌گذاری را رعایت کرده باشند. به عبارت دیگر فرآیند اعتبارسنجی سند XML با توجه به قواعد و محدودیتهای یاد شده در فایل یا سند DTD مربوطه هیچ رابطه‌ای با خوش‌طرح بودن ندارد.

بررسی اینکه آیا حین پردازش سند XML خطایی رخ داده است یا خیر، کار بسیار ساده‌ای است و برای انجام آن کافی است تا مقدار بازگشتی حاصل از فراخوانی تابع ( ) xml _ parser را مورد ارزیابی قرار دهیم. درصورتی‌که حین پردازش سند XML خطایی رخ داده باشد، پردازشگر XML شماره مربوط به آن خطا را درجایی ذخیره می‌کند. برای دستیابی به این شماره خطا می‌توان از تابعی

با نام xml _ get _ error _ code ( ) استفاده نمود. تابع xml _ get _ error _ code ( ) جهت انجام عملیات موردنظر تنها به یک منبع پردازشگر معتبر نیاز دارد. نمونه‌ای از فراخوانی این تابع به شکل زیر است:

```
$code = xml _ get _ error _ code ($parser ) ;
```

درصورت وقوع خطا شماره بازگشتی از این تابع یک عدد صحیح است که باید با یکی از کدهای خطای PHP که در قالب مقادیر ثابتی مانند XML_ ERROR_ TAG _ MISMATCH معین می‌شوند. مطابقت نماید. با در دست داشتن شماره خطا می‌توان پیغام خطای مربوطه را مورد دستیابی قرار داد. شماره خطا یک عدد صحیح است که هیچ‌گونه اطلاعی در مورد کیفیت خطا در دسترس برنامه‌نویس قرار نمی‌دهد. اصلاح چنین خطایی اگر غیر ممکن نباشد، بسیار طاقت‌فرساست. در مقابل می‌توان این شماره خطا را به تابعی با نام xml_ error_ string ( ) ارسال کرد. تابع xml_ error_ string ( ) به‌سادگی شماره کد خطای معتبری را در قالب آرگومان ورودی پذیرفته و پیغام معنی داری را در مورد خطا در قالب یک دنباله کاراکتری به برنامه فراخواننده باز می‌گرداند. نمونه‌ای از فراخوانی تابع xml_ error_ string ( ) را جهت نمونه ارائه می‌کنیم:

```
$str = xml_ error_ string ( $ code ) ;
```

اکنون به تنها چیزی که در مورد خطای حاصل شده نیاز داریم شماره خطی از سند XML است که خطا در آن‌جا واقع شده است. برای انجام این کار نیز تمام کار موردنیاز فراخوانی یک تابع با نام xml_ get_ current_ line _ number ( ) است. تابع xml_ get_ current_ line _ number ( ) جهت انجام عملیات موردنظر تنها نیازمند دریافت یک منبع پردازشگر معتبر به‌عنوان تنها آرگومان ورودی است. تابع مورد بحث به‌عنوان نتیجه عملیات خود، شماره خطی از سند XML را که خطای حین پردازش در آنجا واقع شده است به برنامه فراخواننده بازمی‌گرداند. از آنجا که پردازنده XML در ازای هر خطای پردازش که با آن مواجه می‌شود عملیات خود را متوقف می‌کند، شماره خط جاری در لحظه توقف عملیات پردازش تحت این شرایط شماره خطی از سند XML خواهد بود که در آن خطای حین پردازش به‌وقوع پیوسته است. به نمونه‌ای از فراخوانی این تابع توجه کنید:

```
line = xml _ get_ current_ line_ number ($parser) ;
```

با جمع‌بندی مطالبی که در مورد خطای حین پردازش عنوان کردیم اکنون باید بتوانیم تابعی را جهت گزارش یک خطای به‌وقوع پیوسته ایجاد نماییم. تعریف این تابع چنین خواهد بود:

```
Function format _ error ($p) {
   $code = xml_ get_ error_ code ($p) ;
   $str = xml _ error _ string ($code) ;
   $line = xml _ get _ current _ line _ number ($p) ;
   return "XML ERROR ($code) : $str at line $line" ;
}
```

برنامه موجود در لیست ۲- ۲۲ شامل تمام قطعه کدهایی است که در این مورد ارائه شده‌اند.

```
 1: <?php
 2:
 3: $parser = xml_parser_create();
 4: xml_parser_set_option( $parser, XML_OPTION_CASE_FOLDING, 0 );
 5:
 6: xml_set_element_handler( $parser, "start_handler", "end_handler" );
 7: xml_set_character_data_handler( $parser, "char_data" );
 8: xml_parser_set_option( $parser, XML_OPTION_CASE_FOLDING, 0 );
 9: $xml_str = implode('', file( "listing22.1.xml", 0 ));
10:
11: xml_parse( $parser, $xml_str )
12:     or die( format_error( $parser ) );
13:
14: function start_handler( $parser, $el_name, $attribs ) {
15:     print "START: $el_name: <br>";
16:     foreach( $attribs as $at_name=>$at_val )
17:         print "    $at_name=>\"$at_val\"<br>";
18: }
19:
20: function end_handler( $parser, $el_name ) {
21:     print "END: $el_name><br>";
22: }
23:
24: function char_data( $parser, $data ) {
25:     print "        char data:<i>$data</i><br>";
26: }
27:
28: function format_error( $p ) {
29:     $code = xml_get_error_code( $p );
30:     $str = xml_error_string( $code );
31:     $line = xml_get_current_line_number ( $p );
32:     return "XML ERROR ($code): $str at line $line";
33: }
34:
35: ?>
```

لیست ۲- ۲۲ پردازش یک سند XML

همان‌گونه که در این لیست مشاهده می‌کنید، ابتدا در خط ۳ یک پردازشگر XML ایجاد شده است. سپس در خطوط ۶ تا ۸ از این لیست کنترل‌کننده‌های رخداد فعال (فراخوانی) شده‌اند. ما در این لیست تعریف هر یک از این کنترل‌کننده‌ها را نیز ارائه کرده‌ایم. تعریف کنترل‌کننده start _ handler( ) در خط ۱۴، تعریف کنترل‌کننده end _ handler ( ) در خط ۲۰ و بالاخره تعریف کنترل‌کننده char _ data ( ) در خط ۲۴ از این لیست انجام شده است. لیست ۲- ۲۲ به‌سادگی هر محتوایی را که با آن مواجه می‌شود، بر روی پنجره مرورگر اینترنت ارسال می‌کند. این برنامه عملکرد پردازشگر XML را به‌خوبی نمایش می‌دهد، با این حال استفاده چندانی ندارد. در قسمت بعدی از این درس برنامه کوچکی ایجاد می‌کنیم که خروجی محسوس‌تر و قابل استفاده‌تری دارد.

بررسی یک مثال

همان‌گونه که از ابتدای درس این ساعت نیز متوجه شدید، به عناوین خبری خاصی در مورد یک محصول فرضی علاقه‌مندیم. شرکای تجاری ما اخبار مهمی را در زمینه کاری مشترکمان تدارک دیده‌اند که شامل یک سند XML است. قصد ما از نوشتن برنامه نهایی این است که تنها عناوین خبری و اسامی نویسندگان مقالات را از درون این سند استخراج نماییم و بر روی صفحه نمایش دهیم.

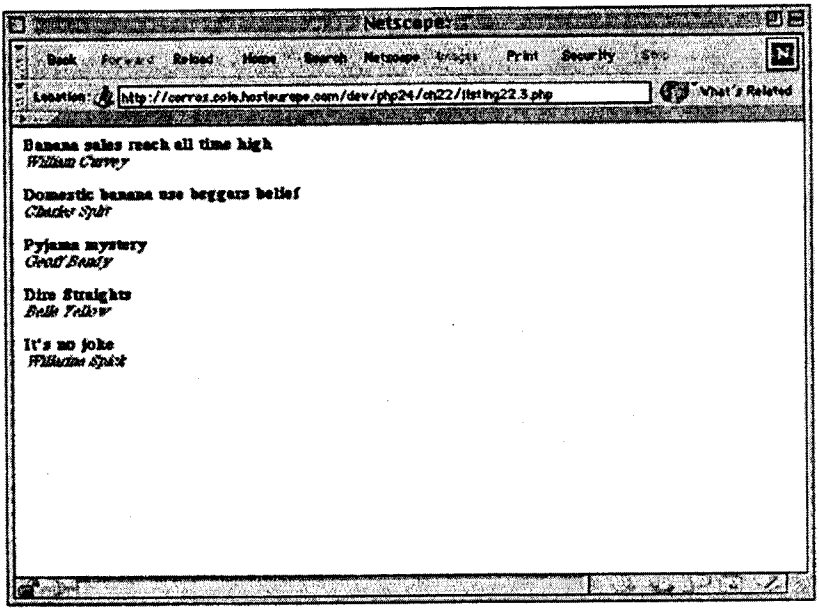
در حال حاضر به تمام ابزارهای موردنیاز جهت حصول نتیجه مطلوب مجهز می‌باشیم. در این قسمت تنها مطلب جدیدی را که عنوان خواهیم کرد، عبارت از یک تکنیک یا روش است. کد برنامه موردنظر در لیست ۳- ۲۲ قابل بررسی و مطالعه است.

```
1: <?php
2: $open_stack = array();
3: $parser = xml_parser_create();
4: xml_set_element_handler( $parser, "start_handler", "end_handler" );
5: xml_set_character_data_handler( $parser, "character_handler");
6: xml_parser_set_option( $parser, XML_OPTION_CASE_FOLDING, 0 );
7: xml_parse( $parser, implode('', file( "listing22.1.xml" )) ) or die(
➥format_error( $parser ) );
8: xml_parser_free( $parser );
9:
10: function start_handler( $p, $name, $atts ) {
11:         global $open_stack;
12:         $open_stack[] = array($name, "");
13: }
14:
15: function character_handler( $p, $txt ) {
16:     global $open_stack;
17:     $cur_index = count($open_stack)-1;
18:     $open_stack[$cur_index][1] .= $txt;
19: }
20:
21: function end_handler( $p, $name ) {
22:         global $open_stack;
23:         $el = array_pop( $open_stack );
24:     if ( $name == "headline")
25:         print "<b>$el[1]</b><br>";
26:     if ( $name == "byline") {
27:         print "<i>$el[1]</i><p>";
28:     }
29: }
30:
31: function format_error( $p ) {
32:     $code = xml_get_error_code( $p );
33:     $str = xml_error_string( $code );
34:     $line = xml_get_current_line_number ( $p );
35:     return "XML ERROR ($code): $str at line $line";
36: }
37:
38: ?>
```

لیست ۳- ۲۲ بررسی مثالی در مورد پردازش یک سند XML

چنانکه در این برنامه نمونه ملاحظه می‌کنید، ابتدا در خط ۲ از لیست یک آرایه سراسری با نام $open _ stack را معرفی کرده‌ایم. از این آرایه سراسری جهت تشخیص نشانه جاری در هر لحظه از فرآیند پردازش سند XML استفاده خواهیم کرد. در خطوط ۳ تا ۶ از برنامه یک پردازشگر XML ایجاد شده و کنترل‌کننده‌های رخداد همگی فعال (فراخوانی) شده‌اند. با این اقدام این کنترل‌کننده‌ها همواره در وضعیت آماده به کار خواهند بود و بدین ترتیب به محض وقوع رخداد مربوطه فراخوانی خواهند شد. هنگامی که پردازشگر XML با یک نشانه آغازین مواجه می‌شود، تابع یا کنترل‌کننده start _ handler ( ) (که تعریف آن در خط ۱۰ برنامه آمده است مورد فراخوانی قرار می‌گیرد. همان‌گونه که مشاهده می‌کنید در تعریف تابع start _ handler ( ) آرایه‌ای شامل دو عنصر، که متشکل از نام نشانه و یک دنباله کاراکتری تهی است، ایجاد نموده و آن‌را به انتهای آرایه $open _ stack ( ) اضافه می‌کنیم (خط ۱۲ را ببینید). به‌محض اینکه پردازشگر XML با داده‌های کاراکتری مواجه می‌شود، تابع character _ handler ( ) مورد فراخوانی قرار می‌گیرد. جهت دستیابی به آخرین نشانه‌ای از سند XML که اخیراً باز شده است می‌توان همواره به آخرین عنصر از آرایه $open _ stack مراجعه نمود. در خط ۱۸ از برنامه چنانکه مشاهده می‌کنید داده‌های کاراکتری را به دومین عنصر از آرایه مذکور که بیانگر آخرین نشانه XML مورد دستیابی قرار گرفته است، اضافه کرده‌ایم. به‌طور مشابه هنگامی که پردازشگر XML به نشانه پایانی می‌رسد تابع end _ handler ( ) که تعریف آن در خط ۲۱ ارائه شده است، فراخوانی می‌گردد. به‌عنوان بخشی از تعریف این تابع در خط ۲۳ از برنامه، ابتدا آخرین عنصر از آرایه $open _ stack حذف می‌شود. آرایه‌ای که به‌عنوان نتیجه این عملیات بازگردانده می‌شود باید تنها شامل دو عنصر باشد. عنصر اول این آرایه نام نشانه‌ای از سند XML است که به‌تازگی بسته شده است. آرگومان دوم نیز هر نوع داده کاراکتری است که نشانه مذکور آن‌را شامل می‌شود. چنانکه عنصر موردنظر همانی است که ما قصد نمایش آن‌را داریم می‌توانیم روند عملیات را با اعمال قالب‌بندی موردنظرمان ادامه دهیم.

خروجی حاصل از اجرای این برنامه در شکل ۲-۲۲ قابل مشاهده و بررسی است. جهت طبیعی‌تر جلوه‌کردن این عملیات از یک سند XML واقعی‌تر استفاده شده است.



شکل ۲- ۲۲ حاصل فرآیند پردازش یک سند XML و ارسال نتایج به صورت قالب‌بندی شده به مرورگر اینترنت

مقدمه‌ای بر توابع DOM

همان‌گونه که در قسمت قبل مشاهده کردید توابعی که جهت پردازش اسناد XML مورد استفاده قرار می‌گیرند، برمبنای وقوع رخداد عمل می‌کنند. این بدان معنی است که سند XML موردنظر توسط پرداشگر XML از بالا به پایین (از ابتدا به انتها) مورد بازخوانی قرار می‌گیرد و طی این فرآیند به محض وقوع یک رخداد (مانند مواجهه با نشانه آغازین یا پایانی و یا مواجهه با داده‌های کاراکتری) اقدام به فراخوانی کنترل‌کننده رخداد مربوطه می‌نماید. برخلاف مدل رخداد، مدل DOM که کوتاه شده اصطلاح Document Object Model است مدلی برمبنای ساختار سلسله مراتبی یا درختی می‌باشد. در این مدل پردازش سند XML مورد بازخوانی قرارگرفته و یک تصویر درختی با توجه به طبیعت نشانه‌های زبان XML درنظر گرفته می‌شود. بدین ترتیب می‌توان این درخت را به شیوه دلخواه مورد پیمایش قرار داده و اجزای مختلف آن‌را بررسی نمود. فرآیند معکوس نیز امکان‌پذیر است. بدین معنی که می‌توان ابتدا یک ساختار درختی از نشانه‌های مختلف XML و محتوای موردنظر ایجاد کرده و سپس بر مبنای مدل پردازش DOM سند XML مربوطه را از درون آن ساختار استخراج نمود.

پشتیبانی از مدل DOM در زبان برنامه‌نویسی PHP در حال حاضر تحت توسعه است، از این‌رو هیچ ضمانت اجرایی در مورد برنامه‌ها و کدهای نمونه‌ای که در این قسمت ارائه می‌شود، وجود نخواهد داشت. با این وجود بهتر است در صورت تمایل خواننده مستندات مربوط به این مدل پردازش و کم و کیف پشتیبانی از آن‌را از طریق آدرس زیر مطالعه نماید:

http : // www . php . net / manual / en / ref . domaxml . php

جهت بهره‌برداری از توابعی که در این قسمت از درس مورد بحث قرار می‌گیرند لازم است تا ابتدا پیش از هر چیز کتابخانه XML منتشر شده از جانب Gnome را که libxml نام دارد، به‌منظور استفاده در برنامه‌های PHP بر روی کامپیوترتان نصب کنید. ما بر استفاده از نسخه 7 . 2 . 2 از این کتابخانه که در زمان انتشار کتاب حاضر جدیدترین نسخه منتشر شده بوده است، تأکید می‌کنیم. اگر به این کتابخانه دسترسی ندارید می‌توانید از طریق آدرس زیر آن‌را مورد دستیابی قرار دهید:

http : // www. xmlsoft . org

علاوه بر این لازم است تا PHP را جهت پشتیبانی از مدل DOM مجدداً کامپایل نمایید. نحوه انجام این عمل چنانکه بارها و بارها در طول کتاب عنوان کردیم به‌صورت زیر است. کافی است تا عبارت زیر را :

- - with - dom = / path / to / libxml / distrib

به‌منظور پیکربندی مجدد به گزینه‌های سطر فرمان اضافه نمایید (جهت مشاهده اطلاعات بیشتر در زمینه نصب PHP و بسته‌های نرم‌افزاری موردنیاز به‌همراه آن به درس ساعت دوم با عنوان "نصب PHP4 بر روی کامپیوتر" مراجعه کنید). درصورتی‌که کتابخانه libxml را در موقعیت نامتعارفی نصب نکرده باشید، این تنظیم به شکل زیر خواهد بود:

- - with - dom = / usr

اولین‌چیزی که جهت کار با توابع کتابخانه DOM که با عنوان libxml آن‌را بر روی کامپیوترتان نصب می‌کنید، نیاز دارید شئ بسیار مهمی با عنوان Dom Document است. شئ DomDocument ظرفی برای تمامی نشانه‌های موجود در سند XML است. تمامی نشانه‌های مذکور در این ظرف در قالب یک شئ مورد استفاده قرار می‌گیرند.

دستیابی به شئ DomDocument

جهت ایجاد شئ DomDocument و بهره‌برداری از آن، لازم است تابعی با عنوان new_ xmldoc ( ) را فراخوانی کنیم. این تابع جهت انجام عملیات موردنظر به یک آرگومان ورودی از نوع دنباله کاراکتری نیاز دارد. این دنباله کاراکتری شامل سند XML موردنظر، یعنی سند مورد پردازش می‌باشد. آرگومان ورودی را به هر صورت ممکن می‌توان مورد استفاده قرار داد. از این‌رو از تابع new_ xmldoc ( ) هنگامی استفاده می‌شود که بخواهیم یک ساختار درختی را از ابتدای امر ایجاد نماییم. کمترین چیزی که این تابع به عنوان آرگومان بدان نیاز دارد، معرفی XML است:

```
$doc = new _ xmldoc ( ' < ? xml version = "1 . 0" ? > ' ) ;
```

در فراخوانی فوق، متغیر $doc شامل اشاره‌گری به شئ DomDocument خواهد بود. بدین ترتیب با در دست داشتن این شئ می‌توان سایر نشانه‌های موردنظر را نیز به ساختار درختی اضافه نمود. در حقیقت می‌توان به‌عنوان آرگومان تابع مورد بحث، تنها شماره نسخه XML را که مورد نظرمان است استفاده نمود (این شماره تقریباً در تمامی موارد به‌صورت دنباله کاراکتری "1 . 0" به تابع ارسال می‌شود). در این حالت نیز می‌توان نشانه‌های موردنظر را به ساختار اضافه کرد. فراخوانی زیر نحوه انجام این عمل را نشان می‌دهد:

```
$doc = new _ xmldoc ( ' 1 . 0 ' ) ;
```

اگر قصد ما پردازش یک ساختار آماده باشد، می‌توانیم به‌جای تابع فوق از تابع دیگری با عنوان xmldocfile ( ) استفاده نماییم. این تابع نام تابعی را که در حال حاضر حاوی سند XML موردنظر است‌درقالب یک دنباله کاراکتری دریافت کرده و به عنوان نتیجه عملیات، شئ از نوع DomDocument را به برنامه فراخواننده باز می‌گرداند. به نمونه‌ای از فراخوانی تابع xmldocfile ( ) توجه کنید:

```
$doc = xmldocfile ( "listing 22 . 1 . xml " ) ;
```

مفهوم نشانه ریشه

همان‌گونه که مدل DOM روشی را برای شبیه‌سازی اسناد XML فراهم می‌سازد، شئ را نیز جهت شبیه‌سازی نشانه‌های موجود در اسناد مذکور در اختیار قرار می‌دهد. دو شئ DomElement و DomDocument هر دو از یک شئ واحد با عنوان DomNode مشتق شده‌اند و از این‌رو ساختارهای مشابهی دارند.

طبق قوانین تدوین شده در مورد اسناد XML، هر سندی از این نوع باید شامل یک نشانه ریشه باشد. به‌طور مشابه، هر شئ از نوع DomDocument یک شئ ریشه با عنوان DomElement دارد. دسترسی یا ایجاد چنین شئ کاملاً امکان‌پذیر است. جهت تعریف یک نشانه به‌عنوان نشانه ریشه سند XML می‌توانیم از متد add _ root ( ) استفاده کنیم. این متد جهت انجام عملیات موردنظر به یک آرگومان از نوع دنباله کاراکتری نیاز دارد. این دنباله کاراکتری نام نشانه‌ای است که قصد ایجاد آن‌را داریم. تابع add _ root ( )، شیئ از نوع DocElement را به‌عنوان نشانه ریشه و آن‌را به ساختار درختی سند مورد نظر اضافه می‌کند و درنهایت آن شئ را به برنامه فراخواننده بازمی‌گرداند. به نمونه فراخوانی این تابع که جهت ایجاد نشانه ریشه‌ای با نام banana _ news مورد استفاده قرار گرفته است، توجه نمایید:

```
$root = $doc → add _ root ( "banana _ news ") ;
```

جهت دستیابی به یک نشانه ریشه موجود که شئ از نوع DomElement است، کافی است تابعی با عنوان root ( ) را بدون هیچ آرگومانی فراخوانی نمایید.

افزودن نشانه‌های جدید به ساختار درختی

پس از اینکه نشانه ریشه یک سند XML را به‌واسطه فراخوانی تابع ( ) root _ add مورد دستیابی قرار دادید، می‌توانید نشانه‌های بیشتری را نیز با بهره‌گیری از تابع یا به‌طور دقیق‌تر متد ( ) new_child از شئ DomElement به ساختار درختی سند اضافه نمایید (بار دیگر یادآوری می‌کنیم که نشانه ریشه یک سندXML را می‌توانید از طریق فراخوانی یکی از دو متد ( ) root _ add یا ( ) root از شئ DomDocument مورد دستیابی قراردهید. متد( ) new_child ساختار نسبتاً ساده‌ای داشته به‌گونه‌ای که تنها به دو آرگومان ورودی جهت اجرای عملیات پیش‌بینی شده احتیاج دارد. اولین آرگومان این تابع یک دنباله کاراکتری است که بیانگر نام نشانه موردنظر است. دومین آرگومان تابع ( ) new_child نیز یک دنباله کاراکتری دیگر است که (در صورت وجود) شامل توصیف نشانه مذکور خواهد بود. درصورتی‌که نشانه‌ای فاقد این توصیف باشد، لازم است تا از یک دنباله کاراکتری تهی به‌جای آن توصیف استفاده شود. در چنین حالتی عدم تأمین این دنباله کاراکتری تهی منجر به سردرگمی متد موردبحث خواهد شد. به نمونه‌ای از این فراخوانی جهت اضافه کردن دو نشانه جدید به‌ترتیب با اسامی "newsitem" و "headline" توجه کنید. دقت کنید که نشانه اول فاقد توصیف است:

```
$item = $root → new _ child ( "newsitem " , " " ) ;
$item → new _ child ( " headline " , "The Banana Story " ) ;
```

متد ( ) new _ child به‌عنوان نتیجه عملیات خود یک شئ جدید از نوع DomElement را باز می‌گرداند که درصورت نیاز می‌توان از آن شئ جهت اضافه کردن نشانه‌های جدید دیگر استفاده نمود.

با دانشی که تا بدین جا در مورد ساختار سلسله مراتبی DOM و توابع مربوطه جهت ایجاد این ساختار فراگرفتید، اکنون می‌توانیم ساختار درختی سند XML موجود در لیست ۱- ۲۲ را که در ابتدای درس این ساعت ارائه دادیم، ایجاد نماییم. در برنامه‌ای که در این قسمت ارائه کرده‌ایم از یک آرایه انجمنی جهت تأمین داده‌های سند مورد بحث استفاده کرده‌ایم (این آرایه چنانکه ملاحظه می‌کنید در خط ۲ با عنوان news ایجاد شده است). با این حال توجه کنید که در برنامه‌های کاربردی واقعی در بیشتر اوقات این داده‌ها از طریق یک بانک اطلاعاتی تأمین می‌شوند. برنامه مذکور در لیست ۴- ۲۲ ارائه شده است.

```
 1: <?php
 2: $news = array(
 3:     array(  "headline" => "Banana sales",
 4:         "image" => "/res/high.gif",
 5:         "byline" => "William Curvey",
 6:         "article" => "Research published today by...",
 7:         "type" => "world"
 8:         ),
 9:     array(  "headline" => "Domestic banana use beggars belief",
10:         "image" => "/res/use.gif",
11:         "byline" => "Charles Split",
```

```
12:         "article" => "Bananas are for more than eating...",
13:         "type" => "world"
14:         )
15: );
16:
17: $doc = new_xmldoc("1.0");
18:
19: $root = $doc->add_root("banana-news");
20: foreach( $news as $newselement ) {
21:     $item = $root->new_child("newsitem", "");
22:     $item->set_attribute("type", $newselement['type'] );
23:     $item->new_child( "headline", $newselement['headline'] ) ;
24:     $item->new_child( "image", $newselement['image'] ) ;
25:     $item->new_child( "byline", $newselement['byline'] ) ;
26:     $item->new_child( "article", $newselement['article'] ) ;
27: }
28: print $doc->dumpmem( );
29: ?>
```

لیست ۴- ۲۲ ایجاد یک سند XML با استفاده از توابع DOM

تنها نکته جدیدی که در رابطه با کد موجود در لیست ۴- ۲۲ وجود دارد فراخوانی یکی از متدهای شئ DomDocument با عنوان ( ) dumpmem است. متد ( ) dumpmem که در خط ۲۸ از این برنامه مورد فراخوانی واقع شده است کل ساختار درختی ایجاد شده از روی سند XML را در قالب یک دنباله کاراکتری به برنامه فراخواننده بازمی‌گرداند و بدین ترتیب تابع ( ) print قادر خواهد بود تا آن‌را بر روی صفحه نمایش دهد. از این‌رو خروجی حاصل از اجرای برنامه موجود در لیست ۴- ۲۲ مشابه خروجی خواهد بود که از اجرای برنامه لیست ۱- ۲۲ به دست می‌آید. با این همه هنوز یک تفاوت کوچک وجود دارد و آن اینکه تابع ( ) dumpmem خروجی خود را با بهره‌گیری از علایم خط جدید با سایر علایم تزئین نمی‌کند.

دستیابی به اطلاعات موجود در اشیای DomElements

معمولاً اولین اطلاعاتی که مایلیم تا در مورد شیئی از نوع DomElement به دست آوریم، نام نشانه مربوطه است. این نام در قالب خصوصیتی از شئ DomElement با عنوان $tagname نگهداری می‌شود و دسترسی به آن همان‌گونه که در عبارت نمونه زیر مشاهده می‌کنید، به‌آسانی امکان‌پذیر است:

```
Print " I am a " . $el  → tagname . " element " ;
```

در عبارت فوق فرض بر این است که $el مرجعی به یک شئ از نوع DomElement می‌باشد. مطلب بعدی مربوط به صفات موجود در یک نشانه است. همین‌که نام نشانه مورد دستیابی قرار گرفت، به‌راحتی می‌توان نام تمامی صفتهای آن‌را به دست آورد. صفتهای مذکور همواره در شیئی از نوع DomAttributeنگهداری‌می‌شوند.به‌واسطه‌فراخوانی‌متدی با عنوان ( ) attributes از شئ DomElement.

می‌توان به‌سادگی آرایه‌ای از اشیای DomAttribute را مورد دستیابی قرار داد. عبارت زیر نحوه انجام این کار را نشان می‌دهد:

```
$atts = $el → attributes ( ) ;
```

با فراخوانی آرایه $atts شامل عناصری از نوع DomAttribute خواهد بود. هر یک از این اشیا معادل یکی از صفتهای نشانه موردنظر ($el) می‌باشند. جهت دستیابی به هریک از زوجهای نام و مقدار ذخیره شده در این اشیای DomAttribute کافی است تا در قالب یک ساختار تکرار متدهای name( ) و value ( ) هریک از آنها را به‌ترتیب جهت دستیابی به نام و مقدار فراخوانی نماییم. قطعه کد زیر روند انجام این عملیات را نشان می‌دهد:

```
$atts = $el → attributes ( ) ;
if ( ! empty ( $atts ) ) {
   foreach ( $atts as $att ) {
      print $att  → name ( ) . " : " . $att  → value ( ) . "< br >" ;
   }
}
```

همان‌گونه که در این قطعه کد ملاحظه می‌کنید، نشانه $el جهت فراخوانی متد attributes ( ) مورد فراخوانی قرار گرفته است. حاصل این فراخوانی که یک آرایه متشکل از اشیایی از نوع DomAttribute است در متغیری با نام $att ذخیره می‌شود و پس از اطمینان از عدم تهی بودن این آرایه نام و مقدار ذخیره شده توسط هر یک از این اشیا با بهره‌گیری از متدهای name ( ) و value ( ) مورد دستیابی واقع شده و با استفاده از تابع print ( ) بر روی صفحه نمایش داده می‌شوند.

یکی از مزایای ساختار درختی پیمایش آن‌است. فرآیند پیمایش به ما اجازه می‌دهد تا در میان گره‌های ساختار درختی حرکت کرده و پردازش لازم را بر روی داده‌های ذخیره شده در هر یک از آنها انجام دهیم. برای انجام این کار لازم است تا از متدهای پیش‌بینی شده در اشیایی از نوع DomElement استفاده کنیم. با بهره‌گیری از این متدها می‌توانیم عملیاتی چون دستیابی به اولین فرزند یا بررسی وجود فرزند را در مورد یک نشانه خاصی انجام دهیم.

با در دسترس داشتن شیئی از نوع DomElement می‌توان تشخیص داد که آیا این شئ دارای فرزندی می‌باشد یا خیر. برای انجام این کار کافی است تا از شئ مزبور متدی با عنوان has _ child _ nodes ( ) را فراخوانی نماییم. چنان‌چه این شئ شامل فرزندی باشد، متد یاد شده مقدار true و در غیر این‌صورت مقدار false را به برنامه فراخواننده باز می‌گرداند. متد مورد بحث فاقد آرگومان است. به نمونه‌ای از نحوه فراخوانی این متد توجه نمایید:

```
if ( $el  → has _ child _ nodes ( ) )
   Print " I am blessed with progeny " ;
```

چنان‌چه شئ مورد بررسی دارای فرزندی باشد، می‌توان به اولین فرزند آن دسترسی پیدا کرد. اولین فرزند هر شئ از نوع DomElement با بهره‌گیری از متدی با نام frist _ child ( ) از این شئ قابل

دستیابی است. این متد نیز به‌مانند متد قبل فاقد آرگومان ورودی است. متد مورد بحث نام اولین فرزند شئ مورد بررسی را درصورت وجود به برنامه فراخواننده بازمی‌گرداند. چنانچه شئ فوق فاقد فرزند باشد، این مقدار false را به‌عنوان نتیجه بازخواهد گرداند. به نمونه‌ای از فراخوانی این متد توجه کنید :

```
if ( $el → has _ child _ nodes ( ) )
   $child = $el → first _ child ( ) ;
```

همان‌گونه که احتمالاً متوجه شده‌اید با بهره‌گیری از متدهایی که در پاراگرافهای قبل بررسی کردیم، می‌توانیم ساختار درختی یک سند XML را از بالا به پایین، یعنی به‌شکل عمودی پیمایش کنیم. یک پرسش منطقی می‌تواند این باشد که آیا همین فرآیند را می‌توان از چپ به راست (یا بالعکس)، یعنی در راستای افقی نیز انجام داد؟ واقعیت این است که نشانه‌های XML از وجود سایر فرزندان پدر خود کاملاً مطلعند. بدین ترتیب باید بتوان به‌روشی نشانه‌های موجود در یک سطح از ساختار درختی را از همان سطح مورد دستیابی قرار داد. این فرآیند از طریق فراخوانی متد next _ sibling ( ) از شئ DomElement امکان‌پذیر است. در مقابل این متد که نشانه بعدی یک نشانه فرضی موجود در یک سطح مشخص از ساختار درختی را به دست می‌دهد، متد دیگری از شئ DomElement با عنوان previous _ sibling ( ) امکان دستیابی به نشانه قبلی از همان سطح از ساختار درختی را فراهم می‌کند. هیچ‌کدام از این متدها آرگومان ورودی دریافت نکرده و درصورتی‌که نشانه موردنظر موجود باشد مرجع مربوطه را به برنامه فراخواننده بازمی‌گردانند. چنان‌چه نشانه قبلی یا بعدی توسط این متدها یافت نشود، متدهای فوق مقدار false را به برنامه فراخواننده باز خواهند گرداند. قطعه برنامه زیر چگونگی بهره‌گیری از متدهای next _ sibling ( ) و previous _ sibling ( ) را جهت پیمایش افقی یک ساختار درختی نشان می‌دهد:

```
$child = $el → first _ child ( ) ;
do {
   print $child → tagname . "< br >" ;
} while ($child = $child → next _ sibling ( ) ) ;
```

علاوه بر این هر نشانه پدر می‌تواند تمامی نشانه‌های فرزند خود را مورد دستیابی قرار دهد. این فرآیند از طریق فراخوانی متدی با نام children ( ) از شئ DomElement انجام‌پذیر است. متد children ( ) هیچ آرگومانی را به‌عنوان ورودی دریافت نمی‌کند اما در مقابل آرایه‌ای را به برنامه فراخواننده بازمی‌گرداند که شامل مراجعی به کلیه اشیای فرزند مثل شئ مورد نظر می‌باشد. درصورتی‌که نشانه موردنظر فاقد هیچ‌گونه فرزندی باشد متد children ( ) به‌عنوان نتیجه عملیات مقدار false را به برنامه فراخواننده بازمی‌گرداند. در ادامه نمونه‌ای از بهره‌گیری از این متد را مشاهده می‌کنید:

```
$kids = $el → children ( ) ;
foreach ( $kids as $child ) {
   print $child → tagname . "< br >" ;
}
```

عکس این قضیه نیز درست است بدین معنی که فرزندان یک شئ اطلاع دقیقی از والدین خود دارند. این فرزندان جهت دستیابی به مرجع والدین خود از متدی با نام ( ) parent از شئ DomElement استفاده می‌کنند. این متد به هیچ آرگومان ورودی جهت تعیین نشانه پدر یک نشانه فرزند از ساختار درختی نیاز ندارد.

کار با گره‌های متنی

با دانشی که تا بدین جای درس درباره متدهای دستیابی مختلف در مورد نشانه‌های یک ساختار درختی به دست آوردید، اکنون می‌توانیم درخت یک سند XML را به‌گونه‌ای دقیق‌تر مورد بهره‌برداری و استفاده قرار دهیم. اما باید این نکته را گوشزد کنیم که هنوز مطالب بسیار مهم‌تری درمورد ساختار درختی یک سند XML باقی‌مانده است که باید مورد بررسی قرار دهیم. نشانه‌های XML تنها یکی از انواع گره‌هایی هستند که ما با آنها سر و کار داریم. علاوه بر گره‌های فرزند، انواع دیگری از گره‌ها وجود دارند که باید چگونگی استفاده و کار با آنها را فرا بگیریم. برخی از انواع این گره‌ها عبارتند از گره‌های متنی، گره‌های توضیح و سایر گره‌ها که بحث در مورد آنها خارج از حوصله این کتاب است. در این قسمت از درس قصد ما این است که گره‌های متنی را مورد بررسی قرار دهیم.

گره‌های متنی شاید یکی از با اهمیت‌ترین انواع گره‌های موجود در ساختار درختی یک سند XML محسوب شوند، چراکه به‌واسطه این گره‌ها می‌توانیم به محتوای یک سند دسترسی پیدا کنیم. اولین نکته‌ای که هنگام کار با این نوع گره‌های ساختار باید به‌خاطر داشته باشیم تفاوتی است که میان اشیایی از نوع DomElement و اشیایی از نوع DomText موجود است. هر دو کلاس DomElement و DomText از یک کلاس پدر مشترک استفاده می‌کنند. به‌عبارت دیگر هر دوی این کلاسها فرزندان کلاس DomNode می‌باشند. تمامی اشیایی که از نوع کلاسDomNode هستند دارای خصوصیت مشترکی با عنوان $type می‌باشند. این خصوصیت حاوی مقداری عددی از نوع صحیح است. اعداد صحیح موجود در خصوصیات $type از اشیای DomNode مختلف (مانند اشیای DomElement و DomText) را می‌توان با بهره‌گیری از ثابتهای سیستمی موجود مورد ارزیابی قرار داد. در مورد اشیایی ازنوعDomElement وDomText به‌ترتیب‌می‌توان‌ازثابتهای‌سیستمی با نام XML_ XML ELEMENT_ NODE و XML_ TEXT_ NODE استفاده کرد.قطعه کدی که در ادامه مشاهده می‌کنید با بهره‌گیری از این دو ثابت سیستمی سعی دارد تا نوع گره‌های مورد ارزیابی را تشخیص دهد:

```
If ($child → type = = XML_ ELEMENT_ NODE ) {
   // work with the element
} elseif ( $child → type = = XML_ TEXT_ NODE ) {
   // work with the text node
}
```

پس از تشخیص گره متنی از گره نوع دیگر، به روشی جهت دستیابی به محتوای آن نیاز داریم. برای این‌کار به‌راحتی می‌توانیم از متدی با عنوان node _ value ( ) استفاده کنیم:

```
If ( $child → type = = XML_ TEXT_ NODE ) {
   Print $child → node _ value ( ) ;
}
```

پیمایش یک ساختار درختی با استفاده از دو رویکرد مختلف

اکنون می‌توانیم ادعا کنیم که اطلاعات کافی برای پیمایش (پردازش) یک ساختار درختی را در اختیار داریم. اما برای انجام این‌کار احتیاج به یک روش سازمان‌یافته و قابل اعتماد داریم. در این قسمت از درس دو روش یا دو تکنیکی را که برنامه‌نویسان جهت انجام این فرآیند معمولاً مورد استفاده قرار می‌دهند، با یکدیگر بررسی خواهیم کرد.

در روش اول ابتدا هر یک از گره‌های درخت را به‌نوبت و یکی پس از دیگری مورد دستیابی قرار داده و آن‌را به برنامه فراخواننده بازمی‌گردانیم. برنامه موجود در لیست ۵- ۲۲ این رویکرد را در عمل نشان می‌دهد.

```
1: <?
2:
3: $doc = xmldocfile("listing22.1.xml");
4: $root = $doc->root();
5: $pointer = $root;
6:
7: do {
8:     print $pointer->tagname()."<br>";
9: } while ( $pointer = next_element( $pointer ) );
10:
11: function next_element( $pointer ) {
12:     while ( $pointer = next_node( $pointer ) ) {
13:         if ( $pointer->type == XML_ELEMENT_NODE )
14:             return $pointer;
15:     }
16:     return false;
17: }
18:
19: function next_node( $pointer ) {
20:     if ( $pointer->has_child_nodes() )
21:         return( $pointer->first_child() );
22:     if ( $next = $pointer->next_sibling() )
23:         return $next;
24:     while( $pointer = $pointer->parent() ) {
25:         if ( $next=$pointer->next_sibling() ) {
26:             return $next;
27:         }
28:     }
29: }
30: ?>
```

لیست ۵- ۲۲ پیمایش یک درخت XML

چنانکه در برنامه فوق مشاهده می‌کنید بار اصلی این روش به‌عهده تابعی با عنوان next_ node( ) است که تعریف آن در خط ۱۹ برنامه ارائه شده است. این تابع مرجع یک گره موجود از ساختار درختی XML را به‌عنوان آرگومان ورودی پذیرفته و این موضوع را که آیا گره مربوطه شامل فرزندی‌می‌باشد یا خیر مورد بررسی قرار می‌دهد. چنانچه این گره از ساختار درختی شامل فرزند باشد این تابع اولین آنها را با فراخوانی متد first _ child ( ) از شئ موردنظر (یعنی شیئی که مرجع آن‌را به‌عنوان آرگومان از ورودی دریافت کرده است) به برنامه فراخواننده بازمی‌گرداند. درصورتی‌که گره موردبحث فاقد هیچ‌گونه فرزندی باشد تابع next _ node ( ) به‌دنبال یافتن گره دیگری که هم سطح گره ورودی به تابع است کار خود را ادامه می‌دهد. چنان‌چه این تابع در فرآیند مذکور موفق باشد گره هم‌سطح بعدی را با فراخوانی متد next _ sibling ( ) از گره ورودی مورد دستیابی قرار داده و در خط ۲۳ از برنامه آن‌را در اختیار برنامه فراخواننده قرار می‌دهد. درصورتی‌که گره ورودی به تابع next _ node ( ) فاقد اولین فرزند (یا به‌عبارت کلی‌تر فاقد فرزند) و همچنین فاقد هرگونه گره هم‌سطح خود باشد تابع مذکور با تشکیل یک ساختار تکرار while در خط ۲۴ سعی می‌کند تا فرآیند را در مورد گره دیگری مجدداً تکرار نماید. ساختار تکرار فوق، ساختاری است که عملیات مشخصی را تا حصول گره ریشه سند XML با ارزیابی‌ای که توسط متد parent ( ) از شئ DomElement صورت می‌پذیرد، انجام می‌دهد. در هر گذر از این ساختار وجود گره هم‌سطح بعدی مورد ارزیابی قرار می‌گیرد (این کار توسط متد next _ sibling ( ) در خط ۲۵ انجام می‌شود) و به محض یافتن چنین گره‌ای از آن گره به‌واسطه عبارت retrurn در خط ۲۶ به برنامه فراخواننده تابع next _ node ( ) ارسال می‌گردد. همان‌گونه که مشاهده می‌کنید با فراخوانی متوالی تابع مذکور که در ساختار تکرار موجود در خط ۱۲ انجام شده است ساختار درختی مربوط به سند XML موردنظر عاقبت پیمایش می‌شود.

رویکرد بعدی نیز مشابه همین روش است چراکه به شیوه‌ای یکسان فرآیند پیمایش درخت سند XML را انجام می‌دهد. با این حال یک تفاوت کوچک در این میان وجود دارد و آن در این است که رویکرد جدید شیوه‌ای را که طی آن برنامه فراخواننده به‌طور مکرر گره بعدی را مورد درخواست قرار می‌دهد، در پیش نمی‌گیرد. در عوض این رویکرد سعی می‌کند تا با بهره‌مندی از یک تکنیک برنامه‌نویسی با عنوان بازگشت یا Recursion ترتیبی دهد تا کد مزبور دائماً خود را مورد فراخوانی قرار دهد. این فرآیند تا بدان جا ادامه می‌یابد که درخت ساختار XML کاملاً مورد پیمایش برنامه قرار گیرد. برنامه موجود در لیست ۶- ۲۲ چگونگی عملکرد این روش پیمایش را در عمل نمایش می‌دهد.

بازگشت یا Recursion روشی است که طی آن یک تابع خود را به‌طور مستقیم یا غیر مستقیم مورد فراخوانی قرار می‌دهد. در روش بازگشت غیر مستقیم تابع اول تابع دیگری را مورد فراخوانی قرارداده و آن تابع مجدداً تابع اول را فرا می‌خواند (این نوع بازگشت به چندین تابع قابل تعمیم است). تکینک فوق علی‌رغم اینکه معمولاً برای برنامه‌نویسان مبتدی به سختی قابل درک است در میان برنامه‌نویسان حرفه‌ای یک ابزار کارآمد و مؤثر محسوب می‌شود. بسیاری از برنامه‌هایی که با استفاده از ساختارهای تکرار قابل پیاده‌سازی نبوده یا به‌سختی پیاده‌سازی می‌شود با استفاده از روش بازگشت به‌راحتی قابل حل و بسط می‌باشند (دو نمونه کلاسیک از این‌گونه مسائل عبارتند از پیداکردن عنصر دلخواهی از دنباله عددی فیبوناچی و مسأله برجهای هانوی). اکثر مسائلی که با بهره‌گیری از ساختارهای تکرار قابل پیاده‌سازی هستند، به روش بازگشتی نیز قابل حل و فصل می‌باشند.

```
1: <?php
2: $doc = xmldocfile( "listing22.1.xml" );
3: $root = $doc->root();
4: traverse( $root );
5:
6: function traverse( $node, $level=0 ){
7:     handle_node( $node, $level );
8:     if ( $node->has_child_nodes() ) {
9:         $children = $node->children();
10:        foreach( $children as $kid ) {
11:            traverse( $kid, $level+1 );
12:        }
13:    }
14: }
15:
16: function handle_node( $node, $level ) {
17:     for ( $x=0; $x<$level; $x++ )
18:         print "  ";
19:     if ( $node->type == XML_ELEMENT_NODE ) {
20:         print $node->tagname()."<br>";
21:     }
22: }
23: ?>
```

لیست ۶- ۲۲ پیمایش درختی از گره‌های XML

چنانکه در این لیست مشاهده می‌کنید تابع ( ) traverse که تعریف آن در خط ۶ از برنامه ارائه شده است کلیه عملیات مربوط به پیمایش درخت XML را در این رویکرد به‌عهده دارد. تابع traverse ( ) تابعی با عملکرد بازگشتی یا خودفراخوان است که شامل دو آرگومان ورودی می‌باشد. آرگومان اول این تابع شیئی از نوع DomElement است که تابع مذکور عملیات را به قصد یافتن آن انجام می‌دهد. آرگومان دوم این تابع یک عدد صحیح است که نشانه سطح می‌باشد. همان‌گونه که ملاحظه می‌کنید

مقدار پیش‌فرض این آرگومان برابر با عددصحیح صفر به نشانه اولین سطح (یا به‌طور دقیق‌تر گره ریشه) انتخاب شده است. چنان‌چه گره ورودی به این تابع که توسط اولین آرگومان مشخص می‌شود شامل گره‌های فرزند باشد، این تابع با بهره‌گیری از متد ( ) children از شئ DomElement کلیه فرزندان آن‌را مورد دستیابی قرار داده و سپس در قالب یک ساختار تکرار به‌ازای هر یک از فرزندان این گره اقدام به خودفراخوانی (recursion) می‌نماید. به تغییر سطح (آرگومان دوم تابع ( ) traverse) در هر فراخوانی توجه کنید. هیچ دلیلی ندارد که تابع بازگشتی از همان مقادیر ورودی قبلی استفاده کند چه در این‌صورت همان نتایج قبلی حاصل خواهند شد. این قاعده‌ای است که آرگومان تابع بازگشتی تا رسیدن به یک نتیجه قطعی (که به حالت پایه یا base case شهرت دارد) و بازگشت از آن دستخوش تغییر شود و این تغییر در مورد تابع بازگشتی مستقیم در درون خود تابع اتفاق خواهد افتاد (مانند تغییر سطحی که در خط ۱۱ در مورد تابع بازگشتی ( ) traverse شاهد آن هستیم). توجه کنید که با هر بار فراخوانی تابع ( ) traverse تابع دیگری با عنوان ( ) handle _ node به‌عنوان بخشی از آن مورد فراخوانی واقع می‌شود (این تابع در خط ۱۶ از برنامه تعریف شده است).

مقدمه‌ای بر XSL

XSL کوتاه شده عبارت Extensible Stylesheet Language است. این زبان درحقیقت مکانیزمی برای الگوسازی اسناد XML است. به‌واسطه وجود یک چنین مکانیزمی به راحتی می‌توان اسناد XML را جهت نمایش در خروجی مورد پردازش قرار داد. در واقع می‌توان الگوهای مختلفی را در قالب XSL به یک سند واحد XML جهت نمایش آن سند در قالب‌بندی‌های مختلف به‌منظور نمایش در پنجره مرورگر اینترنت مورد استفاده قرار داد. امروزه بهره‌گیری از XSL به عرصه‌های جدیدتری از وب، مانند PDA ، تلویزیون محاوره‌ای و تلفنهای همراه نیز توسعه یافته است.

مباحث مربوط به XSL و چگونگی بهره‌گیری از امکانات آن به‌اندازه کافی وسیع است که بررسی‌آن‌رابه‌کتاب دیگری واگذار نماییم، اما با وجود این می‌توانیم تا حدودی به رابطه این زبان آرایشی با زبان برنامه‌نویسی PHP بپردازیم. در این قسمت از درس توابعی از PHP را بررسی می‌کنیم که جهت پشتیبانی از XSL ارائه شده‌اند.

PHP و XSL

مشابه پشتیبانی از مدل پردازش DOM، همچنانکه در قسمت قبل مشاهده کردید، پشتیبانی PHP از زبان آرایشی XSL نیز در مراحل ابتدایی خود می‌باشد. هم توابع مربوط به XSL و هم توابع مربوطه به XSLT (حرف ' T ' در اینجا بیانگر واژه Transformations می‌باشد) در حال حاضر تحت توسعه بوده و احتمال تغییر نام و همچنین تغییر عملکرد توابع مربوطه در PHP بدین ترتیب دور از

ذهن نمی‌باشد. بنابراین توصیه می‌کنیم تا پیش از هرگونه بهره‌برداری از توابع PHP در مورد XSL یا XSLT در پروژه‌های برنامه‌نویسی خود جهت آخرین وضعیت پشتیبانی زبان برنامه‌نویسی PHP از این دو تکنولوژی قالب‌بندی محتویات مطالب موجود در مستندات PHP را در این زمینه مورد مطالعه دقیق و موشکافانه قرار دهید. به‌منظور دستیابی به این مستندات کافی است تا سند موجود در آدرس URL زیر را در فیلد آدرس مرورگر اینترنت خود وارد کنید:

http : // www . php . net / manual / en / ref . xslt . php

همچنین به‌منظور استفاده از توابع XSLT موجود لازم است تا ابتدا یک پردازنده XSLT ویژه با عنوان Sablotron XSL Processor را بر روی کامپیوتر خود نصب نموده و سپس PHP را جهت تحصیل پشتیبانی از XSL مجدداً کامپایل نمایید. توجه کنید که برای انجام این کار هنگامی که برنامه پیکربندی PHP را اجرا می‌کنید لازم است تا گزینه زیر را به‌همراه مقدار آن در این فرآیند مورد استفاده قرار دهید:

- - with - sablot = / path / to / sablotron / libs

با این وجود عبارت زیر یعنی :

- - with - sablot = / usr

به احتمال قوی جهت دستیابی به پشتیبانی مذکور هنگام نصب استاندارد PHP بر روی کامپیوترتان کافی خواهد بود. به‌منظور دستیابی به پردازشگر Sablotron با استفاده از مرورگر اینترنت خود آدرس URL زیر را مورد بازدید قرار دهید:

http : // www . gingerall . com /

بررسی یک سند از نوع XSL

جهت مشاهده نحوه عملکرد اسناد آرایشی XSL در این قسمت یک نمونه واقعی از این‌گونه اسناد را ارائه می‌کنیم. کد موجود در لیست ۷- ۲۲ یک سند آرایشی XSL را به سند XML موجود در لیست ۱- ۲۲ که پیشتر آن‌را مورد بررسی قرار داده‌ایم، اعمال می‌کند. اعمال این سند آرایشی به سند XML مذکور باعث می‌شود تا این سند به شیوه خاصی جهت نمایش در خروجی (بر روی صفحه مرورگر اینترنت) قالب‌بندی شود. این نوع قالب‌بندی باعث خواهد شد تا خروجی در قالب جدولی به‌ازای هر یک از مقالات موجود قالب‌بندی گردد.

```
1: <?xml version="1.0"?>
2: <xsl:stylesheet
3: version="1.0"
4: xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
5:     <xsl:template match="banana-news">
6:         <xsl:for-each select="newsitem">
7:             <p>
8:             <table>
9:              <tr><td>
10:                <i><xsl:value-of select="byline" /></i>
```

```
11:                 <xsl:text> writes</xsl:text>
12:             </td></tr>
13:             <tr><td> <b><xsl:value-of select="headline" /></b> </td></tr>
14:             </table>
15:             </p>
16:         </xsl:for-each>
17:     </xsl:template>
18: </xsl:stylesheet>
```

لیست ۷- ۲۲ یک سند XSL نمونه

بدون اینکه وارد جزئیات مربوط به چگونگی عملکرد این سند نمونه جهت قالب‌بندی اسناد XML بشویم، می‌توانیم عملکرد نهایی این سند را با کمی دقت و تأمل به‌راحتی درک نماییم. پیش از هرچیز ابتدا توجه شما را به اولین خط از این سند XSL که شامل معرفی سند می‌باشد، جلب می‌کنیم. شاید از اینکه متوجه شده‌اید هر سند XSL خود یک سند XML است، شگفت‌زده شده‌اید. عنصر ریشه این سند XSL چنانکه مشاهده می‌کنید، به‌صورت زیر نوشته شده است:

< xsl : stylesheet
version = "1 . 0"
xmlns : xsl = "http : // www . w3 . org / 1999/ XSL / Transform " >

در واقع این مطلب قاعده‌ای کلی محسوب می‌شود، بدین معنی که عنصر ریشه هر سندی از نوع XSL باید به‌ترتیب فوق نوشته شود. در این عنصر ریشه دو مورد فضای نام‌گذاری و شماره نسخه‌ای از XSL (یا به‌عبارت دقیق‌تر، XML) که مورد بهره‌برداری واقع می‌شود، ذکر شده‌اند (فضای نام‌گذاری که معمولاً به شکل آدرس URL نوشته می‌شود جهت جلوگیری از تداخل اسامی نشانه‌های موجود در سند XSL مورد استفاده قرار می‌گیرد. دقت کنید که سند موجود در این آدرس URL هرگز از جانب سند XSL حاضر مورد دستیابی واقع نمی‌شود).

چنانکه در خط ۵ از این کد XSL ملاحظه می‌کنید، نشانه < xsl : template > بخشی از سند XML را مشخص می‌کند که سند XSL فرآیند تبدیل را بر روی آن انجام می‌دهد (دراین مورد بخش فوق کل سند XML است. به‌عبارت دیگر سند XSL قصد دارد تا کل سند XML را دستخوش تبدیل یا به‌عبارت دقیق‌تر قالب‌بندی نماید). به‌واسطه وجود نشانه خاصی در خط ۶ از این سند XSL با عنوان < xsl : for _ each > این امکان برای ما فراهم می‌شود تا یک قالب‌بندی واحد را به هریک از نشانه‌های < newsitem > از سند XML اعمال کنیم (همان‌گونه که مشاهده می‌کنید صفت Select از نشانه <xsl: for _ each> دنباله کاراکتری "newsitem" را مشخص کرده است). با در دست داشتن این حربه کارآمد می‌توانیم به کار خود در مورد قالب‌بندی سند ادامه دهیم. کد HTML تعبیه شده در درون این سند XSL (خطوط ۷ تا ۱۵) نیز به مانند هر سند XML دیگر مشمول قوانین سفت و سخت XML است. این بدان معنی است که نشانه آغازین < p > موجود در خط ۷ باید با نشانه پایانی < p / > همراه باشد (در حالت عادی در یک سند از نوع HTML وجود تنها نشانه آغازین < p > برای تعریف یک پاراگراف یا < br > برای شکست خط جاری و آغاز یک خط جدید کافی است). طی فرآیند قالب‌بندی

سند XML هریک از نشانه‌های < xsl : value _ of > موجود در خطوط ۱۰ و ۱۳ با نشانه‌هایی که به‌عنوان صفتهای مربوطه یعنی < byline > و < headline > مشخص شده‌اند، جایگزین خواهند شد. توجه کنید که موقعیت هر یک از نشانه‌های< byline > و < headline > قابل تطبیق با الگوی قالب‌بندی با یکدیگر تعویض می‌شوند. نکته مهمی که باید در مورد قابلیت اسناد آرایشی XSL به‌خاطر سپرد این است که این گونه اسناد علاوه بر کنترل دقیق ساختار داده‌ها در خروجی قادر به قالب‌بندی خروجی نیز می‌باشند.

فرآیند اعمال یک سند آرایشی XSL به یک سند XML با استفاده از قابلیتهای زبان برنامه‌نویسی PHP

اکنون با در دست داشتن سند آرایشی XSL می‌توانیم سند XML موردنظرمان را با اعمال دستورالعملهای آرایشی موجود در سند XSL به‌گونه‌ای مناسب قالب‌بندی نماییم. در حقیقت به‌منظور انجام این کار به تعداد بسیار محدودی از توابع ارائه در زبان PHP نیاز داریم. برنامه موجود در لیست ۸ - ۲۲ نمونه‌ای از به‌کارگیری این توابع را به‌منظور اعمال یک سند XSL به یک سند از نوع XML جهت قالب‌بندی نشان می‌دهد.

```
1: <?php
2: $xsl_string = join( "", file("listing22.7.xsl") );
3: $xml_string = join( "", file("listing22.1.xml") );
4:
5: if ( xslt_process($xsl_string, $xml_string, $result) )
6:     print $result;
7: else
8:     die( format_xslt_error() );
9:
10: function format_xslt_error() {
11:     $ret = "XSLT ERROR (".xslt_errno().."): ".xslt_error();
12:     return $ret;
13: }
14: ?>
```

لیست ۸- ۲۲ بهره‌گیری از یک سند XSL جهت قالب‌بندی سندی از نوع XML

همان‌گونه که در این لیست مشاهده می‌کنید در خطوط ۲ و ۳ از برنامه هر یک از اسناد XSL و XML را که به‌ترتیب با اسامی listing 22 . 7 . xsl و listing 22 . 1 . xml مشخص شده‌اند با بهره‌گیری از تابع ( ) join از فایل‌های مربوطه بازیابی کرده و در قالب دنباله‌های کاراکتری در دو متغیر با اسامی $xsl_ string و $xml_ string ذخیره کرده‌ایم. سپس در خط ۵ از برنامه، تابعی با نام xslt_ ( ) process را مورد فراخوانی قرار داده‌ایم. تابع ( ) xslt _ process، چنانکه در این لیست مشاهده می‌کنید از ساختار نسبتاً پیچیده‌ای برخوردار است چراکه از سه دنباله کاراکتری به‌عنوان آرگومان‌های ورودی استفاده می‌کند. آرگومان‌های اول و دوم این تابع به ترتیب نماینده اسناد XSL و XML

موردنظر در قالب دنباله‌های کاراکتری هستند. آرگومان سوم متغیری است که نتیجه حاصل از قالب‌بندی، یعنی نتیجه حاصل از اعمال سند آرایشی XSL بر روی سند XML را ذخیره می‌کند (آرگومان اخیر به شیوه ارسال از طریق مرجع به تابع ( ) xslt _ process ارسال می‌گردد). این تابع درصورتی که نتیجه عملیات قالب‌بندی سند XML رضایت‌بخش باشد، مقدار true و در غیر این‌صورت مقدار false را به برنامه فراخواننده باز می‌گرداند.

توجه به این نکته ضروری است که اگر این تابع در عملیات قالب‌بندی سند XML ورودی موفق نباشد، هیچ پیغامی را مبنی بر عدم موفقیت‌آمیز بودن عملیات در خروجی نمایش نداده و تنها مقدار false را باز می‌گرداند. بنابراین برنامه‌نویس در صورت تمایل به نمایش یک چنین پیغامی باید کد موردنظر را خود توسعه دهد. این کار به‌طرز بسیار ساده‌ای همان‌گونه که در خط ۱۱ از برنامه مشاهده می‌کنید با بهره‌گیری از توابع ( ) xslt _ error و ( ) xslt _ errno قابل پیاده‌سازی است. این دو تابع جهت انجام عملیات خود نیازی به آرگومان ورودی ندارند. با این حال برنامه‌نویس می‌تواند درصورت تمایل از یک آرگومان ورودی اختیاری در مورد هر یک از این توابع استفاده نماید. این آرگومان اختیاری در صورت استفاده باید مرجعی به یک پردازشگر XSLT را مشخص نماید. توجه کنید که ( ) xslt_ process به‌گونه‌ای طراحی شده است که حین انجام عملیات خود هیچ‌گونه مرجعی را به یک پردازشگر XSLT تولید نمی کند. بنابراین می‌توان عدم نیاز توابع ( ) xslt _ error و ( ) xslt _ errno به آرگومان ورودی را نوعی خوش شانسی تلقی کرد. درصورتی‌که هنگام فراخوانی این دو تابع از ارسال آرگومان اختیاری خودداری به‌عمل آید تابع ( ) xslt _ error به‌عنوان نتیجه عملیات دنباله‌ای از کاراکترها را به برنامه فراخواننده باز می‌گرداند. این دنباله کاراکتری توصیفی از آخرین خطای XSLT حین دوره حیات برنامه خواهد بود. همچنین تابع ( ) xslt _ errno نیز چنان‌چه بدون آرگومان ورودی فراخوانی شود عدد صحیحی را به‌عنوان نتیجه عملیات به برنامه فراخواننده باز می‌گرداند. این عدد صحیح شماره مرجع مربوط به آخرین خطای XSLT حین دوره حیات برنامه خواهد بود.

جمع‌بندی

XML مبحث بسیار وسیعی برای بحث و گفتگو است و پرداختن به تمام جوانب و ویژگی‌ها و قابلیتهای آن کتابی کاملاً مجزا و تخصصی را در این زمینه طلب می‌کند. وجود کتابهای متعدد و عالی در این زمینه گواهی بر این مطلب است. بنابراین غیر ممکن است بتوان این ویژگی‌ها را تنها در بخشی از یک کتاب تحت پوشش قرار داد. با وجود این حقیقت انکارناپذیر همان‌گونه که در این ساعت متوجه شدید سعی ما بر این بود تا نگاهی گذرا به امکاناتی که این تکنولوژی کارآمد در اختیار برنامه‌نویسان PHP قرار داده است انداخته و چند برنامه نمونه را در این مورد بررسی کنیم.

در درس این ساعت شما چگونگی پردازش اسناد XML را با بهره‌گیری از کتابخانه تدوین شده توسط آقای James Clark با عنوان Expat فراگرفتید. این کتابخانه همان‌گونه که ملاحظه نمودید شامل توابع متعددی جهت پردازش اسناد XML است. ضمن این فرآیند متوجه شدید که این کتابخانه عملکردی بر مبنای رخدادهایی دارد که حین پیمایش اسناد XML به‌وقوع می‌پیوندند. در بخش دیگر این ساعت مبحث مربوط به توابع DOM را مطرح کرده و چگونگی بهره‌گیری از توابع این مدل پردازش را جهت تشکیل یک سند XML شرح دادیم. همان‌گونه که مشاهده کردید مدل DOM اجازه می‌دهد تا علاوه بر تهیه یک ساختار یا مدل درختی از روی یک سند XML موجود، سند جدیدی را نیز برمبنای یک چنین ساختار درختی ایجاد کنیم.

طی بررسی مدل DOM بر این نکته اشاره کردیم که اسناد XML را در ساختار درختی می‌توان با بهره‌گیری از دو رویکرد مختلف مورد پیمایش قرار داد. روش اول مبتنی بر دستیابی به هر یک از گره‌های درخت و ارسال آنها به برنامه اصلی بود. روش دوم تشابه زیادی به روش اول داشت با این تفاوت که به‌جای شیوه تکرار و دستیابی به گره و ارسال آن به برنامه اصلی با بهره‌گیری از یک فراخوانی بازگشتی اقدام به پیمایش درخت می‌کرد. در انتهای درس این ساعت الگوهای XSL را مورد بررسی قرار داده و چگونگی استفاده از توابع XSLT را جهت قالب‌بندی یک سند XML تشریح کردیم.

در درس ساعت آینده مبحثی با عنوان Smarty را عنوان خواهیم کرد. Smarty یک موتور الگوسازی PHP بسیار توانمند و با قابلیت است که جهت توسعه و بهبود ساختار پروژه‌های برنامه نویسی بزرگ طراحی شده است. با دستیابی به این منبع کارآمد توان شما در ایجاد برنامه‌های کاربردی وب به میزان بسیار قابل توجهی افزایش خواهد یافت.

پرسش و پاسخ

پرسش: امروزه به‌نظر می‌رسد که مبحث XML و تکنولوژی‌های مربوطه مانند XSL محافل کامپیوتری را بسیار رونق داده است، آیا صحبت در مورد آن در طیف بسیار وسیعی که شاهد آن هستیم بیشتر جنبه تبلیغاتی ندارد؟

پاسخ: اغلب مردم به بحث و گفتگو در مورد مسائلی که به اصطلاح "مد روز" شده‌اند اهمیت فراوانی داده و از آن لذت می‌برند، اما حقیقت این است که تکنولوژی XML یک ابزار مدرن و بسیار عالی جهت بهره‌برداری مشترک از داده‌ها بوده و امکان کنترل و انجام پروژه‌های بزرگ را به شکلی کاملاً مستحکم و با ثبات و ضمناً قابل توسعه در اختیار برنامه‌نویسان قرارداده است. این حقیقت انکارناپذیر نیز که می‌توان استانداردهای جدیدی را با بهره‌گیری از اسناد DTD تعریف کرد بدین معنی است که ساخت مفسرهایی که با وجود کمی حجم و کوچکی وقت با ارزش برنامه‌نویسان را جهت بررسی وجود اشکالات تلف نمی‌کنند، امکان‌پذیر شده است. چنان‌چه تا به حال اقدام به بارگیری یک

مرورگر اینترنت جهت نصب و استفاده بر روی کامپیوترتان کرده باشید تصدیق خواهید کرد که زمان زیادی را حین فرآیند بارگیری نرم‌افزار مذکور از اینترنت از دست داده‌اید. این امر به‌دلیل حجم بسیار زیاد این مرورگرها و در نتیجه صرف زمان قابل توجه جهت بارگیری آنها می‌باشد. یکی از دلایلی که تکنولوژی XHTML، یا به عبارت دیگر نسخه‌ای از زبان نشانه‌گذاری HTML که بر مبنای قوانین و الگوهای تدوین شده توسط XML توسعه پیدا کرده است تا بدین حد پراهمیت ظاهر شده است به احتمال زیاد این مطلب می‌باشد که در حال حاضر ابزارهای دیجیتالی که در سطح وسیع در حال همگانی شدن می‌باشند؛ مثل تلفنهای همراه، دستگاه‌های PDA و سایر لوازم موجود توانایی کافی جهت پردازش و بهره‌برداری از اسناد HTML موجود بر روی وب را ندارند مگر آنکه این اسناد از استانداردی که برای این ابزارهای دیجیتالی مناسب است، پیروی کنند و پاسخ این مسأله توسعه اسناد جدید در قالب XHTML، یعنی همان استاندارد مورد نظر است. چنان‌چه به این مبحث قابل توجه علاقه‌مند هستید، توصیه می‌کنیم اسناد مربوط به مشخصه‌های این تکنولوژی ارزشمند را در آدرس URL زیر مورد بررسی قرار دهید:

http: // www. w3 . org / TR / xhtml 1 /

پرسش: همان‌گونه که در این درس اظهار شد، توابع مربوط به XSLT و DOM هم‌اکنون مراحل توسعه خود را طی می‌کنند؛ آیا این بدان معنی است که بهره‌گیری از آنها در برنامه‌های PHP مشکل‌آفرین است؟

پاسخ: پیش از اینکه پاسخی به این پرسش داده باشیم توصیه می‌کنیم تا مستندات نسخه‌ای از زبان برنامه‌نویسی PHP را که در حال حاضر مورد استفاده قرار می‌دهید در آدرس URL زیر بازخوانی نمایید:

http : // www . php . net /

احتمال زیادی وجود دارد که زمانی که در حال مطالعه این کتاب هستید توسعه توابع XSLT و DOM به اندازه کافی مراحل بلوغ و رشد خود را طی کرده و کاملاً قابل اعتماد باشند. در حال حاضر توصیه ما این است که اگر در حال انجام یک پروژه برنامه‌نویسی با PHP هستید، به‌ویژه اگر حاصل این پروژه به‌عنوان یک بسته نرم‌افزاری منتشر خواهد شد، از توابع پردازشگر XML مانند توابع کتابخانه Expat نوشته James Clark یا کتابخانه Xerces که توسط پروژه Apache هدایت می‌شود (مراجعه کنید به http : // xml . apache . com /)، استفاده نمایید چراکه این توابع کاملاً از ثبات بالایی برخوردارند. با این وجود اگر قصد شما از پرداختن به این تکنولوژی‌ها صرفاً آموزشی باشد، بهتر است آستینهای خود را بالا زده و به مهارت خود در مورد XSLT و DOM بیندیشید. با این همه فراموش نکنید که اگر تمایل به ایجاد یک ساختار درختی از اشیا، مانند ساختار درختی اشیای مدل DOM داشته باشید با کمی همت و تلاش تقریباً به سادگی قادر خواهید بود تا با بهره‌گیری از توابع پردازشگر XML و اشیایی که به‌دقت تعریف می‌کنید، خودتان اقدام به ایجاد کتابخانه‌ای مشابه DOM نمایید.

تمرینها

هدف از این بخش ارائه تمرینهایی در قالب آزمون و فعالیت برنامه‌نویس است. پاسخ بخش آزمون بلافاصله پس از هر آزمون ارائه شده است. بخش فعالیتها شامل تمرینهایی است که با هدف افزایش مهارت و قابلیت برنامه‌نویسی خواننده طراحی شده و البته فاقد پاسخ است.

آزمون

۱- چگونه می‌توان به یک منبع پردازشگر دسترسی پیدا کرد؟

۲- بهره‌گیری از تابع کنترل‌کننده نشانه آغازین از توابع پردازشگر XML، مستلزم تأمین چه آرگومان‌هایی است؟

۳- چنانکه ملاحظه کردید، توابع کنترل‌کننده XML پیش از پردازش نشانه‌ها کلیه حروف کوچک را به حروف بزرگ تبدیل می‌کنند. چگونه می‌توان این ویژگی را غیر فعال نمود؟

۴- چگونه می‌توان حین پردازش یک سند XML نمونه توسط توابع پردازشگر XML به شماره خط جاری (خط در حال پردازش) دسترسی پیدا کرد؟

۵- از کدام تابع می‌توان جهت دستیابی به شیئ از نوع DomElement از یک سند XML در حال پردازش استفاده کرد؟

۶- با فرض در دسترس بودن شیئ از نوع DomElement، چگونه می‌توان شئ فرزند را به شئ مذکور در ساختار درختی سند XML اضافه نمود؟

۷- از کدام تابع می‌توان جهت اعمال یک سند آرایشی XSL به یک سند XML بهره‌گرفت؟

پاسخ آزمون

۱- با بهره‌گیری از تابعی با عنوان ( ) xml _ parser _ create می‌توان به‌طریقی که در عبارت نمونه زیر مشاهده می‌کنید یک منبع پردازشگر XML را جهت استفاده در توابع مختلف مورد دستیابی قرار داد:

```
$parser = xml _ parser _ create ( ) ;
```

۲- تابع کنترل‌کننده نشانه آغازین یک سند XML که با عنوان ( ) start _ handler مشخص می‌شود، مستلزم دریافت سه آرگومان ورودی می‌باشد. اولین آرگومان یک منبع پردازشگر XML را مشخص می‌کند. دومین آرگومان نام نشانه مورد پردازش در قالب یک دنباله کاراکتری است و سومین آرگومان نیز آرایه‌ای شامل صفتهای نشانه آغازین مورد پردازش است.

۳- با بهره‌گیری از تابعی با نام ( ) xml _ parser _ set _ option که گزینه‌های مربوط به کیفیت پردازش اسناد XML را کنترل می‌کند، می‌توان به‌صورت زیر این ویژگی را غیر فعال نمود:
xml _ parser _ set _ option ($parser , XML _ OPTION _ CASE _ FOLDING , 0) ;
۴- با بهره‌گیری از تابع ( ) xml _ get _ current _ line _ number می‌توان به شماره خط جاری از یک سند در حال پردازش از نوع XML دسترسی پیدا کرد.
۵- تابع ( ) xmldocfile شئ از نوع DomElement را به دست می‌دهد:
$doc = xmldocfile ("my _ doc . xml ") ;
۶- متد ( ) new _ child امکان افزودن یک نشانه جدید را به یک ساختار درختی فراهم می‌کند:
$child = $el → new _ child ("element" , "element text goes here") ;
۷- با بهره‌گیری از تابع ( ) xslt _ process، می‌توان سندی آرایشی از نوع XSL را به یک سند XML اعمال کرد. تابع مذکور به‌عنوان آرگومان‌های اول و دوم، اسناد XSL و XML را در قالب دنباله‌ای از کاراکترها و به‌عنوان آرگومان سوم یک متغیر جهت نگهداری سند قالب‌بندی شده نهایی دریافت می‌کند:
xslt _ process ($xsl _ string , $xml _ string , $result) ;

فعالیتها

۱- برنامه‌ای بنویسید که با بهره‌گیری از یک منبع خبری در آدرس . http: // slashdot org / slashdot . xml اخبار روزانه را در قالب یک سند HTML جهت مطالعه بازدیدکنندگان نمایش دهد. برنامه دیگری بنویسید که خروجی را در قالب متنی خوش‌طرح نمایش دهد.

۲- با بهره‌گیری از توابع پردازشگر XML و تکنیکهای برنامه‌نویسی شیئ‌گرا، مدلی مشابه مدل DOM جهت تولید ساختار درختی یک سند XML توسعه دهید. درخت نمونه‌ای را که براین پایه تشکیل شده است، پیمایش نمایید.

« بخش چهارم: توسعه PHP »

- ساعت بیست و سوم: موتور الگوسازی Smarty
- ساعت بیست و چهارم: بررسی یک مثال جامع؛

page. inc. php

ساعت بیست و سوم

موتور الگوسازی Smarty

در سراسر کتاب حاضر تا بدین جا ویژگی‌ها و قابلیتهای زبان برنامه‌نویسی PHP را در رابطه ایجاد برنامه‌های کاربردی معمولی و برنامه‌های کاربردی وب مورد بررسی قرار دادیم. در درس این ساعت قصد داریم تا به بررسی کتابخانه‌ای از توابع بپردازیم که خود به زبان برنامه‌نویسی PHP توسعه یافته است. موتور الگوسازی Smarty، یعنی کتابخانه موردنظر یک ابزار بسیار توانمند جهت ایجاد و سازماندهی برنامه‌ها و کنترل پروژه‌های بزرگ محسوب می‌شود. با بررسی و به‌کارگیری این ابزار کارآمد در انتهای این درس تصدیق خواهید کرد که کتابخانه‌ها تا چه اندازه در توسعه زبانهای برنامه‌نویسی اهمیت دارند.

در این درس به بررسی موارد زیر خواهیم پرداخت:

- **چگونگی نصب موتور الگوسازی Smarty**
- **چگونگی ارسال متغیرهای برنامه به الگوهای Smarty**
- **چگونگی اعطای قابلیت تصمیم‌گیری به Smarty در مورد محتوای نمایشی**
- **چگونگی استفاده از ساختار تکرار در الگوهای Smarty**
- **چگونگی دستکاری متغیرها در الگوهای Smarty**

در ادامه به بررسی هر یک از این موارد می‌پردازیم:

ماهیت Smarty

Smarty یک موتور (ابزار) الگوسازی و سیستمی است که به ما کمک می‌کند تا خروجی نمایشی حاصل از اجرای برنامه را (که معمولاً با اصطلاح Presentation می‌شناسیم) از منطق یا الگوریتمی که برنامه بر مبنای آن عمل می‌کند (و معمولاً به آن Application Logic گفته می‌شود)، جدا کنیم. این جداسازی به‌ویژه در مورد پروژه‌های برنامه‌نویسی مقیاس بزرگ اهمیت مضاعفی دارد، چراکه در این‌گونه پروژه‌ها طراحان و مهندسین در دو گروه مجزا تقسیم‌بندی شده و تمایل بر این است که کار هر گروه کاملاً از دیگری تفکیک شود.

با استفاده از یک سیستم الگومند که توسط برنامه‌نویسی یا به‌واسطه بهره‌گیری از یک نرم‌افزار آماده به کار تهیه شده باشد، به‌راحتی و با توانمندی بسیار می‌توان قابلیت انعطاف بالایی را به پروژه موردنظر اعطا نمود. در پروژه‌های برنامه‌نویسی که منطق (الگوریتم) برنامه کاربردی در قالب عناصر طراحی تعبیه شده باشند، اشیای موجود در برنامه را به‌راحتی نمی‌توان مدیریت و اشکال‌زدایی کرد. در مقابل چنان‌چه این دو بخش تفکیک از یکدیگر جدا شوند، برنامه‌نویسان به‌راحتی و با آسودگی خیال قادر خواهند بود تا به امر کدنویسی پرداخته و بر روی قابلیت و توسعه آن متمرکز شوند. به‌همین ترتیب طراحان HTML نیز می‌توانند با خیال آسوده‌تری فرآیند طراحی را با کارایی بهتر و بدون درگیری با حجم زیادی از کد PHP انجام دهند.

در همین راستا و با داشتن این ایده در ذهن، موتور الگوسازی Smarty توسط دو تن با نامهای Monte Ohrt و Andrei Zmievski جهت حل این مشکل ابداع شد. Smarty غنی از ویژگی‌ها و قابلیت‌های چشمگیر است آن‌چنان غنی که می‌توان گفت موارد بررسی شده در این ساعت تنها نقطه آغازی برای کار با این موتور الگوسازی کارآمد است. جالب آنکه Smarty خود به زبان برنامه‌نویسی PHP توسعه پیدا کرده و اکنون درصدد توسعه قابلیتهای این زبان در عرصه برنامه‌نویسی وب برآمده است. Smarty فاقد بسیاری از نقاط ضعیف سایر موتورهای الگوسازی می‌باشد.

موتورهای الگوسازی عملکرد جالبی دارند. این‌گونه ابزارها قادرند تا واژه‌های کلیدی مشخصی را در درون یک فایل الگو یافته و آنها را با مقادیر تولید شده توسط برنامه اسکریپت جایگزین نمایند. برای انجام یک چنین کاری لازم است تا فایل الگوی موردنظر توسط موتور الگوسازی مورد پردازش قرار بگیرد. در مورد اسناد بزرگ‌تر فرآیند مورد بحث ما یعنی پردازش فایل الگو و جایگزین کردن واژه‌های کلیدی قابل تشخیص توسط موتور الگوسازی با مقادیر تولید شده توسط برنامه اسکریپت به‌طور محسوسی کندتر بوده و البته این وضعیت به منابع موجود در دسترس این موتور الگوسازی بستگی خواهد داشت. Smarty این مشکل را به‌طریقی مبتکرانه و با ترجمه الگوها به کد PHP حل کرده است.

دستیابی و نصب Smarty بر روی کامپیوتر

دستیابی به Smarty بسیار ساده است. کافی است تا آدرس URL زیر را برای این کار مورد بازیابی قرار دهید:

http : // www. Phpinsider . com / php / code / smarty / download

آخرین نسخه Smarty در زمان تالیف کتاب حاضر نسخه 5 . 4 . 1 می‌باشد.

جهت نصب این ابزار ابتدا باید فایل مربوطه را از حالت فشردگی درآورید (در دنیای UNIX معمولاً فرآیند فشرده‌سازی فایل‌ها با استفاده از برنامه‌هایی چون gzip و bzip2 انجام می‌شود. جهت انتشار برنامه‌های تحت UNIX معمولاً ابتدا با استفاده از یک برنامه متداول با عنوان tar فایل‌های موردنظر را در قالب یک فایل واحد با همین پسوند بایگانی کرده و سپس فایل حاصل را با بهره‌گیری از برنامه‌های فشرده‌ساز آماده انتشار می‌کنند. جهت استفاده از یک چنین فایل فشرده‌ای دو فرآیند فوق باید به‌ترتیب عکس انجام شود) :

tar - xvzf Smarty - 1 . 4 . 5 . tar . gz

با نگاهی به حاصل این عملیات می‌توان فایل‌های php تشکیل‌دهنده این موتور الگوسازی را به‌صورت زیر تشخیص داد:

Config _ File . class . php
Smarty . addons . php
Smarty . class . php
Smarty _ compiler . class . php

به‌منظور استفاده از این فایل‌ها لازم است تا آنها را به فهرستی از سیستمتان که شامل فایل‌های کتابخانه‌ای است، منتقل کنید. بدین ترتیب این فایل‌ها به‌راحتی در اختیار برنامه‌های اسکریپت قرار خواهند گرفت. حتی می‌توانید فهرستی را به‌طور اختصاصی برای این فایل‌ها درنظر بگیرید اما باید توجه داشته باشید که پردازشگر PHP را از موقعیت مذکور مطلع نمایید. برای مثال از فهرستی با عنوان Smartylib جهت نگهداری این فایل‌ها استفاده می‌کنیم. فهرست نامبرده خود در درون فهرست www واقع است. برای اینکه پردازشگر PHP را از موقعیت این فایل‌ها آگاه کنیم، می‌توانیم گزینه include _ path از فایل تنظیمات php . ini را به‌صورتی که در زیر مشاهده می‌کنید، تنظیم نمایید:

include _ path = " . : / usr / local / lib / php : / www / smartylib "

(توجه داشته باشید که مسیرهای مختلف در سیستم عامل UNIX با علامت کولون یا : از یکدیگر جدا می‌شوند. این فرآیند در سیستمهای Windows با استفاده از علامت سمی‌کولون یا ; انجام می‌شود).

دقت کنید که به‌عنوان مقدار گزینه include _ path می‌توانیم چندین مسیر مختلف را از سیستم فایل موجود مشخص نماییم. در این صورت باید این مسیرها را با علامت : از یکدیگر جداکنیم. درصورتی‌که به‌دلایلی دسترسی شما به فایل تنظیمات php . ini محدود شده باشد، باز هم می‌توانید

مسیر فایل‌های Smarty را در اختیار پردازشگر PHP قرار دهید. برای این‌منظور می‌توانید فایل htaccess. از وب سرور Apache را جهت تنظیم گزینه فوق یعنی include _ path دستخوش تغییر نمایید:

Php _ value include _ path = " . : / usr / local / lib / php : / www / smartylib "

اما شاید مطمئن‌ترین روش برای انجام این کار از طریق خود برنامه اسکریپت باشد. برای این منظور کافی است تابعی با عنوان ini _ set ( ) را به صورت زیر فراخوانی نمایید:

ini _ set ("include _ path " . : / usr / local / lib / php : / www / smartylib ") ;

علاوه بر تعیین مسیر موقعیت فایل‌های Smarty در سیستم فایل کامپیوتر لازم است تا PHP از موقعیت کد ویژه‌ای که در قالب یک فایل بخصوص با عنوان PEAR . php نگهداری می‌شود، آگاه باشد. این کد با عنوان PEAR مشخص شده و کوتاه شده عبارت PHP Extension and Application Repository است. این کد ویژه جهت عملکرد کدهای کتابخانه‌ای Smarty ضروری است. تمام کاری که در این مورد باید انجام دهیم این است که موقعیت کلاسی با عنوان PEAR . php را مشخص کنیم. این موقعیت در سیستم عامل UNIX به‌صورت زیر مشخص می‌شود:

/ usr / local / lib / php

اما در سیستم عامل Windows مسیر موقعیت موردنظر چنین مشخص می‌شود:

\ php \ pear

چنان‌چه دستورالعملهای فوق را به‌دقت دنبال کرده باشید، اکنون باید کتابخانه Smarty بر روی سیستم شما نصب شده باشد. تنها موردی که جهت استفاده از قابلیتهای این موتور الگوسازی باقی می‌ماند تنظیم فهرستهایی جهت نگهداری الگوهای موردنظر است. برای این منظور لازم است تا در درون فهرستی از سیستم‌عامل که پروژه‌های برنامه‌نویسی PHP را در آنجا نگهداری می‌کنید اقدام به ایجاد سه فهرست با اسامی configs، templates و templates _ c نمایید. واضح است که ما الگوهای موردنیازمان را در فهرست templates نگهداری خواهیم کرد. حاصل فرآیند ترجمه (کامپایل) این الگوها به‌طور خودکار در فهرست templates _ C نگهداری خواهد شد. از این‌رو باید اطمینان حاصل کنید که مجوز نوشتن در فهرست مزبور را در اختیار دارید (سه مجوز موجود در سیستم عامل UNIX عبارتند از مجوز خواندن، نوشتن و اجرا کردن که به ترتیب با حروف r برای read، w برای write و x برای execute مشخص می‌شوند. این مجوزها در مورد فایل‌ها معنی دارند و جالب است بدانید که در سیستم عامل UNIX فهرستها نیز تحت عنوان فایل دسته‌بندی می‌شوند). برای این منظور می‌توانید مالکیت فهرست موردنظر، یعنی templates_ c را به‌کاربری که سرور تحت نام وی اجرا می‌شود (معمولاً nobody)، تغییر دهید. نحوه انجام این کار به‌صورتی است که در اینجا مشاهده می‌کنید:

Chown nobody : nobody templates _ c

درصورتی که از وب سرور Apache جهت سرویس دهی استفاده می‌کنید، می‌توانید نام کاربری که برنامه httpd (برنامه راه‌انداز وب سرور Apache که در قالب یک daemon اجرا می‌شود) را اجرا

می‌کند از طریق دستیابی به فایل تنظیمات این برنامه با عنوان http . conf تشخیص دهید. برای این کار نگاهی به گزینه‌های User و Group از این فایل تنظیمات بیندازید. درصورتی که از مجوز کافی برای تغییر مالکیت فهرست templates _ c برخوردار نیستید، می‌توانید ترتیبی دهید که فهرست مذکور توسط هر کسی که به سیستم دسترسی دارد، قابل نوشتن باشد. اما معمولاً این روش توصیه نمی‌شود به‌هرحال جهت انجام این ریسک می‌توانید فرمان زیر را جهت اجرا به سیستم خود بدهید:

Chown 777 templates _ c

با وجودی که طی بررسی خود در مورد Smarty صحبتی درباره قابلیت ویژه‌ای تحت عنوان Smarty caching به میان نخواهیم آورد اما در همین رابطه به‌منظور بهره‌برداری از این قابلیت لازم است تا فهرستی با عنوان cache ایجاد نمایید. تمام شرایط این فهرست از نظر اعطای مجوز مشابه فهرست template _ c می‌باشد.

هم‌اینک باید ابزار Smarty به‌درستی بر روی کامپیوترتان نصب شده و آماده به‌کار باشد. درصورتی که اجرای قطعه برنامه‌های موجود در این درس منجر به بروز خطا می‌شود احتمالاً موردی را به‌درستی‌انجام نداده‌اید. لذا باید به عقب برگشته و دستورالعملهای نصب و پیکربندی Smarty را مجدداً با دقت بیشتری انجام دهید. توجه به این نکته نیز قابل اهمیت است که ابزار مورد بحث به‌همراه یک دستورالعمل نصب گام به گام منتشر می‌شود.

اولین تجربه با Smarty

در این قسمت اولین برخورد خود با Smarty را تجربه خواهیم کرد. پیش از هر چیز اجازه دهید تا الگویی را در قالب یک سند ایجاد نماییم. بنا به پیش‌فرض و مطابق آنچه که در بین برنامه‌نویسان مرسوم است فایل حاوی این الگو را با پسوند tpl . ذخیره خواهیم کرد. همچنین چنانکه پیشتر نیز اظهار کردیم تمام الگوها در فهرستی با عنوان templates ذخیره می‌شوند. کد موجود در لیست ۱- ۲۳ اولین الگوی ما را نشان می‌دهد.

```
1: <html>
2: <head>
3: <title>{$page_title}</title>
4: </head>
5: <body>
6: <h3>{$page_title}</h3>
7: <p>
8: <b>{$page_subhead}</b>
9: </p>
10: {$page_text}
11: </body>
12: </html>
```

لیست ۱- ۲۳ اولین الگوی Smarty

کد موجود در لیست فوق به احتمال زیاد به طور غریبی مأنوس است. این لیست همان‌گونه که ملاحظه می‌کنید ترکیبی از کد HTML و چند متغیر PHP است. با این حال با نگاهی دقیق‌تر به این لیست، نکات جدیدی خود را آشکار خواهند کرد. این نکات شاید شما را تا حدودی به تعجب وادارند. همان‌گونه که تا بدین‌جای کتاب دریافتید، یک سند PHP ترکیبی از کد HTML و دستورالعملهای زبان PHP است. اما چنانکه در این لیست مشاهده می‌کنید هیچ اثری از علایم آغازین و پایانی کد PHP یعنی علایم php ? > و < ? به چشم نمی‌خورد. ضمن اینکه کلیه متغیرهای PHP در این لیست توسط جفت علامت { } محصور شده‌اند. جفت علامت { } شیوه‌ای است که Smarty از آن جهت تعیین موقعیت جای‌گذاری مقادیر واقعی متغیرها یا کدهای قابل اجرا استفاده می‌کند.

با این همه Smarty به اندازه کافی منعطف بوده و به شما اجازه می‌دهد تا از علامت دیگری جهت تعیین این موقعیتها در دروں یک الگوی نمونه استفاده نمایید. در این درس ترجیح می‌دهیم تا از علامت پیش‌فرض { } برای این منظور استفاده کنیم.

جهت بهره‌برداری از الگوی ساخته شده اکنون به یک برنامه PHP احتیاج داریم تا از قابلیتهای Smarty برای جای‌گذاری مقادیر متغیرها در الگوی مذکور استفاده نماید. اجازه دهید تا ابتدا کد مربوط به این برنامه را ارائه داده و سپس جزئیات آن‌را مورد بررسی قرار دهیم. لیست ۲- ۲۳ این برنامه PHP را نشان می‌دهد.

```
1: <?php
2: require_once("Smarty.class.php");
3:
4: $page_vals = array(
5:          "page_title" => "Listings 23.1 and 23.2",
6:          "page_subhead"=>"Separating script logic from formatting",
7:          "page_text" => "The look and feel of this data is handled by
➥listing23.1.tpl"
8:          );
9:
10: $templ = new Smarty();
11: $templ->assign( $page_vals );
12: $templ->display("listing23.1.tpl");
13:
14: ?>
```

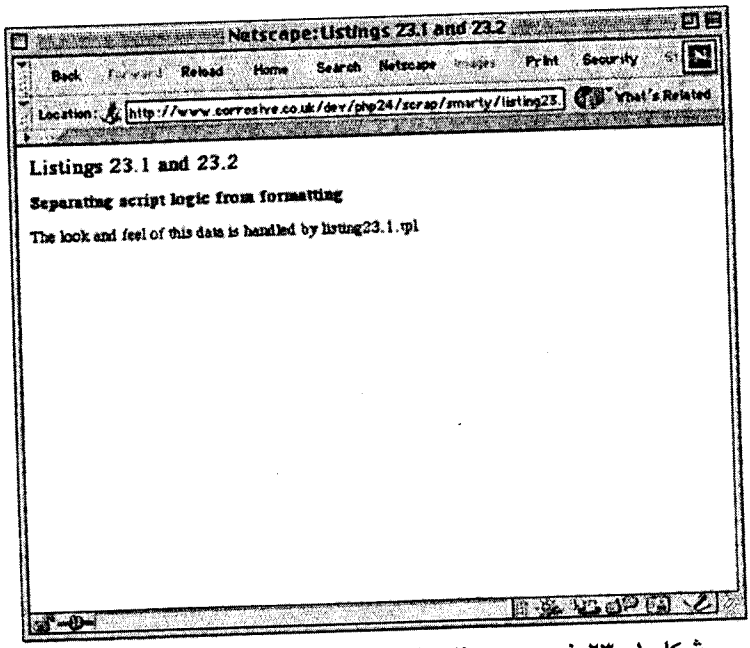
لیست ۲- ۲۳ اولین برنامه Smarty

همان‌گونه که در این برنامه مشاهده می‌کنید، پیش از انجام هر کاری در خط ۲ تابعی با عنوان require _ once ( ) فراخوانی شده است. این تابع از آن جهت مورد فراخوانی قرار گرفته است تا فایل کتابخانه‌ای smarty . class . php را به‌منظور برخورداری از قابلیتهای Smarty در اختیار برنامه قرار دهد. چنان‌چه فراخوانی تابع require _ once ( ) منجر به تولید خطا شود، لازم است تا کمی به عقب بازگشته و اطلاعات موجود در گزینه تعیین مسیر از فایل تنظیمات php . ini یا htaccess . را مورد

بررسی مجدد قرار دهید. باید اطمینان حاصل کنید که فهرستی که شامل فایل کتابخانه‌ای Smarty . class . php است در این گزینه، یعنی گزینه include _ path مشخص شده باشد.

در ادامه برنامه، چنانکه در خط ۴ مشاهده می‌شود یک آرایه انجمنی با نام $page _ vals ایجاد شده است. این آرایه جهت نگهداری کلیه متغیرهای موجود در سند اصلی (که به‌زودی آن‌را آرایه خواهیم کرد) تشکیل شده است. توجه کنید که مقادیر کلیدهای دستیابی این آرایه انجمنی با اسامی متغیرهای موجود در سند الگو، یعنی لیست ۱- ۲۳ کاملاً مطابقت می‌کند. در خط ۱۰ از این برنامه با بهره‌گیری از تابع ( ) Smarty اقدامی جهت ایجاد شیئ از نوع کلاس Smarty اقدامی جهت ایجاد یک شئ از نوع کلاس Smarty (یا به‌عبارت دیگر یک موتور الگوساز Smarty) صورت گرفته است. حاصل فراخوانی تابع سازنده ( ) Smarty، یعنی مرجع شئ موردنظر در متغیری با نام $templ ذخیره شده است. متد ( ) assign از یک شئ که در خط ۱۱ برنامه مورد فراخوانی واقع شده است، آرگومانی را به‌عنوان تنها ورودی ضروری از برنامه دریافت می‌کند. این آرگومان می‌تواند یک آرایه انجمنی بود. (همان‌گونه که در مثال مورد بررسی ما چنین است) و یا اینکه زوجی شامل نام متغیر و مقدار متناظر آن باشد. پس از فراخوانی این تابع جهت همراه کردن داده‌های برنامه با شئ Smarty می‌توانیم به کار خود ادامه داده و خروجی موردنظرمان را چنانکه در خط ۱۲ از این برنامه مشاهده می‌کنید، تولید نماییم. متد ( ) display از شئ Smarty عهده‌دار تولید چنین خروجی است. این متد در کمال سادگی مسیر فایل الگوی موردنظر (listing 23 . 1 . tpl) را به‌عنوان تنها آرگومان ورودی دریافت می‌کند. توجه کنید که در این فراخوانی هیچ نیازی به تعیین موقعیت فهرست templates نیست، چراکه Smarty خود می‌داند که کلیه الگوهای موردنظر در آن فهرست قرار دارند. آنچه که این متد بدان نیاز دارد مسیردهی در درون این فهرست است. مبنای این مسیردهی نقطه ورود به این فهرست است.

خروجی حاصل از برنامه لیست ۲- ۲۳ به‌همراه الگوی موجود در لیست ۱- ۲۳ را در شکل ۱- ۲۳ مشاهده می‌کنید.



شکل ۱- ۲۳ خروجی برنامه لیست ۲- ۲۳ و الگوی لیست ۱- ۲۳

اولین مرتبه‌ای که برنامه موجود در لیست ۲- ۲۳ به اجرا در می‌آید، Smarty الگوی موجود در فایل listing 23 . 1 . tpl را مورد بازخوانی قرار داده و در صورت نیاز جایگزینی‌های لازم (جایگزینی متغیرها با مقادیر متناظر) را به‌دقت انجام می‌دهد. با این حال Smarty مستقیماً اقدام به جای‌گذاری مقادیر متغیرها نمی‌کند، بدین معنی که ابتدا یک نسخه ترجمه شده از الگوی مورد بررسی را به کد PHP متناظر تبدیل می‌کند. بدین ترتیب Smarty قدم مثبتی در بهبود کارآیی بر می‌دارد؛ چراکه در فراخوانی‌های بعدی برنامه موجود در لیست ۲- ۲۳ سربار ناشی از ترجمه الگوی موجود در فایل listing23 . 1 . tpl از میان می‌رود. در حال عادی نیازی نیست تا از نحوه این ترجمه و محتوای فایل ترجمه شده که در فهرست template_ C جای می‌گیرد، مطلع باشید. اما به‌دلیل تکمیل بررسی خود در مورد فرآیند مورد بحث اجازه دهید تا نگاهی کوتاه به حاصل فرآیند ترجمه الگوی لیست ۱- ۲۳ توسط Smarty بیندازیم. ما این کد ترجمه شده را در لیست ۳- ۲۳ قرار داده‌ایم.

```
1: <?php /* Smarty version 1.4.5, created on 2001-10-12 16:21:01
 2:          compiled from listing23.1.tpl */ ?>
 3: <html>
 4: <head>
 5: <title><?php echo $this->_tpl_vars['page_title']; ?>
 6: </title>
 7: </head>
 8: <h3><?php echo $this->_tpl_vars['page_title']; ?>
 9: </h3>
10: <p>
11: <b><?php echo $this->_tpl_vars['page_subhead']; ?>
```

```
12: </b>
13: </p>
14: <?php echo $this->_tpl_vars['page_text']; ?>
15:
16: </html>
```

لیست ۳- ۲۳ نسخه ترجمه شده‌ای از الگوی لیست ۱- ۲۳

به‌عنوان توضیحی کوتاه Smarty آرایه‌ای انجمنی با عنوان tpl _ vars _ را جهت استفاده مهیا می‌کند که کلیدهای دستیابی آن متغیرهای موردنظر می‌باشند.

متغیرهای الگو

به‌دلیل قابلیتی که روش استفاده از آرایه انجمنی جهت نگهداری مقادیر موردنظر از طریق فراخوانی متد ( ) assign از Smarty در اختیار برنامه‌نویس قرار می‌دهد، در مثال قبل ترجیح دادیم تا از این تکنیک استفاده کنیم. با این حال هیچ اجباری در رابطه با بهره‌گیری از آرایه انجمنی در متد ( ) assign وجود نداشته و درصورت تمایل می‌توان اسامی متغیرها و مقادیر متناظر با هر یک از آنها را درقالب‌یک‌جفت‌آرگومان ورودی به متد مورد بحث ارسال نمود. برنامه‌ای‌را که در لیست ۴- ۲۳ مشاهده می‌کنید بازنویسی مجددی از برنامه لیست ۲- ۲۳ است، با این تفاوت که به‌جای یک آرایه انجمنی از متغیرهای منفرد جهت نگهداری مقادیر استفاده می‌کند.

```
1: <?php
2: require_once("Smarty.class.php");
3:
4: $templ = new Smarty();
5: $templ->assign( "page_title", "Listings 23.1 and 23.4");
6: $templ->assign( "page_subhead", "Separating script logic from formatting");
7: $templ->assign( "page_text", "The look and feel of this data is handled by
➥listing23.1.tpl");
8: $templ->display("listing23.1.tpl");
9:
10: ?>
```

لیست ۴- ۲۳ ارسال متغیرهای مجزا به متد ( ) assign

خروجی حاصل از اجرای برنامه موجود در لیست فوق دقیقاً مشابه خروجی است که در رابطه با اجرای برنامه لیست ۲- ۲۳ مشاهده نمودید. در این برنامه به‌جای ارسال یک آرایه انجمنی به متد ( ) assign از Smarty ترجیح داده‌ایم تا متد مذکور را سه مرتبه پشت سر هم فراخوانی کرده و در هر بار فراخوانی دو آرگومان مجزا شامل نام متغیر الگوی موردنظر و مقدار متناظر با آن‌را به این متد ارسال نماییم. چنان‌چه تعداد دفعات فراخوانی برابر با تعداد متغیرهای موجود باشد، تأثیر هر دو روش یکسان خواهد بود.

روش دیگری نیز جهت ارسال یک آرایه انجمنی به تابع ( ) assign موجود می‌باشد:

$templ → assign ("page _ vals " , $page _ vals) ;

همان‌گونه که در عبارت فوق مشاهده می‌کنید مشابه آنچه که در مورد لیست ۲- ۲۳ شاهد آن بودید متغیر $page _ vals را به متد سازنده ( ) assign ارسال کرده‌ایم، اما این مرتبه متغیر نامبرده، یعنی $page _ vals را به‌عنوان آرگومان دوم مورد استفاده قرار داده‌ایم. همان‌گونه که مشاهده می‌کنید به‌عنوان اولین آرگومان متد ( ) assign از یک متغیر الگوی خاص بهره گرفته‌ایم. بهره‌گیری از این متغیر بدان معناست که فایل حاوی الگوی موردنظر هم‌اکنون قادر است تا به واسطه شیوه‌ای که در ادامه ملاحظه می‌کنید، به تک تک عناصر آرایه $page _ vals دسترسی داشته باشد:

```
{ $page _ vals . page _ title }
{ $page _ vals . page _ subhead }
{ $page _ vals . page _ text }
```

علاوه بر این می‌توانیم از مزایای ارسال یک شئ به متد ( ) assign نیز بهره‌مند شویم. قطعه کد زیر چگونگی ارسال شیئ از نوع page _ vals را به متد مذکور نشان می‌دهد:

```
Class page _ vals {
    var $page _ title = "passing an object to assign ( )" ;
    var $page _ subhead = "Separating script logic from formatting" ;
    var $page _ text = "objects use the '→' notation in templates " ;
}
$templ → assign ( $page _ vals , new page _ vals ( ) ) ;
```

با این اقدام می‌توان متغیرهای موردنظر را به‌عنوان خصوصیات شئ $page _ vals با بهره‌گیری از عملگر → مورد دستیابی قرار داد. به‌نحوه انجام این کار توجه نمایید:

```
{ $page _ vals → page _ title }
{ $page _ vals → page _ subhead }
{ $page _ vals → page _ text }
```

علاوه بر ارسال متغیرهای موردنظرتان از برنامه PHP به یک سند الگو با بهره‌گیری از متغیر {$smarty}، می‌توانید برخی از آرایه‌های سیستمی PHP را نیز مورد دستیابی و استفاده قرار دهید. جدول ۱- ۲۳ عناوین این آرایه‌های سیستمی را به‌همراه متغیر {$smarty} متناظر و همچنین یک مثال از نحوه استفاده از هر یک از این متغیرها را نشان می‌دهد. برنامه موجود در لیست ۵- ۲۳ نیز سند الگویی را نشان می‌دهد که از ویژگی فوق بهره‌گیری نموده است.

جدول ۱- ۲۳ متغیرهای {$smarty} متناظر با آرایه‌های سیستمی

نام آرایه سیستمی	نام متغیر {$smarty}	مثال کاربردی
$ENV	env	{$smarty . env. LANGUAGE }
$SERVER	Server	{$smarty. Server. REMOTE_ ADDR}
$HTTP_ GET_ VARS	get	{$smarty. get. Username }

{$smarty. post. Username}	post	$HHTP_ POST_ VARS
{$smarty. session. firstname}	session	$HTTP_ SESSION_ VARS
{$smarty. cookie. last_ visit}	cookie	$HTTP_ COOKI_ VARS

```
 1: <html>
 2: <head>
 3: <title>Listing 23.5 using the special {$smarty} variable</title>
 4: </head>
 5: <body>
 6: Hello user at <b>{$smarty.server.REMOTE_ADDR}</b><br>
 7: You submitted this value: <b>{$smarty.get.name}</b><br>
 8: I am set up to handle <b>{$smarty.env.LANGUAGE}</b><br>
 9: <body>
10: </html>
```

لیست ۵- ۲۳ استفاده از متغیرهای {$smarty} در یک الگوی نمونه

متغیر {$smarty} شباهت زیادی به یک آرایه انجمنی چند بعدی دارد. تنها تفاوتی که بین این دو موجودیت می‌توان تشخیص داد این است که جهت دستیابی به عناصر مختلف آن باید از عملگر نقطه‌استفاده کرد (ستون دوم از جدول فوق را ببینید). حال آنکه این عمل در مورد آرایه‌های چندبعدی با استفاده از جفت علامت [ ] انجام می‌شود.

توابع Smarty

علی‌رغم تمام تلاشی که در جداسازی کد PHP از سند الگو شده به‌گونه‌ای که معمولاً تمامی منطق برنامه در قالب سند PHP شکل می‌گیرد، گاهی از اوقات مواردی پیش‌می‌آید که لازم است تا سند الگوی مذکور تصمیماتی را بر مبنای خروجی برنامه PHP اتخاذ نماید. به این دلیل طراحان Smarty سعی کرده‌اند تا طیف گوناگونی از توابع را در این ابزار کارآمد تعبیه نمایند. بحث این قسمت از درس این ساعت به بررسی همین توابع اختصاص دارد.

توابع نیز تقریباً به همان شیوه‌ای که در مورد متغیرها شاهد بودیم از محتوای ایستای اسناد تفکیک می‌شوند، بدین معنی که نام آنها به‌همراه آرگومان‌های دریافتی این توابع با بهره‌گیری از جفت علامت { } محصور می‌گردند.

توابع {if}، {elseif} و {else}

شاید باور نکنید که Smarty نوعی از ساختارهای تکرار و تصمیم‌گیری (یا به‌طور کلی ساختارهای کنترلی) را جهت استفاده در الگوها مورد پشتیبانی قرار می‌دهد. این ساختارهای کنترلی دقیقاً مشابه ساختارهایی هستند که از یک زبان برنامه‌نویسی یا اسکریپت‌نویسی جامع و همه‌منظوره

انتظار داریم.

تابع {if} دقیقاً عملکردی مشابه ساختار تصمیم‌گیری if در زبان برنامه‌نویسی PHP دارد. توجه کنید که علاوه بر واژه if (نام تابع)، عبارت منطقی مورد ارزیابی نیز در درون جفت علامت { } واقع می‌گردد. چنان‌چه حاصل این عبارت منطقی برابر با مقدار true ارزیابی شود، دستورالعمل مربوطه اجرا شده و در غیر این‌صورت از اجرای آن دستورالعمل صرف‌نظر به‌عمل می‌آید. به‌نمونه‌ای از چگونگی استفاده از این تابع توجه نمایید:

```
{if $user = = "admin"}
   < p > Message of the day : {$motd} < / p >
{ / if }
```

به عبارتی که در ازای ارزیابی عبارت منطقی به صورت true باید اجرا شود، توجه نمایید. یک چنین عبارت یا عباراتی باید مابین فراخوانی تابع {if} و نشانه پایانی { / if } واقع شود. این همان منطقی است که Smarty از آن جهت تعریف یک بلوک از خروجی استفاده می‌کند.

در قطعه کد بالا همان‌گونه که مشاهده می‌کنید، مقدار متغیر {$user} را مورد ارزیابی قرار دادیم و در صورتی که این مقدار برابر با دنباله کاراکتری "admin" باشد، پیغامی را در خروجی نمایش می‌دهیم. حقیقت آن‌است که Smarty در مورد نحوه استفاده از تابع { if } بسیار حساس است. فراموش نکنید که از دید Smarty درج یک کاراکتر فضای خالی مابین هر عملگر و عملوندهای آن بخشی از ضروریات استفاده از این تابع محسوب می‌شود.

همانند زبان برنامه‌نویسی PHP در رابطه با شیوه‌های مختلف در تصمیم‌گیری، در اینجا نیز می‌توانید به هنگام نیاز از توابع {else} و {elseif} به همراه تابع { if } استفاده نمایید:

```
{ if $user = = "admin" }
      < p > Message of the Day : { $motd } < / p >
{ elseif $user = = "super" }
      < p > System status : {$sys _ status } < / p >
{ else }
      < p > Hope you enjoy our site ! < / p >
{ / if }
```

چنانچه در قطعه کد بالا ملاحظه می‌کنید تابع { if } دو تابع دیگر یعنی { else } و {elseif } را در برگرفته است. این بدان معنی است که بهره‌برداری از این توابع به صورت تودرتو کاملاً عملی می‌باشد. اگرچه ممکن است شکل ظاهر این توابع با آنچه که پیشتر شاهد بودید اندکی متفاوت باشد اما این قطعه برنامه در مجموع شکل کاملاً آشنایی را در ذهن تداعی می‌کند.

تشکیل یک ساختار تکرار با استفاده از تابع {section}

یکی از ارکان اساسی هر موتور الگوسازی داشتن ابزاری است که با استفاده از آن بتوان فرآیندی را به‌صورت تکراری (مشابه ساختارهای تکرار در زبان‌های برنامه‌نویسی) انجام داد. با فرض

اینکه تعداد دفعات تکرار یک فرآیند از طریق یک بانک اطلاعاتی تأمین شده باشد به شیوه‌ای کاملاً قابل‌انعطاف‌جهت نمایش‌و ارائه اطلاعات‌درخروجی نیاز خواهیم داشت. در برنامه‌ای که در لیست ۶- ۲۳ مشاهده خواهید نمود، فرض بر این است که عناصر آرایه $search از طریق دستیابی به یک بانک اطلاعاتی تأمین شده‌اند. چنانکه ملاحظه می‌کنید این آرایه جهت پردازش به متد ( ) assign ارسال شده است.

```
 1: <?php
 2: require_once("Smarty.class.php");
 3:
 4: $search = array(
 5:                 "Douglas Adams",
 6:                 "Neal Stephenson",
 7:                 "Dan Simmons",
 8:                 "Peter F. Hamilton"
 9:                 );
10:
11: $templ = new Smarty();
12: $templ->assign( "search", $search );
13: $templ->display("listing23.7.tpl");
14: ?>
```

لیست ۶- ۲۳ آرایه‌ای از اسامی فرضی جهت ارسال به یک الگوی نمونه

با تعریف این آرایه در خط ۴ از برنامه فوق یک الگوی نمونه قادر خواهد بود تا با استفاده {section}، ساختاری تکراری بر مبنای عناصر ذخیره شده در آن تشکیل داده و در هر بار گذر از این ساختار تکرار یکی از عناصر آن‌را مورد دستیابی و پردازش قرار دهد. لیست ۷- ۲۳ چگونگی انجام این عمل را نشان می‌دهد.

```
 1: <html>
 2: <head>
 3: <title>Listing 23.7 Looping with the section Function</title>
 4: </head>
 5: <body>
 6:
 7: <h3>Author Search Results</h3>
 8: {section name=search_row loop=$search}
 9:     {$search[search_row]}<br>
10: {/section}
11:
12: </body>
13: </html>
```

لیست ۷- ۲۳ تشکیل یک ساختار تکرار با استفاده از تابع {section}

مشابه تمام ساختارهای تکراری که تاکنون در زبان برنامه‌نویسی PHP بررسی کردیم تابع {section} نیز نیازمند روشی برای محدود کردن تعداد دفعات تکرار یک فرآیند است. جهت انجام این کار در تابع{section} ما از آرگومان ویژه‌ای با عنوان loop استفاده می‌کنیم. این آرگومان به‌طور دقیق

تعداد دفعاتی را که فرآیند موردنظر باید تکرار شود مشخص می‌کند. در برنامه موجود در لیست ۶- ۲۳ چنان‌که مشاهده کردید ما آرایه‌ای با نام $search را ایجاد کرده و آن‌را در اختیار الگوی موردنظرمان قرار داده‌ایم. بدین ترتیب تابع {section} که در خط ۸ از این فایل الگو فراخوانی شده است به‌خوبی قادر به شمارش تعداد عناصر آرایه $search خواهد بود. در حقیقت فراخوانی زیر:

```
{section name = search _ row loop = $search}
```

با این فراخوانی، یعنی :

```
{section name = search _ row loop = 4}
```

کاملاً معادل خواهد بود. همان‌گونه که ملاحظه می‌کنید آرگومان loop به‌خوبی تعداد دفعات تکرار عملیات موردنظر را مشخص می‌کند. همچنین آرگومان دیگر با عنوان name از این تابع روشی را جهت دستیابی به اندیس‌ها یا شاخصهای عناصر موجود در آرایه در اختیار قرار می‌دهد. بنابراین در اولین حلقه از این ساختار تکرار، عبارت زیر:

```
$search [ search _ row ]
```

معادل این عبارت خواهد بود:

```
$Search [ 0 ]
```

و به همین ترتیب چنین وضعیتی برای سایر حلقه‌ها نیز عیناً وجود خواهد داشت. به‌عنوان مقدار آرگومان name از تابع مورد بحث یعنی {section} می‌توانید از هر دنباله کاراکتری که شامل یک کلمه (فاقد فضای خالی) باشد، استفاده نمایید. در داخل حلقه تکرار می‌توان ازاین مقدار به‌عنوان متغیری جهت دستیابی به شاخص متناظر از عناصر هر آرایه‌ای که الگوی موردنظر به آن دسترسی دارد، استفاده نمود. با این وجود استفاده از این آرگومان تنها محدود به دستیابی به آرایه مشخص شده توسط آرگومان loop نمی‌باشد.

از تابع{section} می‌توان جهت دستیابی به فیلدهای آرایه انجمنی از یک آرایه چندبعدی نیز استفاده نمود. به‌واسطه یک چنین قابلیتی می‌توان برنامه نمونه‌ای را که در لیست ۶- ۲۳ در مورد اطلاعات مؤلفین ارائه دادیم، به‌راحتی گسترش داد. برای این منظور فرض کنید آرایه موردنظرمان به‌صورت زیر تعریف شده باشد. این آرایه جدید علاوه بر اسامی مؤلفین شامل عناوین کتابهای مربوطه نیز می‌باشد:

```
$search = array (
        array (' name ' ⇒ "Douglas Adams" ,
              'book' ⇒"So Long and Thanks for all the fish") ,
        array ( ' name ' ⇒ "Neal Stephenson" ,
              'book' ⇒ "Cryptonomicon") ,
        array ( ' name ' ⇒ "Dan Simonson" ,
              'book' ⇒ "Endymion") ,
        array ( ' name ' ⇒ "Peter F . Hamilton" ,
              'book' ⇒ "The Neutronium Alchemist") ,
        ) ;
```

با فرض استفاده از این آرایه چندبعدی در برنامه لیست ۶- ۲۳ می‌توانیم به هر یک از عناصر آرایه‌های انجمنی موجود در آن با استفاده از عملگر آشنای (.) به صورت زیر دسترسی داشته باشیم:

```
{section name = search _ row loop = $search}
    {$search [search _ row]. name} . . {$search [search _ row] . book} < br >
{ / section}
```

با این وضعیت این پرسش منطقی را می‌توان مطرح کرد که اگر آرایه $search تهی باشد چه اتفاقی روی خواهد داد؟ آیا می‌توان با به‌کارگیری یک روش خاص هنگام ایجاد الگوی موردنظر حالت پیش‌فرض را نیز به‌نحوی کنترل نمود؟ پاسخ به این پرسش توسط به‌کارگیری تابع دیگری با عنوان {sectionelse} قابل ارائه است. این تابع هنگامی فراخوانی می‌شود که آرگومان loop از تابع {section} برابر با عدد صحیح صفر ارزیابی شود:

```
{section name = search _ row loop = $search}
   {$search [ search _ row ] . name} . . {$search [ search _ row ] . book} < br >
{sectionelse}
   No authors found . Try another search .
{ / section}
```

علاوه بر موارد فوق، تابع {section} با تدارک چند متغیر ویژه امکان تشخیص وضعیت حلقه را در اختیار برنامه‌نویس قرار می‌دهد. این متغیرها قالب خاصی دارند که به شکل زیر است:

```
{% loopname . variable %}
```

در میان این متغیرها شاید بتوان گفت که مهم‌ترین متغیر iteration می‌باشد. متغیر iteration شماره حلقه‌ای از ساختار تکرار را در هر لحظه ثبت می‌کند. قابل ذکر است که شمارش حلقه‌ها از عدد 1 آغاز می‌شود. بدین ترتیب متغیر مذکور ابزار بسیار ایده‌آلی برای شمارش سطرها محسوب می‌شود. به نمونه‌ای از به‌کارگیری این متغیر توجه کنید:

```
{section name = search _ row loop = $search}
   {% search _ row . iteration %} . {$search [ search _ row ] . name} ..
      {$search [search _ row] . book} < br >
{sectionelse}
```

متغیر مهم دیگر از سری متغیرهای تابع {section} با عنوان index کاربردی شبیه به متغیر iteration دارد. این متغیر شماره شاخص جاری از حلقه تکرار را مشخص می‌کند. بدین ترتیب می‌توان حدس زد که مبنای شمارش این متغیر عدد صحیح صفر باشد. شاید این اختلاف مابین متغیرهای index و iteration ناچیز جلوه کند. درحقیقت اختلاف اساسی مابین این دو متغیر هنگامی آشکار می‌شود که از دو آرگومان دیگر در تابع {section} استفاده شود. این دو آرگومان با اسامی start و step مشخص می‌شوند. آرگومان start مقدار اولیه شاخص شمارنده حلقه را تعیین می‌کند. آرگومان step نیز میزان افزایش شاخص شمارنده مذکور را به‌ازای هر بار گذر از این ساختار تکرار مشخص می‌کند (همان‌گونه که مشاهده می‌کنید این رفتار کاملاً مشابه رفتار حلقه‌های تکرار در زبانهای برنامه‌نویسی، همچون PHP است). درصورتی که از این دو آرگومان جهت کنترل ساختارتکراری که توسط تابع

{section} پیاده‌سازی می‌شود، استفاده کنیم؛ آن‌گاه متغیر index منعکس‌کننده مقادیر آرگومان‌های start و step بوده حال آنکه متغیر iteration بدون توجه به مقادیر این متغیرها کماکان از عدد صحیح 1 آغاز شده و با گامهایی به میزان یک واحد افزایش می‌یابد.

متغیرهای مفید دیگری که می‌توان از سری متغیرهای تابع {section} به آنها اشاره نمود متغیرهایی با عنوان first و last می‌باشند. با شروع ساختار تکرار مقدار متغیر first برابر با true خواهد بود. در این حالت مقدار متغیر last برابر با false است. در دومین حلقه تکرار مقدار متغیرهای first و last هر دو برابر با false خواهد بود (این به شرطی است که ساختار مذکور تنها شامل دو حلقه تکرار نباشد). با به انتها رسیدن ساختار تکرار مذکور مقدار متغیر first کماکان برابر با true بوده ولی مقدار متغیر last برابر با true خواهد شد. با بهره‌گیری از این دو متغیر می‌توان خروجی برنامه را با توجه به مقادیر ابتدایی و انتهایی حلقه قالب‌بندی نمود. قطعه برنامه زیر این روند را نشان می‌دهد:

```
{section name = search _ row loop = $search}
   {if % search _ row . first %}
      < hr >
   < / if >
   {% search _ row . iteration %}. {$search [ search _ row ]. name}. .
      {$search [ search _ row ] . book} < br >
   {if % search _ row. last %}
      < hr >
      { / if }
{sectionelse}
   No authors found . Try another search
{ / section}
```

در برخی موارد نمایش تعداد سطرهای بازیابی شده توسط پرس و جویی که کاربر از طریق برنامه ارسال نموده می‌تواند برای وی جالب توجه باشد. متغیر total از متغیرهای تابع {section} به ما این امکان را می‌دهد تا تعداد مجموع دفعات تکرار یک ساختار را بدون توجه به اینکه در کدام حلقه تکرار واقع هستیم، در اختیارمان قرار دهد. به‌عبارت دیگر این متغیر مستقل از حلقه‌های تکرار می‌باشد. قطعه برنامه زیر چگونگی استفاده از این متغیر را نشان می‌دهد:

```
{section name = search _ row loop = $search}
   {if % search _ row. first %}
      {% search _ row. total %} found
      < hr >
      { / if }
   { % search _ row. iteration %} . {$search [ search _ row ]. name} . .
      {$search [ search _ row ] . book} < br >
   {if % search _ row. last %}
      < hr >
      { % search _ row. total %} found
   { / if }
{sectionelse}
```

No authors found . Try another search
{ / section}

بدین ترتیب اکنون با چند ویژگی و قابلیت مهم در رابطه با طراحی الگوها آشنا شدید. به‌واسطه این قابلیتها قادر خواهید بود تا عباراتی شبیه به عبارات تصمیم‌گیری و همچنین ساختارهایی شبیه به ساختارهای تکرار موجود در زبانهای برنامه‌نویسی را در طراحی الگوهای مورد نظرتان مورد استفاده قرار داده و بدین ترتیب الگوهای پیچیده‌تری را ایجاد نمایید. این نکته را در طراحی برنامه‌های کاربردی همواره به‌خاطر داشته باشید، بخشی از برنامه که مسئول نمایش اطلاعات به کاربر است (معمولاً این بخش با عنوان لایه نمایش یا presentation Layer شناخته می‌شود) باید قادر باشد تا به سادگی با اطلاعاتی که در اختیار آن گذاشته می‌شود، کار کند. این روند باعث می‌شود تا بخشی از برنامه که پیاده‌سازی منطق یا الگوریتم برنامه را به‌عهده دارد از خطراتی که به‌واسطه ترکیب شده با کد لایه نمایش گریبان‌گیر آن می‌شود، مصون بماند. از طرف دیگر باید درمورد بهره‌گیری از برخی از بخشهای مربوط به پیاده‌سازی الگوریتم برنامه کاربردی در درون الگوها بسیار ملاحظه کارانه و با احتیاط عمل نمایید. به‌عنوان یک قانون نانوشته الگوهایی که در برنامه کاربردی خود مورد استفاده قرار می‌دهید، باید اطلاعات بسیار محدودی درباره هر نوع فرآیند دیگری که در خارج از فرآیند مربوط به نمایش اطلاعات (لایه نمایش) انجام می‌شود؛ داشته باشند.

طراحی برنامه‌های کاربردی طی گذشت سالها دستخوش تغییرات بسیاری شده است و این موضوع شامل حال برنامه‌های کاربردی وب نیز می‌شود. در حالی‌که مدلهای طراحی قدیمی بر روی یک الگوی دو لایه تاکید داشتند، طرحهای جدید مدلهای سه لایه و حتی N لایه را توصیه می‌کنند. وجود لایه‌ها اساساً خود بر اهمیت تفکیک کد برنامه بر قسمتهای مختلف تاکید می‌کند. در مدل دو لایه، لایه اول مسئول نمایش اطلاعات و ارائه یک رابط قابل استفاده در اختیار کاربر برنامه کاربردی است. ضمن اینکه کد مربوطه به برنامه کاربردی یا همان Application Logic نیز در همین لایه پیاده‌سازی می‌شود. لایه دوم در این مدلها مسئول کنترل داده‌ها است. این لایه معمولاً یک بانک اطلاعاتی است که اطلاعات لایه اول را تأمین می‌کند. لایه مذکور معمولاً با عنوان Data Layer مشخص می‌شود. در مدل سه لایه، از طرف دیگر، لایه سوم باز هم مسئول کنترل داده‌هاست اما بخشی از لایه اول که مسئول پیاده‌سازی کد برنامه کاربردی است در قالب لایه دیگری که به Application Layer معروف است، پیاده‌سازی می‌شود. بدین ترتیب این لایه از لایه‌ای که مسئول نمایش اطلاعات است، یعنی Presentation Layer کاملاً تفکیک می‌شود. چنین تفکیکی مزایای متعددی را به‌همراه می‌آورد که از آن جمله می‌توان به سادگی در اشکال زدایی برنامه‌ها اشاره کرد. همچنین امکان توسعه موازی این دو لایه توسط برنامه‌نویسان (که بر روی Application Layer کار می‌کنند) و طراحان رابط برنامه کاربردی (که برروی Presentation Layer کار می‌کنند) به‌خوبی فراهم می‌شود- مترجم

ترکیب الگوهای مختلف با استفاده از تابع {include}

شاید برای شما جالب باشد که بدانید هنگام فراخوانی متد ( ) display از شئ Smarty هیچ محدودیتی در استفاده از تعداد الگوها سد راه ما نیست. برخی از عوامل موجود در صفحات وب (برای نمونه ابزارهایی که جهت بازدید صفحات مختلف یک وب سایت مورد استفاده قرار می‌گیرند) را به احتمال قوی می‌توان در صفحات مختلف بارها و بارها مورد بهره‌برداری قرار داد. از این‌رو نیازی نیست تا این‌گونه عوامل بخشی از طراحی تمام الگوهایی را که بدانها نیاز دارند تشکیل دهند. این‌گونه عوامل رامی‌توان‌درقالب‌الگوهایی‌جداگانه‌پیاده‌سازی نموده و سپس در الگوهای مختلف جهت در اختیارگذاشتن قابلیت موردنظر جاسازی کرد (این فرآیند مشابه قابلیت استفاده مجدد از کدهای نوشته شده در برنامه‌های مختلف است. درصورتی‌که تا به حال برنامه‌نویسی به شیوه شئ‌گرا یا OOP را تجربه کرده باشید به‌طور حتم اکنون اطلاع دقیقی از این ویژگی خواهید داشت).

بهره‌گیری از الگوهای مختلف در درون یکدیگر به شیوه‌ای بسیار ساده در Smarty قابل پیاده‌سازی است. این کار به سادگی فراخوانی تابعی با عنوان {include} است. تابع {include} دارای آرگومان ورودی با عنوان file است که مسیر الگوی موردنظر را جهت درج در الگوی جاری (یعنی الگویی که تابع {include} در آن‌جا فراخوانی شده است) مشخص می‌کند. باردیگر تأکید می‌کنیم که مسیر فوق را باید به‌صورت نسبی و نسبت به فهرست templates که حاوی تمام الگوهای مورد استفاده در Smarty است بیان نمود. تابع {include} علاوه بر گنجاندن یک الگو در درون دیگری قابلیت مضاعفی نیز دارد. با بهره‌گیری از این تابع می‌توان در صورت تمایل می‌توان متغیرهای مختلفی را به هر تعداد که مایل باشیم در قالب آرگومان‌های تابع فوق از طریق الگوهایی که در الگوی اصلی شامل می‌شود در اختیار آن قرار دهیم. بدین ترتیب کلیه آرگومان‌های تابع {include} به‌همراه مقدار مربوطه در دسترس الگوی اصلی قرار خواهند گرفت. هنگام بهره‌گیری از قابلیت اخیر تابع {include} توجه به این نکته مهم کاملاً ضروری است. از آنجا که آرگومان file از این تابع یک آرگومان ضروری برای تابع فوق می‌باشد استفاده از نام file به‌عنوان یکی از متغیرها جهت ارسال مقدار آن به الگوی اصلی عملکرد نامطلوبی را از سوی Smarty در پی خواهد داشت. غیر از نام file از هرنام دیگری می‌توانید برای منظور فوق استفاده کنید.

فراخوانی زیر را درنظر بگیرید:

{include file = "brand_ top . tpl" section = "author search" }

با بهره‌گیری از این فراخوانی در الگوی نمونه‌ای که پیش از این ملاحظه نمودید محتوای الگوی موجود در فایل brand _ top . tpl دراختیار الگوی مثال ما قرار خواهد گرفت. علاوه بر این الگوی مثال ما به متغیری با نام {$section} و مقدار مربوط به آن نیز دسترسی خواهد داشت:

< h 1 > books unlimited < / h1 >
< uL >

```
< h4 > {$section} < / h4 >
< / ul >
```

تغییر متغیرهای الگو

همان‌گونه که در قسمتهای مختلف درس این ساعت مشاهده کردید Smarty امکانات بسیار ارزشمندی را در رابطه با تفکیک لایه نمایش از منطق برنامه کاربردی در اختیار ما قرار داده است. با این وجود هنوز صحبتی در مورد یکی از امکانات مهمی که هر موتور الگوسازی باید در اختیار برنامه‌نویس قرار دهد به‌میان آورده نشده است. چنان‌که اطلاع دارید داده‌های مختلف پیش از آنکه وضعیت نهایی آنها جهت ارائه در لایه نمایش مشخص شود لازم است تا بارها و بارها دستخوش تغییر شوند. در این فرآیند، برای مثال ممکن است برخی از نشانه‌ها حذف شوند و یا علایم خط جدید که به‌صورت n \ مشخص می‌شوند به نشانه‌هایی از نوع < br > تبدیل شوند. همچنین ممکن است بخشهایی از یک متن به حروف بزرگ یا کوچک تبدیل شوند. انتظار می‌رود که Smarty به‌عنوان یک موتور الگوسازی کارآمد و مدعی ابزارهایی را جهت انجام این‌گونه فرآیندها پیش‌بینی کرده باشد.

Smarty البته برای هر کدام از این تغییرات ابزارهایی را با عنوان "تغییردهنده‌های متغیر" یا variable modifier در دسترس طراحان و برنامه نویسان قرار داده است. جهت اعمال یک تغییردهنده به متغیری از یک الگوی نمونه لازم است تا از کاراکتر ویژه‌ای که به علامت خط لوله یا pipe شهرت دارد استفاده شود. شکل ظاهری این علامت به‌صورت | بوده و نحوه استفاده از آن جهت تغییر متغیر به‌صورت زیر است:

```
{$variable | modifier}
```

جهت افزایش قابلیت این فرآیند می‌توان چندین علامت | را به‌طور زنجیره‌ای مورد استفاده قرار داد. در این صورت خروجی هر یک از آنها به‌عنوان ورودی دیگری مورد استفاده قرار خواهد گرفت:

```
{$variable | firstmodifier | secondmodifier}
```

درصورت استفاده از این شیوه زنجیره‌ای، تغییردهنده‌ها به ترتیب از چپ به راست بر متغیر موردنظر اعمال می‌شوند. بدین ترتیب که ابتدا سمت چپ‌ترین تغییردهنده بر روی متغیر اعمال شده و خروجی آن به‌عنوان مقدار جدید متغیر درنظر گرفته می‌شود. سپس تغییردهنده بعدی بر روی این مقدار اعمال می‌شود و این روند تا انتها ادامه می‌یابد.

چنانچه تغییردهنده نیازمند دریافت آرگومان باشد (برخی از تغییردهنده‌ها به‌مانند توابع آرگومان دریافت می‌کنند)، آرگومان‌ها به‌ترتیب یکی پس از دیگری بعد از نام تغییردهنده ظاهر شده و با بهره‌گیری از علامت کولون (:) از یکدیگر جدا می‌شوند:

```
{$variable | modifier : "an _ arg" : "another _ arg" }
```

آرگومان تغییردهنده‌ها می‌توانند از نوع دنباله‌های کاراکتری و اعداد (صحیح و اعشاری) بوده و خود به‌سادگی متغیر دیگری باشند.

تغییردهنده‌های capitalize و lower

تغییردهنده capitalize چنانکه از نام آن نیز پیداست اولین حرف از کلیه کلمات موجود در یک دنباله کاراکتری را به حروف بزرگ تبدیل می‌کند. بدین ترتیب عبارت زیر اولین حرف از نام مؤلفین را به حرف بزرگ تبدیل خواهد کرد:

```
{$author | capitalize}
```

تغییردهنده lower فرآیندی معکوس capitalize را انجام می‌دهد، با این تفاوت که تمامی حروف از کلیه کلمات موجود در یک دنباله کاراکتری را به حروف کوچک تبدیل می‌کند:

```
{ $author | lower }
```

البته می‌توان این دو تغییردهنده را به‌صورت زنجیره‌ای نیز مورد استفاده قرار داد. این فرآیند در برخی موارد کاربرد بسیار مناسبی دارد، چرا که در این حالت ابتدا تغییردهنده lower کلیه حروف موجود در دنباله کاراکتری را به حروف کوچک تبدیل کرده و سپس تغییردهنده capitalize اولین کاراکتر از هر کلمه از این دنباله کاراکتری را به حروف بزرگ تبدیل می‌کند. نحوه استفاده زنجیره‌ای از این دو تغییردهنده به صورت زیر است:

```
{ $author | lower | capitalize }
```

دقت کنید که در صورت عدم رعایت ترتیب تغییر دهنده‌ها، نتیجه مطلوب را در پی نخواهد داشت.عبارت زیر باعث می‌شود تا کلیه کاراکترهای موجود در دنباله کاراکتری $author (بدون اینکه تغییردهنده capitalize تأثیر خود را در بزرگ کردن حروف حفظ کرده باشد)، به حروف کوچک تبدیل شوند:

```
{ $author | capitalize | lower }
```

تغییردهنده regex _ replace

گاهی‌ازاوقات‌به‌توانی‌بیش‌ازآنچه‌که‌تغییردهنده‌های عادی مثل capitalize و lower در اختیارمان قرار می‌دهند، نیاز داریم. تغییردهنده regex _ replace به ما این امکان را می‌دهد تا یک عبارت منظم سازگار با زبان برنامه‌نویسی perl را به یک متغیر اعمال نماییم. تغییردهنده regex _ replace جهت انجام عملیات موردنظر خود به دو آرگومان نیاز دارد. اولین آرگومان این تغییردهنده الگویی را مشخص می‌کند که عملیات موردنظر بر مبنای آن انجام می‌شود. دومین آرگومان نیز دنباله کاراکتری جایگزین را مشخص می‌کند. در عبارتی که در ادامه مشاهده خواهید کرد از یک عبارت منظم جهت تعویض نام و نام خانوادگی موجود در دنباله کاراکتری $author استفاده شده است:

```
{$author    regex _ replace : " / ^ ( . * ) \ s ( . * ) $ / " : ' $2 , $1 ' }
```

تغییردهنده string _ format

تغییردهنده string _ format یکی از توابع PHP با عنوان ( ) sprintf را جهت تغییرقالب‌بندی محتوای یک متغیر مورد استفاده قرار می‌دهد (با این توضیح که اثری از نام تابع ( ) sprintf در این ترکیب‌بندی به‌چشم نخواهد خورد). این تغییردهنده جهت انجام عملیات موردنظر مستلزم دریافت یک آرگومان است. این آرگومان یک دنباله کاراکتری است که شیوه قالب‌بندی توسط تابع ( ) sprintf را مشخص‌می‌کند.برای‌نمونه به‌منظور نمایش یک مقدار پولی (که معمولاً با عنوان Currency می‌شناسیم) کافی است تا از دنباله کاراکتری " 2 f . % " به‌عنوان آرگومان این تغییردهنده به‌صورتی که در قطعه کد زیر مشخص کرده‌ایم، استفاده نماییم. (معمولاً دو رقم اعشار برای نمایش مقادیر پولی کافی‌است):

```
That will be
{$price    regex _ replace : "% . 2 f "}
dollars please.
```

دقت‌کنیدکه‌کلیه‌ملاحظات‌مربوط‌به‌آرگومان‌تابع( ) sprintf درموردتغییردهنده string _ format نیز صدق می‌کند. جهت اطلاع از این ملاحظات می‌توانید به درس ساعت هفدهم با عنوان "بهره‌گیری از دنباله‌های کاراکتری" مراجعه کنید.

تغییردهنده default

تغییردهنده default یکی از تغییردهنده‌های بسیار مفید است. با بهره‌گیری از این تغییردهنده می‌توان درصورت عدم امکان دستیابی به متغیر موردنظر، سند وب را به‌گونه‌ای مناسب با شرایط پیش‌آمده تطبیق داد. این تغییردهنده جهت انجام عملیات خود تنها به یک آرگومان نیاز دارد. آرگومان مذکورمتغیری را مشخص می‌کند که درصورت عدم دستیابی Smarty به متغیر موردنظر به عنوان خروجی مورد استفاده قرار می‌گیرد. به نمونه‌ای از چگونگی استفاده از این تغییردهنده توجه کنید:

```
{$author | default : "anonymous" }
```

بررسی یک مثال کامل از نحوه بهره‌برداری از Smarty

چنانکه در این درس متوجه شدید Smarty را می‌توان در واقع یک کتابخانه جامع جهت استفاده به‌همراه آن دسته از برنامه‌های کاربردی وب که توسط زبان برنامه‌نویسی PHP توسعه یافته‌اند دانست. با وجودی که طی بررسی‌های خود در این ساعت به‌دلیل کمبود زمان و فضا قادر به موشکافی دقیق کلیه ویژگی‌ها و قابلیتهای این کتابخانه کارآمد و توانمند نشدیم اما در مجموع می‌توانیم چنین ادعا کنیم که مطالب مفیدی را در رابطه با آن فراگرفتیم. در این قسمت از درس این ساعت جهت

تکمیل مطالب فوق سعی خواهیم کرد تا با جمع‌آوری قطعه برنامه‌هایی که در قسمتهای مختلف از این ساعت ارائه کردیم و قالب‌بندی آن به صورت یک مجموعه واحد و همچنین اضافه کردن برخی قابلیتهای دیگر به مثال جامعی که درخور Smarty باشد، برسیم.

برنامه موجود در لیست ۸- ۲۳ بار دیگر کدبرنامه PHP را ارائه کرده است. همان‌گونه که شاهد هستید تغییرات بسیار عمده‌ای را نسبت به قبل مشاهده نمی‌کنیم.

```
1: <?php
2: require_once("Smarty.class.php");
3:
4: $search = array(
5:                 array( 'name'=>"Douglas Adams",
6:                'book'=>"So Long and Thanks for all the Fish" ),
7:                 array( 'name'=>"NEAL STEPHENSON",
8:                        'book'=>"CRYPTONOMICON" ),
9:                 array( 'name'=>"dan simmons",
10:                       'book'=>"endymion" ),
11:                array( 'name'=>"Peter F Hamilton",
12:                       'book'=>"The Neutronium Alchemist" ),
13:                array( 'name'=>"",
14:                        'book'=>"Prime Colors" ),
15:                );
16: $templ = new Smarty();
17: $templ->assign( "section", "book search" );
18: $templ->assign( "search", $search );
19: $templ->display("listing23.9.tpl");
20: ?>
```

لیست ۸- ۲۳ یک برنامه PHP که از الگوهای Smarty استفاده می‌کند.

جالب توجه‌ترین نکته‌ای را که می‌توان در رابطه با این برنامه عنوان کرد این است که حجم آن بسیار پایین می‌باشد. هدف اصلی از توسعه این برنامه ترجمه داده‌هاست. پس از انجام این کار برنامه فوق داده‌های ترجمه شده را جهت قالب‌بندی به الگوی مورد استفاده تحویل می‌دهد. تنها تصور کنید درصورتی‌که مجبور به درج خروجی در برنامه لیست ۸- ۲۳ بودیم، وضعیت به چه ترتیب تغییر می‌کرد. در واقع ما تغییرات اندکی را نسبت به گذشته در این برنامه اعمال کرده‌ایم. یکی از این تغییرات این است‌که در خط ۱۳ از برنامه، کتاب جدیدی را بدون اینکه نام مؤلف آن‌را مشخص کرده باشیم به لیست کتابهای موجود در آرایه $search اضافه نموده‌ایم. تغییر دیگری که می‌توان به آن اشاره کرد فراخوانی جدیدی برای تابع ( ) assign است که در خط ۱۷ برنامه انجام گرفته است. در این فراخوانی جدید نام بخش موردنظر از کتاب جهت جستجو به الگوی مورد استفاده ارسال می‌گردد.

کد موجود در لیست ۹- ۲۳ الگوی اصلی مورد استفاده در این مثال را نشان می‌دهد.

```
 1: {include file="listing23.10.tpl" section=$section}
 2:
 3: {section name=search_row loop=$search}
 4:     {if %search_row.first%}
 5:         {%search_row.total%} found
 6:         <hr>
 7:     {/if}
 8:     {%search_row.iteration%}.
 9:
➥{$search[search_row].name|lower|capitalize|regex_replace:"/^(.*)\s(.*)$/":'$2,
➥$1'|default:"anonymous"}
10:     ..
11:     {$search[search_row].book|lower|capitalize} <br>
12:     {if %search_row.last%}
13:         <hr>
14:         {%search_row.total%} found
15:     {/if}
16: {sectionelse}
17:     No authors found. Try another search
18: {/section}
19:
20: </body>
21: </html>
```

لیست ۹- ۲۳ الگوی اصلی

چنانکه در کد مربوط به این الگو مشاهده می‌کنید، قاعده تفکیک بخش.نمایش از بخش اصلی رعایت شده است به‌گونه‌ای که مسئولیت نمایش هدر سند به‌عهده فایل دیگری گذاشته شده است. این فرآیند با بهره‌گیری از تابع {include} در اولین خط از کد این الگو انجام می‌شود. ضمن اینکه متغیر {$section} نیز به‌عنوان آرگومان این تابع مورد استفاده قرار گرفته است.

ارسال آرگومانی با عنوان {$section} به تابع {include} به این معنی است که بیشتر کاری که این الگو انجام می‌دهد توسط تابع {section} پیاده‌سازی شده است (خط ۳ تعریف این تابع را نشان می‌دهد). به‌ازای اولین حلقه تکرار (که توسط عبارت موجود در خط ۴ مشخص می‌شود) یک دنباله کاراکتری شامل تعداد کل موارد قابل تطبیق (مثلاً تعداد سطرهای بازیابی شده از بانک اطلاعاتی) به عنوان خروجی، نمایش می‌یابد. همان‌گونه که در خط ۵ از این لیست مشاهده می‌کنید جهت دستیابی به مجموع موارد قابل تطبیق ازمتغیر total که پیش از این درباره آن صحبت نمودیم، استفاده کرده‌ایم.

متغیر iteration یکی دیگر از متغیرهایی است که در توسعه این الگو از آن استفاده شده است. ما از این متغیر در خط ۸ به‌منظور شمارش سطرها استفاده کرده‌ایم. بدین ترتیب از شماره‌گذاری از حلقه که در حال پردازش آن هستیم، مطلع خواهیم بود. چنان‌چه در خطوط ۹ و ۱۱ از این لیست ملاحظه می‌کنید با بهره‌گیری از عوامل تغییردهنده lower، capitalize، regex_ replace و default اقدام به قالب‌بندی متن مورد نظر نموده‌ایم. دو تغییردهنده lower و capitalize در این میان با بهره‌گیری از شیوه زنجیره‌ای ترتیبی می‌دهند تا هرگونه عدم هماهنگی در متن خروجی از میان رفته و

متن مذکور به‌شیوه‌ای همگن در خروجی به نمایش درآید. همچنین تغییردهنده regex_ replace به‌گونه‌ای عمل می‌کند که موقعیت نام و نام خانوادگی هر یک از مؤلفین در خروجی با یکدیگر تعویض شوند. درنهایت عامل تغییردهنده default ترتیبی می‌دهد تا متن دلخواهی هرجا که نام مؤلف در آرایه $search ذکر نشده است، به‌جای آن به نمایش درآید. در این مورد همان‌گونه که عبارت خط ۹ از لیست ۹- ۲۳ نشان می‌دهد از دنباله کاراکتری "anonymous" به‌عنوان متن جایگزین استفاده می‌شود.

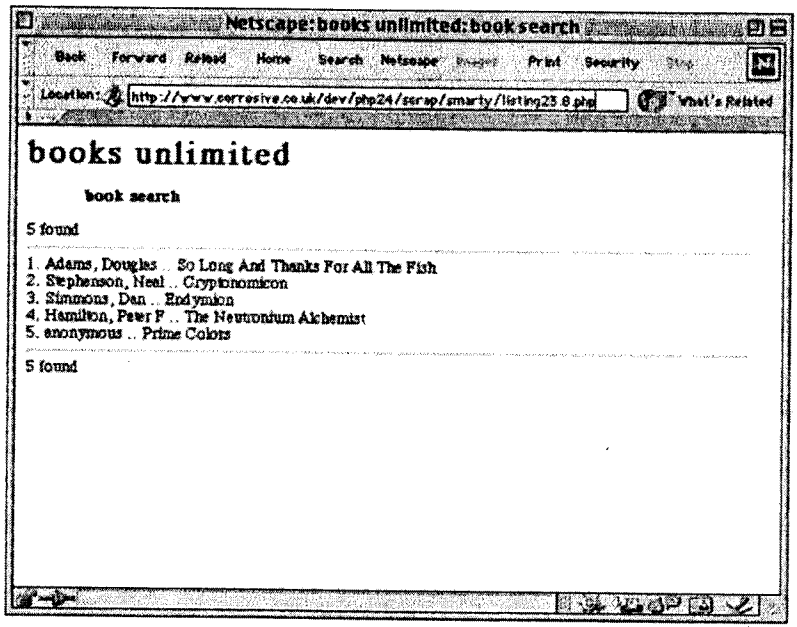
در مورد آخرین حلقه تکرار (که با استفاده از عبارت موجود در خط ۱۲ مشخص شده است) چنانکه مشاهده می‌کنید باردیگر تعداد موارد قابل تطبیق بر روی صفحه به نمایش درآمده است. اگر به هر دلیلی عملیات مذکور قادر به بازیابی هیچ اطلاعاتی از آرایه $search نشود، بلوکی از کد موجود در لیست ۹- ۲۳ که با عنوان {sectionelse} مشخص شده است به اجرا درآمده و بدین ترتیب دنباله‌ای از کاراکترها بر روی صفحه به نمایش درخواهد آمد (خطوط ۱۶ و ۱۷).

کد موجود در لیست ۱۰- ۲۳ فایل شامل هدر خروجی حاصل از اجرای برنامه را نمایش می‌دهد (این همان فایلی است که با استفاده از تابع {include} در اولین خط از الگو، در فایل مربوط به آن درج شده است). تنها نکته قابل ذکری که می‌توان در مورد این کد به آن اشاره کرد، این حقیقت است که فایل مذکور به متغیر {$section} دسترسی کامل داشته و از عامل تغییردهنده default در خط ۳ و ۹ بهره‌گرفته است (توجه کنید که فایل موجود در لیست ۱۰- ۲۳ نیز یک الگو بوده و با پسوند tpl . بر روی کامپیوتر ذخیره می‌شود). استفاده از این عامل تغییردهنده این اطمینان خاطر را به ما می‌دهد که درصورت عدم دستیابی به متغیر موردنظر به هر دلیل ممکن دنباله کاراکتری با مفهومی به‌جای آن نمایش پیدا می‌کند.

```
1: <html>
2: <head>
3: <title>books unlimited: {$section|default:"The Book Source"}</title>
4: </head>
5: <body>
6: <h1>books unlimited</h1>
7:
8: <ul>
9: <h4>{$section|default:"The Book Source"}</h4>
10: </ul>
```

لیست ۱۰- ۲۳ الگوی مربوط به هدر خروجی

خروجی حاصل از عملکرد الگوهای موجود در این فرآیند به‌صورتی که در شکل ۲- ۲۳ مشاهده می‌کنید، نمایان می‌شود.



شکل ۲- ۲۳ عملکرد الگوهای مورداستفاده در برنامه

جمع‌بندی

دردرس‌این‌ساعت‌راجع‌به‌یکی از مفاهیم مهم در توسعه برنامه‌های کاربردی وب، یعنی بهره‌گیری ازالگوهابحث‌و گفتگو نمودیم. همان‌گونه که متوجه شده‌اید، درصورتی‌که عضوی از یک تیم برنامه‌نویسی و طراحی برنامه کاربردی وب بوده و به‌ویژه درگیر انجام پروژه‌های برنامه‌نویسی بزرگ شامل تعداد زیادی سند با محتوای پویا (Dynamic Content) باشیم، به‌طور حتم باید استفاده از یک موتور الگوسازی مثل Smarty را در دستور کار خود قرار دهیم. مورد دیگری که در درس این ساعت به بررسی آن پرداختیم اهمیت تفکیک دو بخش عمده برنامه کاربردی از یکدیگر، یعنی منطق برنامه (Application) از لایه نمایش (Presentation Layer) بود. متوجه شدید که چنان‌چه منطق مربوط به پیاده‌سازی برنامه کاربردی را از کدی که عهده‌دار نمایش خروجی در قالبی خوشایند است به‌درستی تفکیک کنیم آنچه که درنهایت در قالب یک برنامه کاربردی عایدمان می‌شود به‌راحتی قابل نگهداری و توسعه خواهد بود. درصورتی که از زبان برنامه‌نویسی PHP جهت توسعه برنامه کاربردی وب موردنظر استفاده می‌کنید، توصیه ما این است که از موتور الگوسازی Smarty جهت کنترل خروجی برنامه استفاده کنید؛ چراکه این موتور الگوسازی به‌واسطه ترجمه الگوهای ساخته شده به کد PHP از سرعت بالایی برخوردار است، ضمن اینکه از قابلیتهای بسیار زیادی برخوردار بوده و به‌راحتی قابل گسترش و توسعه است.

در درس این ساعت مطالب بسیار مفیدی را درباره چگونگی استفاده از قابلیتهای موتور الگوسازی Smarty فراگرفتید و ضمن این‌کار با چگونگی نصب و ایجاد الگوهای ساده و همچنین استفاده از آنها در قالب یک برنامه بزرگ آشنا شدید. نحوه استفاده از متغیرهای خاص با عنوان متغیرهای الگو و نیز چگونگی ارسال این متغیرها از درون یک برنامه PHP بدون یک الگوی Smarty، که با پسوند tpl . در سیستم فایل کامپیوتر مشخص می‌شود، مطلب دیگری بود که در درس این ساعت با آن آشنا شدید. همچنین در این ساعت با توابع ویژه‌ای که به‌واسطه استفاده از موتور الگوسازی Smarty در اختیار برنامه‌نویس قرار می‌گیرند و معمولاً با عنوان توابع الگو شناخته می‌شوند آشنا شده و چگونگی استفاده ازآنهارا جهت کنترل خروجی حاصل از برنامه و قالب‌بندی نمایشی در قالب چند برنامه کوتاه فراگرفتید. در انتهای درس نیز با چگونگی تغییر متغیرهای الگو با استفاده از عوامل تغییردهنده آشنایی پیدا کردید.

اکنون‌با دانشی که درباره زبان برنامه‌نویسی PHP و قابلیتهای توانمند آن پیدا کرده‌ایم، می‌توانیم آستینها را بالا زده و خودمان اقدام به توسعه یک کتابخانه از توابع مفید نماییم. درس ساعت آخرمان طی این دوره آموزشی به این مبحث اختصاص یافته است.

پرسش و پاسخ

پرسش: آیا در درس این ساعت کلیه ویژگی‌های مربوط به موتور الگوسازی Smarty عنوان گردید؟

پاسخ: به هیچ وجه. موتور الگوسازی Smarty در واقع از ساختارهای بسیار منطقی برخوردار است و ما در درس این ساعت تنها این فرصت را داشتیم که برخی از مهم‌ترین قابلیتهای آن‌را جهت ایجاد و استفاده از الگوها مورد بحث و بررسی قرار دهیم. با این همه با دانشی که در همین حد راجع به Smarty به‌دست آورده‌اید، می‌توانیم به‌صورت محدودی استفاده از قابلیتهای آن‌را آغاز کرده و در پی مطالب آموزشی دیگری جهت توسعه مهارتهای خود در این زمینه باشید. اما اگر قصد شما این است که استفاده از موتور الگوسازی مورد بحث را در فهرست مطالبی قرار دهید که باید در آن زمینه استاد شوید، توصیه ما این است که مستنداتی را که توسط توسعه‌دهندگان Smarty در زمینه بهره‌برداری از آن‌تهیه و تدوین شده است حتماً مورد مطالعه دقیق و موشکافانه قرار دهید. دسترسی به این مستندات بسیار ساده بوده و تنها کافی است تا آدرس URL زیر را مورد بازیابی قرار دهید:

http : // www. Phpinsider. com / php / code / smarty / docs

با دستیابی به این سند منبع بسیار ارزشمندی از لیست توابع قابل استفاده از این موتور الگوسازی و جزئیات به‌کارگیری هر یک از آنها را در اختیار خواهید داشت. علاوه بر این اطلاعات بسیار

مفیدی درباره چگونگی توسعه توابع جدید و همچنین عوامل تغییردهنده موردنیاز جهت استفاده با Smarty به دست خواهید آورد.

تمرینها

هدف از این بخش ارائه تمرینهای مفید در قالب آزمون است. پاسخ آزمون بلافاصله بعد از آن آمده است. بخش فعالیتها شامل تمرینهای برنامه‌نویسی بوده و با هدف افزایش مهارت و قابلیتهای خواننده در زمینه برنامه‌نویسی کاربردی با PHP طراحی شده است و البته فاقد پاسخ می‌باشد.

آزمون

۱- در اختیار داشتن چه فایلی جهت بهره‌برداری از Smarty ضروری است؟

۲- از کدام متد Smarty می‌توان جهت ثبت یک متغیر به‌همراه آن استفاده کرد؟

۳- وجود کدام فهرست برای PHP لازم است تا برنامه توسعه‌یافته به این زبان برنامه‌نویسی قادر به بهره‌برداری از الگوهای توسعه‌یافته از طریق Smarty باشد؟

۴- چگونه می‌توان به یک متغیر نمونه مانند متغیر $SERVER ["SERVER _ NAME"] از یک برنامه PHP از درون یک الگوی Smarty دسترسی پیدا کرد؟

۵- چه اشتباهی در قطعه کد زیر مشاهده می‌کنید؟

```
{if $user = = "admin"}
   < p > Message of the Day : {$motd} < / p >
{ / if }
```

۶- چگونه می‌توان به تعداد حلقه‌های پردازش شده در یک ساختار تکرار با فرض اینکه عنوان حلقه در تابع {section} به صورت "thisloop" مشخص شده باشد، پی برد؟

۷- از کدام عامل تغییردهنده در Smarty می‌توان یک دنباله کاراکتری ایجاد کرد به‌گونه‌ای که تمامی کاراکترهای این دنباله با حروف کوچک به نمایش درآیند؟

پاسخ آزمون

۱- سند Smarty. class. Php شامل هسته اصلی موتور الگوسازی Smarty بوده و لذا دسترسی به این سند برای هر برنامه‌ای که قصد استفاده از آن را داشته باشد، ضروری است.

۲- با بهره‌گیری از متد ( ) assign از Smarty می‌توان متغیری از یک برنامه PHP را به آن ارسال نمود.

۳- فهرستی با عنوان include_ c فهرستی است که Smarty نتیجه حاصل از ترجمه (کامپایل) فایل شامل الگوها را در آن می‌نویسد.

۴- با بهره‌گیری از متغیری با نام {$smarty} به‌صورتی که در عبارت زیر مشاهده می‌کنید می‌توان به آرایه سیستمی $SERVER از PHP دسترسی پیدا کرد:

{$smarty. server. SERVER _ NAME}

۵- همان‌گونه که در مثال درس نیز تذکر دادیم موتور الگوسازی Smarty در شیوه‌ای که ما از دستورالعمل {if} ارائه شده توسط آن در پیش می‌گیریم، بسیار حساسیت نشان می‌دهد. در این مورد اشتباه از آنجا ناشی می‌شود که هیچ فضای خالی مابین عملوندها و عملگر مربوطه در دستورالعمل {if} وجود ندارد. وجود فضای خالی مابین عملوندها و عملگر مربوطه در این تابع از نکاتی است که رعایت آن همواره الزامی است. قطعه کد اصلاح شده به صورت زیر است:

```
{if $user = = "admin"}
   < p > Message of the Day : {$motd} < / p >
{ / if }
```

۶- متغیر total همواره شامل مجموع تعداد حلقه‌هایی است که باید طی یک ساختار تکرار مورد پردازش قرار گیرد. مقدار موجود در این متغیر مستقل از حلقه در حال پردازش بوده و بدون توجه به این مطلب همواره شامل یک مقدار ثابت است. عبارت زیر چگونگی دستیابی و نمایش مقدار این متغیر را نشان می‌دهد:

{% thisloop. total %}

۷- عامل تغییردهنده Lower قادراست تا کلیه کاراکترهای موجود در یک دنباله کاراکتری مانند $myval را در قالب حروف کوچک نمایش دهد. نمونه زیر چگونگی استفاده از آن‌را نشان می‌دهد:

{$myval Lower}

فعالیتها

۱- موتور الگوسازی Smarty را از طریق وب سایت http: // www. Phpinsider. com مورد دستیابی قرار داده و آن‌را جهت کار با برنامه‌های کاربردی PHP بر روی کامپیوتر تان نصب کنید.

۲- به برنامه‌هایی که تا بدین‌جای کتاب طی درسهای مختلف با هم توسعه دادیم نگاهی مجدد انداخته و برخی از آنها را جهت توسعه مجدد با بهره‌گیری از توانمندی‌های موجود در Smarty جهت استفاده از الگوها بازنویسی کنید. در هریک از این موارد ایده تفکیک منطق برنامه از لایه نمایش را تا حد امکان در فرآیند توسعه رعایت نمایید.

ساعت بیست و چهارم

توسعه کتابخانه‌ای با عنوان

page. inc. php

طی آخرین ساعت از دوره بیست و چهار ساعته آموزش زبان برنامه‌نویسی PHP قصد داریم تا مجموعه‌ای از کدهای کتابخانه‌ای را با قابلیت استفاده مجدد در برنامه‌هایی که در آینده ایجاد خواهیم کرد، توسعه دهیم. در حین انجام این فرآیند از کلیه ساختارها و تکنیکهایی که طی درسهای ساعات گذشته فراگرفتیم، استفاده خواهیم کرد.

مطالب زیر را در درس این ساعت مورد بررسی قرار می‌دهیم:

- **مکانیزه کردن کنترل جلسات**
- **تسهیل در فرآیند بازخورد (feedback)**
- **بررسی جهت اطمینان از ارسال فرم**
- **پیاده سازی یک محیط امن و محافظت شده توسط کلمه عبور**
- **کنترل دستیابی به اطلاعات در یک محیط محافظت شده**

در ادامه به بررسی هر یک از این موارد می‌پردازیم.

توسعه کلاسی به‌عنوان چارچوب کاری

پیش از هرچیز اجازه دهید تا به‌منظور تسهیل در فرآیند توسعه آنچه که به دنبالش هستیم یک کلاس مرجع با عنوان Page را که نقش چارچوب کاری را برای توسعه برنامه ایفا می‌کند، ایجاد نماییم. این کلاس رویه‌هایی را شامل می‌شود که تقریباً هر برنامه اسکریپتی به آنها احتیاج دارد. علاوه بر این قابلیتهایی را نیز به این کلاس اضافه می‌کنیم که معمولاً در بین تمامی برنامه‌های کاربردی متداول و مرسوم هستند. نکته مهمی که در توسعه این چارچوب مرجع باید همواره به‌خاطر داشته باشید، این است که کلاس Page درحقیقت یک مفهوم ذهنی (abstract) بوده و در واقع وجود خارجی نخواهد داشت. به‌عبارت دیگر قصد ما از ایجاد یک چنین کلاسی این نیست که در سایر برنامه‌های کاربردی خود نمونه‌هایی را از نوع کلاس Page، ایجاد کنیم. قصد واقعی ما از توسعه یک چنین کلاسی این است که فرآیند توسعه برنامه‌ها را با سرعت، دقت و قابلیت بیشتری انجام دهیم.

شیوه استفاده از کلاسهای مرجع یکی از ارکان اصلی توسعه برنامه‌های مختلف با استفاده از زبانهای برنامه‌نویسی شئ‌گرا یا Object_ Oriented می‌باشد. معمولاً در طراحی برنامه‌ها به شیوه شئ‌گرا، کلاسهای مرجع (که بیشتر آنها را با عنوان کلاسهای مجرد یا abstract classes می‌شناسیم) سطوح بالای ساختار سلسله مراتبی کلاسها را تشکیل می‌دهند. یک چنین سلسله مراتبی از کلاسها بر مبنای یکی از سه اصل مهم برنامه‌نویسی به شیوه شئ‌گرا، یعنی اصل وراثت یا Inheritance شکل می‌گیرد، بدین معنی که هر کلاس سطح پایین (کلاس فرزند) قابلیتها و خصوصیاتی را از کلاس موجود در سطح بالاتر (کلاس پدر) به ارث می‌برد (دو اصل دیگر عبارتند از کپسوله سازی یا Encapsulation و چندشکلی یا چندریختی یا Polymorphism). معمولاً کلاسهای سطح بالا در طراحی برنامه به‌قدری کلی تعریف می‌شوند که در عمل، نمونه گیری از آنها امکان‌پذیر نیست (مانند کلاس " اتومبیل" که بسیار کلی بوده و نمی‌توان به‌دلیل عمومیتی که دارد از آن نمونه‌گیری کرد). در مقابل کلاسهای سطوح پایین‌تر که قابلیتها را به‌طور دقیق‌تر و خاص‌تری تعریف می‌کنند قابل نمونه‌گیری هستند. به چنین کلاسهایی معمولاً کلاسهای واقعی یا concrete classes گفته می‌شود ـ مترجم

کلاس مرجع Page

کلاس مرجع Page به‌گونه‌ای طراحی می‌شود که هر برنامه‌ای که نمونه‌ای از آن‌را مورد استفاده قرار می‌دهد، شامل سه حالت مختلف باشد. این حالات عبارتند از مقداردهی ابتدایی (initialization)،

اجرا یا پردازش (execution) و تخریب (clean up). حالت مقداردهی درقالب متدی از این نمونه با عنوان ( ) init صورت می‌پذیرد. هرگونه عملیاتی که جهت شروع به کار برنامه، مانند مقداردهی متغیرها لازم باشد، باید در درون این متد انجام شود. نمونه‌های کلاس Page در حد مطلوب عملیاتی را از طریق اجرای این متد انجام می‌دهند اما بیشتر عملیات یا به عبارت دیگر، قسمت اعظم کار توسط کلاسهایی که طبق رابطه توارث (ویژگی Inheritence) از این کلاس به ارث می‌برند، انجام می‌شود.

در پاراگراف قبل در چند مورد اشاره به " نمونه‌های کلاس Page " داشتیم حال آنکه پیشتر چنین اظهار کردیم که این کلاس به‌دلیل عمومیت بیش از اندازه یک مفهوم ذهنی یا غیرواقعی (abstract) تلقی شده و قابل نمونه‌گیری نمی‌باشد. برصحت این گفته بار دیگر تاکید کرده و ضمن آن، نکته دیگری را در مورد رابطه بین کلاسها در سلسله مراتبی که در یک برنامه شئ‌گرا وجود دارد، عنوان می‌کنیم. بین کلاس سطح بالا (پدر) و کلاسهای سطوح پایین‌تر (فرزندان) رابطه ویژه‌ای با عنوان رابطه "بودن" یا "is- a" وجود دارد، بدین معنی که می‌توان گفت " نمونه‌ای از کلاس فرزند در واقع نمونه‌ای از کلاس پدر است." و البته عکس این موضوع صحیح نیست. برای مثال اگر کلاس "اتومبیل" را کلاس پدر فرض کنیم، می‌توانیم کلاس " اتومبیل مسابقه‌ای فرمول ـ ۲" را به‌عنوان کلاس فرزند درنظر بگیریم. بدین ترتیب می‌توان صحت این جمله را به‌راحتی تصدیق کرد که "هر اتومبیل مسابقه‌ای فرمول ـ ۲ یک نمونه اتومبیل است" (اما عکس این موضوع ابداً صحیح نیست). بنابراین می‌توان یک " اتومبیل فرمول ـ ۲ " خاص مانند " اتومبیل فرمول ـ ۲ راننده مشهور، مایکل شوماخر" را نمونه‌ای از کلاس " اتومبیل" فرض کرد (این فرض صحیح مبنای بهره‌گیری از قابلیتهای اصل سوم برنامه‌نویسی به شیوه شئ‌گرا، یعنی ploymorphism است). بدین ترتیب در مورد کلاس مرجع Page می‌توان نمونه‌هایی از کلاسهای فرزند را نمونه‌ای از این کلاس ذهنی دانست ـ مترجم

حالت بعدی با بهره‌گیری از متد دیگری با عنوان ( ) main حاصل می‌شود. این حالت با عنوان اجرا، پردازش یا execution مشخص می‌گردد. به‌طور معمول متد ( ) main محلی است که قسمت اعظم‌عملیات برنامه در آنجا انجام می‌شود. برخی از این عملیات عبارتند از پردازش ورودی‌های دریافتی از طریق فرمهای HTML یا اسناد PHP یا هر فرم ورودی دیگر، تهیه و بازیابی بازخورد حاصل از بازدید سایرین از وب سایت و کارهای مشابه. نکته جالب توجه این است که تعریف این متد در کلاس مرجع Page انجام شده و امکان رونویسی (تعریف دقیق یا جزئیات) آن در اختیار کلاسهای سطوح پایین‌تر یا همان فرزندان گذاشته شده است (در واقع هم تعریف متد ( ) init و هم تعریف متد ( ) main و چنانکه

به‌زودی مشاهده خواهید کرد تعریف متد ( ) clean_ up یعنی متد مربوط به حالت سوم به‌طور دقیق در کلاس مرجع Page انجام نشده است. همین عدم ارائه تعریف دقیق موجب می‌شود تا این کلاس به‌عنوان یک کلاس سطح بالا یا یک کلاس ذهنی تلقی شود. تعریف دقیق هر سه متد مذکور در کلاسهای فرزند از کلاس پدر Page ارائه می‌شوند). حالت سوم که اشاره به تخریب شئ دارد از طریق متد ( ) clean _ up پیاده‌سازی می‌شود. متد ( ) clean _ up محلی مناسب برای عملیاتی همچون قطع ارتباط با بانک اطلاعاتی و هر منبع دیگری می‌باشد (از آن جهت به این حالت عنوان تخریب داده‌ایم که شئ موردنظر تمام امکانات اخذ شده حین دو حالت دیگر را از دست می‌دهد و دیگر چیزی برایش باقی نمی‌ماند). به‌عنوان یک جمع‌بندی می‌توان چنین گفت که دوره حیات هر شئ در این چارچوب از سه مرحله تشکیل شده است: ۱- شئ موردنظر نمونه‌گیری شده و مقادیر ابتدایی آن مشخص می‌شود ۲- شئ مذکور کار پردازشی را که جهت انجام آن به‌وجود آمده است انجام داده و به انتها می‌رساند ۳- و درنهایت این شئ تخریب می‌شود؛ چراکه دیگر نیازی به وجود آن نبوده و منابع اخذ شده توسط آن (مانند اتصال بانک اطلاعاتی) باید به سیستم بازگردانده شوند.

برنامه موجود در لیست ۱- ۲۴ آناتومی این کلاس را نشان می‌دهد.

```
1: <?
2: class Page {
3:
4:         function Page( ) {
5:                 session_start();
6:                 global $page_class_sess;
7:                 if ( ! session_is_registered( "page_class_sess" ) ) {
8:                         $page_class_sess = array();
9:                         session_register( "page_class_sess" );
10:                }
11:        }
12:
13:    function init() { }
14:
15:    function main() { }
16:
17:    function clean_up() { }
18: }
19: ?>
```

لیست ۱- ۲۴ ساختار کلاس Page

بدین‌ترتیب، همان‌گونه که در لیست فوق مشاهده می‌کنید، تنها کاری که کلاس مرجع Page انجام می‌دهد ایجاد یک جلسه کاری است که با استفاده از فراخوانی تابع ( ) session _ start در خط ۵ از برنامه انجام می‌شود. ما کد برنامه مربوط به جلسه مذکور را به‌طور مفصل در قسمت دیگری از درس این ساعت مورد بررسی قرار خواهیم داد؛ بنابراین آنچه که تا بدین‌جا در اختیار داریم ساختاری خام به شکل یک اسکلت است که البته می‌توانیم درصورت تمایل اجزای دیگری را به آن اضافه کنیم.

دستیابی به پارامترهای GET و POST

باید اعتراف کرد که PHP امکانات بسیار درخور توجهی را جهت کار با متغیرهای GET و POST به ترتیب از آرایه‌های سیستمی $HTTP_ GET_ VARS و $HTTP_ POST_ VARS در اختیار برنامه‌نویس قرار داده است. با این همه قصد ما این است که پارامترهای مذکور را در قالب خصوصیاتی از یک شئ ذخیره و بازیابی نماییم. برای انجام این کار می‌توانیم کدهای موردنیاز را به‌صورتی که در ادامه می‌بینید، به ساختار متد ( ) init اضافه کنیم:

```
$this → rdata = array _ merge ($GLOBALS [ ' HTTP _ GET_ VARS ' ] ,
    $GLOBALS [ ' HTTP_ POST_ VARS ' ] ;
```

حقیقت این است که در نسخه‌های ابتدایی PHP4 آرایه‌های $HTTP_ GET_ VARS و $HTTP_ POST_ VARS تا زمانی‌که فاقد هرگونه عنصری بودند اصلاً وجود خارجی نداشتند. این وضعیت البته در نسخه‌های جدیدتر PHP4 کاملاً دگرگون شده است به‌گونه‌ای که وجود این دو آرایه منوط به داشتن عنصر نبوده و این آرایه‌ها همواره موجود و در دسترس می‌باشند. این شرایط بدان معنی است که هیچ‌گونه پیش‌شرطی وجود نداشته و نیازی به انجام هیچ‌گونه بررسی در این زمینه نمی‌باشد. تنها کافی است تا این دو آرایه با بهره‌گیری از تابع ( ) array _ merge در یکدیگر ادغام شده و حاصل این عملیات به خصوصیت $rdata از کلاس نسبت داده شود. چنانکه مشاهده می‌کنید ترتیب این ادغام به‌گونه‌ای انتخاب شده است که متغیرهای نوع POST مقدار متغیرهای نوع دیگر، یعنی نوع GET را رونویسی می‌کنند.

استفاده از پیغام

یکی از بخشهای اصلی هر برنامه‌ای این است که ارتباطی منطقی با کاربر خود برقرار کرده و به‌ازای اعمالی که کاربران انجام می‌دهند عکس‌العملهایی را در قالب پیغامهایی به آنها نمایش دهد. چنین بازخوردهایی، مانند پیغامهایی چون "! Welcome " یا "OOPS ! You forgot to fill in the phone field" از جمله پیغامهایی هستند که اکثر کاربران در ارتباطات روزانه خود با برنامه‌های کاربردی با آن مواجه می‌شوند. از این جهت کلاس مرجع Page امکاناتی را به‌منظور تولید یک چنین پیغامهایی به‌همراه خواهد داشت. ضمناً این کلاس حتی از یک سیستم شبه الگو نیز جهت قالب‌بندی و نمایش پیغامهای تولید شده از این طریق بهره‌مند خواهد بود، همچنین این امکان وجود خواهد داشت که پیغامهای مذکور از یک برنامه اسکریپت به برنامه دیگر ارسال شده و در داخل کلاس مقصد (کلاس دریافت‌کننده پیغام) از طریق خصوصیت $rdata دراختیار قرار گیرند. بدین ترتیب ارتباطی از طریق ارسال پیغام مابین برنامه‌ها ایجاد می‌شود. اجازه دهید تا قطعه کدی را که برای کنترل این فرآیند باید به متد ( ) init اضافه شود در اینجا ارائه کنیم:

```
if ( ! empty ($this → rdata [ ' page _ obj _ msg ' ] ) ) {
    $this → message = $this → rdata [ ' page _ obj _ msg ' ] ;
    unset ($this → rdata [ ' page _ obj _ msg ' ] ) ;
}
```

چنانکه در این قطعه کد ملاحظه می‌کنید با بهره‌گیری از ساختار تصمیم‌گیری if وجود پارامتری با عنوان page _ obj _ msg مورد بررسی قرار گرفته است. درصورت وجود این پارامتر گام بعدی عبارت از نسبت دادن مقدار آن به خصوصیت جدیدی با عنوان $message است. گام نهایی نیز مرتب سازی آرایه $rdata به‌واسطه حذف عنصری از آن است که با کلید دستیابی "page _ obj _ msg" مشخص شده است. این عمل با استفاده از تابع ( ) unset انجام می‌شود.

علاوه بر این مایلیم تا کد استفاده‌کننده از این ساختار به طریقی قادر به تنظیم خصوصیت $message باشد. برای انجام این فرآیند به متد دیگری نیاز داریم. ساختار این متد می‌تواند به صورت زیر باشد:

```
Function setMessage ($str) {
    $this → message = $str ;
}
```

همچنین باید به‌طریقی قادر به دستیابی به مقدار خصوصیت $message باشیم. برای انجام این فرآیند نیز احتیاج به متد دیگری داریم که ساختار آن‌را در این‌جا ملاحظه می‌کنید:

```
Function getMessage ( ) {
    return stripslashes ($this → message) ;
}
```

> دو نوع اخیر از متدها که در قالب دو تابع ( ) setMessage و ( ) getMessage در مثال بالا ارائه شده‌اند معمولاً در طراحی کلاسها در ساختار یک برنامه کاربردی شئ‌گرا مورد استفاده فراوان قرار می‌گیرند. متدهای نوع اول از آنجا که وظیفه تنظیم مقادیر خصوصیتهای کلاس را به‌عهده دارند معمولاً با عنوان متدهای setter یا mutator شناخته شده و اغلب اسامی آنها با دنباله کاراکتری "set" همراه است. در مقابل متدهای نوع دوم به‌دلیل وظیفه‌ای که جهت دستیابی به مقادیر خصوصیتهای کلاس دارند معمولاً با عنوان متدهای getter یا accessor شناخته شده و اسامی آنها اغلب با دنباله کاراکتری "get" همراه می‌باشد. درحالی‌که متدهای setter به‌واسطه طبیعت کاری که انجام می‌دهند آرگومان یا آرگومان‌هایی را به‌عنوان ورودی دریافت می‌کنند متدهای getter مگر درحالت خاص فاقد آرگومان ورودی می‌باشند ـ مترجم

و بالاخره در نهایت مایل هستیم تا به نوعی امکان نمایش این پیغام بازخورد طبق یک مکانیزم ساده دراختیار برنامه‌نویس بوده و به‌واسطه‌آن‌هرجاکه‌مایل‌باشدبتواندپیغام موردنظر خود را جهت نمایش

به خروجی ارسال نماید. معمولاً یک چنین فرآیندی از طریق متدی که خود متد ( ) getMessage را فراخوانی می‌کند در اختیار قرار می‌گیرد.

با این وجود در رابطه با استفاده از این کد نکته‌ای را باید درنظر بگیریم و آن این‌است که فرآیند قالب‌بندی پیغام قابل نمایش (در قالبهایی چون جدول، استفاده از حروف bold یا italic، بهره‌گیری از رنگ و غیره) اغلب تنها هنگامی موردنیاز است که پیغامی جهت نمایش در دست باشد. چنانچه هیچ پیغامی درکار نباشد، در مجموع تمایلی به نمایش خروجی نیز وجود نخواهد داشت. باردیگر برنامه‌نویسی که عهده‌دار کنترل عملیات سمت کاربر (لایه نمایش) است خود باید اقدام به بررسی این موضوع نماید (جهت انجام این بررسی وی می‌تواند از یک ساختار تصمیم‌گیری if بهره بگیرد). با این حال طبق آنچه که متداول است ما از راه حلی مشابه الگوها جهت حل این مسأله استفاده می‌کنیم و برای این منظور متدی با عنوان ( ) outputMessage استفاده می‌کنیم. متد ( ) outputMessage چنانکه خواهید دید الگویی را در قالب یک دنباله کاراکتری به‌عنوان آرگومان ورودی دریافت می‌کند و درصورتی که پیغامی جهت ارائه به کاربر وجود داشته باشد، آن پیغام را در قالب الگو مورد استفاده قرار می‌دهد (این پیغام در تابع ( ) str _ replace جایگزین دنباله کاراکتری "%msg %" می‌شود). ساختار این متد به‌ترتیبی است که در ادامه مشاهده می‌کنید:

```
Function outputMessage ($template _ str = " " ) {
   if ( ! empty ($template _ str) )
      $out = str _ replace (" % msg % " , $this → message , $template _ str) ;
   else
      $out = $this → message ;
   print $out ;
}
```

اکنون کلیه قابلیتهای کلاس مرجع Page بدین ترتیب تعریف شده است و وقت آن‌است تا عملکرد آن‌را در قالب برنامه‌ای به بوته آزمایش بگذاریم. اما برای این کار لازم است تا ابتدا کلاس فرزندی را که قابلیتها و عملکردهای کلاس Page را از طریق رابطه توارث به ارث می‌برد ایجاد کرده و سپس نمونه‌گیری لازم را جهت ایجاد شئ و کار با آن انجام دهیم.

بهره‌گیری از یک کلاس فرزند جهت ارزیابی عملکرد کلاس مرجع Page

در برنامه‌ای که در لیست ۲- ۲۴ مشاهده می‌کنید کلاس جدیدی را با عنوان my _ page ایجاد کرده‌ایم. این کلاس به‌گونه‌ای تعریف شده است که فرزند کلاس Page باشد.

```
1: <?php
2: include_once("lib/Page.inc.php");
3:
4: class my_page extends Page {
5:     function my_page() {
6:         Page::Page();
7:     }
8:
9:     function main() {
10:        $this->setMessage("Hello, welcome to Page");
11:    }
12: }
13:
14: $p = new my_page();
15: $p->init();
16: $p->main();
17: $p->clean_up();
18: ?>
19: <html>
20: <head>
21: <title>Testing the Page class</title>
22: </head>
23: <body>
24:
25: <?php
26: $p->outputMessage('
27:     <table cellpadding="5">
28:     <tr><td bgcolor="gray">%msg%</td></tr>
29:     </table>');
30: ?>
31:
32: </body>
33: </html>
```

لیست ۲- ۲۴ ایجاد کلاس فرزندی برای کلاس مرجع Page

توجه کنید که در ساختار کلاس فرزند Page با عنوان my _ page تنها دو متد سازنده و متد ( ) main را مورد رونویسی (تعریف مجدد) قرار داده‌ایم. از آنجا که متد ( ) main در کلاس مرجع Page هیچ کاری را صورت نمی‌دهد، نیازی نمی‌بینیم تا آن‌را در نسخه رونویسی شده در کلاس فرزند مورد فراخوانی قرار دهیم. این فرآیند (فراخوانی متدی از کلاس پدر در نسخه بازنویسی شده از همان متد در کلاس فرزند) عملی است که معمولاً در مورد متدهای سازنده یا constructor و همچنین متدهایی همچون ( ) init انجام می‌دهیم.

ارتباط مابین کلاس فرزند با کلاس پدر خود در زبان PHP (به مانند زبان برنامه‌نویسی Java) از طریق واژه کلیدی خاصی با عنوان extends برقرار می‌شود بدین ترتیب که معرفی کلاس به صورت زیر:

class sub extends super { // ...

موجب می‌شود تا کلاس sub طبق رابطه توارث به‌عنوان کلاس فرزند Super درنظر گرفته شود. نکته مهم دیگر متد خاصی از کلاس است که با عنوان متد سازنده یا Constructor مشخص می‌شود. متد سازنده هر کلاسی همواره همنام با خود آن کلاس است. متد سازنده کلاس فرزند به‌عنوان اولین کار خود متد سازنده کلاس پدر خویش را مورد فراخوانی قرار می‌دهد. این کار از آن جهت لازم است که کلیه اطلاعات مربوط به خصوصیاتی که از پدر به فرزند به ارث می‌رسد تنها در اختیار کلاس پدر بوده و از دید سایر کلاسها، از جمله کلاس فرزند پنهان است (این پنهان کاری تداعی‌کننده اصل اول برنامه‌نویسی به شیوه شیئ‌گرا یعنی کپسوله‌سازی یا Encapsulation است. مطابق این اصل هر کلاسی باید اطلاعات خود را از دید سایر کلاسها پنهان کرده و درنهایت از طریق ارائه متدهایی امکان دسترسی محدود به آنها را در اختیار آن کلاسها قرار دهد). معمولاً رونویسی متد کلاس پدر در کلاس فرزند به‌همراه فراخوانی متد پدر از طریق همان متد در کلاس فرزند انجام می‌پذیرد. اما این قاعده عمومیت و ضرورت ندارد ـ مترجم

چنانکه در نسخه بازنویسی شده از متد ( ) main در کلاس فرزند مشاهده می‌کنید تمام کاری که در درون این متد انجام می‌شود، فراخوانی متد ( ) setMessage با یک دنباله کاراکتری آزمایشی است (خط ۱۰ از برنامه را ببینید). درخارج از کلاس فرزند page _ my و در داخل برنامه اصلی با بهره‌گیری از واژه کلیدی new اقدام به نمونه‌گیری (ایجاد شئ) از کلاس page _ my شده است (خط ۱۴ را ببینید). درخطوط ۱۵ تا ۱۷ از برنامه به ترتیب متدهای ( ) init، ( ) main و ( ) clean _ up جهت پیاده‌سازی سه فاز مقداردهی اولیه، اجرا (پردازش) و بالاخره تخریب فراخوانی شده‌اند. در داخل بدنه سند HTML متد موسوم به ( ) outputMessage با یک دنباله کاراکتری که نقش الگو را به‌عهده دارد فراخوانی شده است (خط ۲۶ از برنامه را ببینید). این دنباله کاراکتری به‌دلیل اینکه پیغامی جهت نمایش در دست می‌باشد، به‌عنوان خروجی به نمایش درخواهد آمد. بدون وجود یک چنین پیغامی دنباله کاراکتری مذکور هرگز بر روی صفحه به نمایش در نمی‌آید.

پشتیبانی از جلسات

ساختار متد سازنده کلاس Page شاید این تفکر را در ذهن شما قوت داده باشد که ما در پی پشتیبانی قابلیت استفاده از جلسات هستیم. از آنجا که قصد ما در درس این ساعت این است که ابزاری

را جهت استحکام و افزایش قابلیت اعتماد محیط برنامه‌نویسی‌مان ایجاد کنیم، لذا پشتیبانی از جلسات را می‌توان یکی از موارد اصولی دانست.

چنانکه‌ازبرنامه‌موجوددر لیست۱- ۲۴ به یاد دارید، ما فرآیند پشتیبانی از جلسات را با تعریف جلسه جدید درمتد سازنده کلاس مرجع Page با بهره‌گیری از تابع ( ) session_ start آغاز کردیم و سپس با استفاده از تابع ( ) session _ register، متغیری سراسری با عنوان $page _ class _ sess را با این جلسه جدید همراه نمودیم. تمام متغییرهایی از جلسه که شئ مورد نظر از آنها استفاده می‌کند در قالب آرایه‌ای با نام $page _ class _ sess ذخیره خواهند شد. استفاده از یک چنین آرایه‌ای برای نگهداری متغیرهای جلسه موجب خواهد شد تا داده‌های مربوط به این جلسات را از هر متغیر جلسه‌ای که خارج از کلاس مورد بحث باشد، جدا نگه داریم. اهمیت یک چنین جداسازی اکنون باید بر شما آشکار شده باشد.

درحال حاضر می‌توانیم چند متد مختلف را به‌منظور تسهیل در فرآیند تنظیم داده‌های جلسات و دستیابی به آنها ایجاد نماییم. تعریف سه نمونه از این توابع را می‌توانید در لیست ۳- ۲۴ مشاهده کنید.

```
function forget_session ( ) {
        global $page_class_sess;
        $page_class_sess = array ( ) ;
}
function set_session_var ($name, $val) {
        global $page_class_sess;
        $page_class_sess[$name] = $val; }
function get_session_var ( $name) {
        global $page_class_sess;
        return $page_class_sess[$name];
}
```

لیست ۳-۲۴

اطمینان از ارسال فرم

اگر از آن دسته برنامه‌نویسانی هستید که از یک برنامه اسکریپت واحد هم برای ارائه فرم و هم برای پردازش آن استفاده می‌کنند، ضروری است تا به طریقی از این مطلب که فرم مورد نظرتان به سرور ارسال شده است، اطمینان پیدا کنید. از آنجا که این فرآیند بسیار متداول می‌باشد، لذا بهتر آن‌است تا توابعی را که یک چنین امکانی را در اختیارمان قرار می‌دهند به‌عنوان متدهایی از کلاس مرجع Page تعریف نماییم. لیست ۴- ۲۴ تعریف دو متد در این زمینه را نشان می‌دهد.

```
        function fflag ($val) {
                return "<input type= \"hidden\" name=\"fflag\"
value=\"$val\">n";
```

```
}
function checkfflag ($val) {
        return (isset( $this->rdata['fflag'] ) &&
                $this->rdata['fflag'] = = $val );
}
```

لیست ۴-۲۴

چنانکه در لیست فوق مشاهده می‌کنید، متد ( ) fflag به‌عنوان تنها آرگومان ورودی یک دنباله کاراکتری را پذیرفته و به‌عنوان مقدار بازگشتی حاصل از عملیات خود یک فیلد مخفی (hidden) را که نام آن مشابه نام متد، یعنی fflag و مقدار آن نیز برابر با دنباله کاراکتری ورودی به متد است به برنامه فراخواننده بازمی‌گرداند؛ همچنین متد دیگر، یعنی ( ) checkfflag نیز مقداری را به‌عنوان آرگومان ورودی اخذ می‌کند. عملی که متد مذکور انجام می‌دهد این است که مطمئن شود آیا پارامترهای موردنظر شامل عنصری با عنوان ' fflag ' با مقداری برابر با آرگومان ورودی به متد می‌باشد یا خیر.

در برنامه‌ای که لیست ۵-۲۴ نشان می‌دهد همان‌گونه که ملاحظه می‌کنید، کلاس فرزند my _ page را جهت استفاده از این توابع به‌منظور اطمینان از ارسال فرم توسعه داده‌ایم.

```
1: <?php
2: include_once("lib/Page.inc.php");
3:
4: class my_page extends Page {
5:     function my_page() {
6:         Page::Page();
7:     }
8:
9:     function main() {
10:        if ( $this->checkfflag("subbed") )
11:            $this->setMessage("FORM SUBMITTED");
12:        else
13:            $this->setMessage("Hello, welcome to Page");
14:
15:    }
16: }
17:
18: $p = new my_page();
19: $p->init();
20: $p->main();
21: $p->clean_up();
22: ?>
23: <html>
24: <head>
25: <title>Testing the Page class</title>
26: </head>
27: <body>
28:
29: <?php
30: $p->outputMessage('
31:     <table cellpadding="5">
32:     <tr><td bgcolor="gray">%msg%</td></tr>
33:     </table>');
34: ?>
35: <form>
36: <?php print $p->fflag("subbed") ?>
37: Type into this box and hit return<br>
38: <input type="text" name="input">
39: </body>
40: </html>
```

لیست ۵- ۲۴ بررسی ارسال فرم

چنانکه در این لیست ملاحظه می‌کنید، متد ( ) fflag در خط ۳۶ برنامه مورد فراخوانی قرار گرفته است. این فراخوانی به نوبه خود منجر به تولید یک فیلد مخفی می‌شود. در متد ( ) main از کلاس my _ page می‌توانیم آزمایشی را جهت اطمینان از وجود پارامتری با عنوان fflag انجام دهیم. این کار با فراخوانی متد دیگری با نام ( ) checkfflag در خط ۱۰ از برنامه انجام گرفته است. چنانچه فرم مورد نظر ارسال شده باشد، ترتیبی می‌دهد تا پیغام بازخورد دستخوش تغییر شود (خط ۱۱ را ببینید).

تغییر مسیر

یکی از متداول‌ترین فرآیندهایی که در توسعه برنامه‌های کاربردی وب انجام می‌شود فرآیندی موسوم به تغییر مسیر یا redirection است. فرآیند مذکور به‌قدری متداول است که کلاس مرجع Page به‌عنوان یکی از سرویسها یا متدهای خود باید از آن پشتیبانی به‌عمل آورد. پیاده‌سازی یک چنین قابلیتی در عمل بسیار ساده می‌باشد. با این حال ما ترجیح می‌دهیم این پیاده‌سازی ساده را با توسعه پشتیبانی خود از متغیر message$، یعنی یکی از خصوصیات کلاس Page اندکی پیچیده‌تر کنیم. لیست ۶- ۲۴ کد مربوط به پیاده‌سازی این قابلیت را در قالب متدی با نام ( ) redirect نشان می‌دهد.

```
fuction redirect ($page, $msg=" " ) {
        if ( ! empty ($msg) )
              $this->message = $msg;
        $this->clean_up ( ) ;
        if ( ! strstr ($page, "?" ) )
              header ("location: $page?page_obj_msg=" .urlencode ($this-
➥>message) );
        else
              header ("location: $page&page_obj_msg=" .urlencode ($this-
➥>message) );
        exit;
    }
```

لیست ۶- ۲۴ پیاده‌سازی قابلیت تغییر مسیر

همان‌گونه که در تعریف ساختار متد ( ) redirect در لیست فوق مشاهده می‌کنید، لازم است تا از یک آدرس URL به‌عنوان آرگومان ورودی این تابع استفاده نماییم. این آدرس در قالب متغیری با نام page$ که حاوی دنباله‌ای از کاراکترها خواهد بود به متد ( ) redirect ارسال می‌شود. متد مورد بحث علاوه بر این آرگومان ضروری قادر است تا آرگومان دیگری را نیز به‌عنوان یک آرگومان اختیاری بپذیرد. این آرگومان اختیاری در تعریف متد ( ) redirect از لیست ۶- ۲۴ با عنوان msg$ مشخص شده است. چنانچه این آرگومان که مقدار آن از نوع دنباله کاراکتری خواهد بود، خالی نباشد، مقدار ذخیره شده در آن به خصوصیتی با عنوان message$ از کلاس Page نسبت داده می‌شود. پیش از تغییر مسیر مرورگر اینترنت به آدرس مورد نظر، چنانچه ملاحظه می‌کنید ابتدا متد ( ) clean _ up به منظور اطمینان از صحت عملیات فرآیند مورد بحث فراخوانی می‌شود. با این عمل، زمینه برای تغییر مسیر مهیا می‌گردد. برای انجام این کار ابتدا وجود علامت " ? " در متغیر page$، یعنی متغیر حاوی آدرس تغییر مسیر با بهره‌گیری از یک ساختار تصمیم‌گیری از نوع if مورد بررسی قرار می‌گیرد. درصورتی که دنباله کاراکتری ذخیره شده در متغیر page$ حاوی علامت موردنظر باشد، می‌توان چنین نتیجه‌گیری کرد که آدرس URL مورد بحث حاوی یک دنباله پرس و جو می‌باشد. در چنین حالتی با بهره‌گیری از

علامت & مقدار ذخیره شده در متغیر $message به انتهای این آدرس ضمیمه می‌گردد. درصورتی‌که علامت مذکور در متغیر $page موجود نباشد باید ترتیبی داد تا این علامت در قالب آدرس، مورد استفاده قرار بگیرد. برای این منظور خودمان اقدام به درج علامت ? خواهیم کرد.

بدین ترتیب چارچوب اصلی ما جهت توسعه برنامه‌های کاربردی وب با استفاده از زبان برنامه‌نویسی PHP آماده بهره‌برداری خواهد بود. با وجودی که این چارچوب از قابلیتهای مختلفی جهت توسعه برخوردار است، اما به‌دلیل ساختار قابل بسطی که دارد به‌راحتی می‌توان نمونه‌هایی از کلاس‌های، فرزند این کلاس مرجع را جهت برخورداری از این قابلیتها مورد استفاده قرار داد. در قسمت بعدی از درس این ساعت مواردی را که مستلزم انجام یک چنین نمونه‌گیری می‌باشد، مورد بررسی قرار می‌دهیم.

برنامه موجود در لیست ۷- ۲۴ شامل کد کاملی از کلاس مرجع Page است.

```
<?
class Page {
    var $rdata;
    var $message;

    function Page( ) {
        session_start();
        global $page_class_sess;
        if ( ! session_is_registered( "page_class_sess" ) ) {
            $page_class_sess = array();
            session_register( "page_class_sess" );
        }
    }

    function init() {
        $this->set_session_var("lastclick", time());
        $this->rdata = array_merge( $GLOBALS['HTTP_GET_VARS'],
➥$GLOBALS['HTTP_POST_VARS'] );
        if ( ! empty( $this->rdata['page_obj_msg'] )) {
            $this->message = $this->rdata['page_obj_msg'];
            unset( $this->rdata['page_obj_msg'] );
        }
    }

    function main() { }

    function clean_up() { }

    function forget_session() {
        global $page_class_sess;
        $page_class_sess = array();
```

```
31:     }
32:
33:     function set_session_var( $name, $val ) {
34:         global $page_class_sess;
35:         $page_class_sess[$name] = $val;
36:     }
37:
38:     function get_session_var( $name ) {
39:         global $page_class_sess;
40:         return $page_class_sess[$name];
41:     }
42:
43:
44:     function fflag( $val ) {
45:         return "<input type=\"hidden\" name=\"fflag\" value=\"$val\">\n";
46:     }
47:
48:     function checkfflag( $val ) {
49:         return ( isset( $this->rdata['fflag'] ) && $this->rdata['fflag'] ==
➥$val );
50:     }
51:
52:     function redirect( $page, $msg="" ) {
53:         if ( ! empty( $msg ) )
54:             $this->message = $msg;
55:         $this->clean_up();
56:         if ( ! strstr( $page, "?" ))
57:             header("Location: $page?page_obj_msg=".urlencode($this-
➥>message));
58:         else
59:             header("Location: $page&page_obj_msg=".urlencode($this-
➥>message));
60:     }
61:
62:     function setMessage( $str ) {
63:         $this->message = $str;
64:     }
65:
66:     function getMessage( ) {
67:         return stripslashes( $this->message );
68:     }
69:
70:     function outputMessage( $template_str = "" ) {
71:         if ( ! empty( $template_str ) )
72:             $out = str_replace( "%msg%", $this->message, $template_str );
73:         else
74:             $out = $this->message;
75:         print $out;
76:     }
77: }
78: ?>
```

لیست ۷- ۲۴ تعریف کامل کلاس مرجع Page

توسعه کلاس مرجع Page

کلاس Page به‌گونه‌ای طراحی شده است که به‌راحتی قابل توسعه باشد. در این قسمت از درس این ساعت به‌واسطه توسعه این کلاس مرجع درصدد این هستیم تا قابلیتها و امکانات دیگری را علاوه بر موارد موجود، جهت ایجاد یک محیط امن محافظت شده با کلمه عبور به این مجموعه اضافه نماییم. به‌ویژه مایلیم تا با بهره‌گیری از قابلیتهای کلاس مرجع Page در رابطه با کنترل و پشتیبانی از جلسات اقدام به ارزیابی کلمه عبور و نام کاربردی با توجه به مقادیر مربوطه از یک بانک اطلاعاتی نماییم.علاوه بر این خیال داریم تا یک سیستم کنترل دسترسی کارآمد را پیاده‌سازی کنیم. برنامه‌نویسی که عهده‌دار پیاده‌سازی لایه نمایش برنامه کاربردی است، می‌تواند انواع صفحات و نیز انواع کاربران وب سایت را با توجه به یک دسته‌بندی مبنا مشخص نماید. به‌واسطه یک چنین دسته‌بندی‌ای تنها کاربرانی از یک نوع خاص قادر به مشاهده صفحات و اسناد مربوطه خواهند بود.

بهره‌گیری‌از یک چنین سیستم کنترل دسترسی در محیطهایی که چندین کاربر مختلف به‌طور هم‌زمان از سیستم استفاده می‌کنند، بسیار متداول است. در این گونه سیستمها که به سیستمهای multi _ user یا چند کاربره معروف هستند کاربران تنها به بخشهایی از سیستم که مربوط به شغل آنهاست، دسترسی خواهند داشت (یک چنین محدودیتی در این گونه سیستمها کاملاً منطقی است، چرا که به عنوان مثال، هیچ دلیلی وجود ندارد که یک کارمند بخش خدمات از یک شرکت به اطلاعات مربوط به بخش مالی و حسابداری شرکت مذکور دسترسی داشته باشد).

سیستمی که درصدد ایجاد آن هستیم به‌راحتی امکان ایجاد کلاسهای فرزند از یک کلاس پدر را در اختیارمان قرار می‌دهد. نام این کلاس را Access می‌گذاریم.

کلاس Access فرزند کلاس مرجع Page است. متد سازنده این کلاس مسیر و نام بانک اطلاعاتی موردنظر را در قالب یک دنباله کاراکتری دریافت می‌کند و آن‌را در یکی از خصوصیات با نام $user _ file ذخیره می‌نماید. تعریف این متد سازنده چنین است:

```
function Access ($user _ file) {
   $this → user _ file = $user _ file ;
   Page : : Page ( ) ;
}
```

همان‌گونه که ملاحظه می‌کنید متد سازنده کلاس Access به‌عنوان بخشی از عملیات خود، متد سازنده کلاس پدر خود یعنی ( ) Page را فرامی‌خواند. اکنون این کلاس تمامی قابلیتهای کلاس پدر خود را به‌واسطه رابطه توارث به ارث برده است. وقت آن‌است تا ساختار این کلاس را اندکی توسعه دهیم.

تعیین بخشهای قابل دسترس از یک سایت

کلاس Access به‌گونه‌ای طراحی می‌شود که به‌طور کاملاً قابل انعطافی با هر نوع سایتی کار کند. از این‌رو قصد ما این نیست که اطلاعات مربوط به بخشهای مختلف سایت را در قالب برنامه‌نویسی دردرون‌این‌کلاس‌پیاده‌سازی‌کنیم.در عوض آنچه که به‌دنبالش هستیم، ایجاد متدی است که برنامه‌نویس را قادر به تعریف مجموعه‌ای از بخشهای قابل دسترس از سایت می‌سازد. در این راه به‌ازای هر بخش موردنظر اقدام به نگهداری یک برچسب و یک عدد نموده و در قالب خصوصیتی از کلاس با نام $access _ type که یک آرایه است، ذخیره خواهیم کرد.

اجازه دهید تا در این قسمت به تعریف متدهای لازم جهت پشتیبانی از این روش بپردازیم. اولین متد مورد بررسی، متدی است با نام ( ) addAccessType که کد مربوط به آن را در لیست ۸- ۲۴ ارائه کرده‌ایم.

```
function addAccessType ($label)  {
        if ( ! empty ( $this->access_types[$label]  )  )
                return false;
$this->access_types[$label] = $this->type_pointer
      $this->type_pointer <<= 1;
      return true;
}
```

لیست ۸- ۲۴ متد ( ) addAccessType

چنانکه در این لیست مشاهده می‌کنید، در راه تعیین نوع دسترسی یک برچسب ویژه ایجاد خواهد شد. برای این منظور عدد جاری موجود جهت نسبت دادن به انواع دسترسی‌ها را در قالب خصوصیتی از کلاس Access با عنوان $type_ pointer ذخیره می‌کنیم. پس از عمل نسبت دهی با بهره‌گیری از عملگر انتقال، یک مقدار ذخیره شده در خصوصیت $type_ pointer را در قالب یک عدد باینری به اندازه یک بیت به سمت چپ انتقال می‌دهیم (کلیه اطلاعات و از آن جمله مقادیر عددی در قالب مجموعه‌هایی از صفر و یک در درون کامپیوتر ذخیره می‌شوند). عمل انتقال به چپ به اندازه یک واحد در دنیای برنامه‌نویسی سطح پایین (زبان اسمبلی) به مفهوم ضریب عدد موردنظر در عدد ۲ می‌باشد. از دیدگاه باینری این بدان معنی می‌باشد که به ازای هر نوع دستیابی عدد مربوطه دو برابر می‌شود. چند نمونه متوالی از انواع دسترسی‌ها می‌تواند به این شکل باشد:

00001
00010
00100
01000
10000

این روش نوید یک شیوه قابل انعطاف را می‌دهد، اما توجه به این نکته ظریف و با اهمیت ضروری است که تنها اعداد محدودی را می‌توان از این طریق مورد استفاده قرار داد. با روشی که توضیح داده شد تنها می‌توانیم حداکثر تا تعداد ۳۱ نوع دسترسی را مشخص نماییم. با این همه این تعداد معمولاً پاسخگوی بیشتر کاربردها می‌باشد به‌ویژه اگر متوجه باشیم که کاربران مختلف ترکیبات مختلفی از این ۳۱ نوع دسترسی را مورد استفاده قرار می‌دهند.

علاوه بر امکان تعریف انواع دسترسی‌ها توسط برنامه‌نویس یا طراح لایه نمایش، باید امکاناتی را نیز جهت اطلاع از اعدادی که به این انواع دسترسی‌ها اختصاص داده‌ایم، در اختیارشان قرار دهیم. در این مورد یک متد getter یا accessor می‌تواند مفید واقع شود. تعریف این متد می‌تواند به صورتی که در لیست ۹- ۲۴ مشاهده می‌کنید، باشد.

```
function getAccessType( $label ) {
      if( empty( $this -> access_types[ $label ] ))
            return false;

      return $this -> access_types[ $label ];
}
```

لیست ۹- ۲۴ تعریف متد ( ) getAccessType جهت دستیابی به کد دسترسی

همان‌گونه که در لیست فوق مشاهده می‌کنید، متد ( ) getAccessType جهت انجام عملیات موردنظر به یک آرگومان ورودی نیاز دارد. این آرگومان که از نوع دنباله کاراکتری است، در واقع مشخص‌کننده برچسب مربوط به نوع دسترسی موردنظر است. چنانچه آرایه $access_ types که در قالب خصوصیتی از کلاس Access ذخیره می‌شود شامل مقدار عددی متناظر با این نوع دسترسی باشد، متد مورد بحث آن مقدار عددی را به‌عنوان کد دسترسی معادل به برنامه فراخواننده باز می‌گرداند. در غیر این‌صورت متد فوق مقدار false را باز خواهد گرداند.

دو تابع (متد) ( ) addAccessType و ( ) getAccessType جهت تنظیم نوع دسترسی و همچنین دستیابی به آن کاملاً کفایت می‌کنند. با این حال با کمی دقت می‌توان جای خالی متد دیگری را نیز در همین رابطه مشاهده نمود. در حقیقت در این سیستم به روشی نیاز داریم تا به‌واسطه آن کتابخانه را از این مطلب که برنامه اسکریپت جاری مشمول کدام انواع دسترسی موجود است، مطلع کنیم. همچنین برای پیاده‌سازی روشی از یک متد دیگر با عنوان ( ) setPageType استفاده می‌کنیم. تعریف متد مذکور را می‌توانید در لیست ۱۰- ۲۴ مشاهده نمایید.

```
function setPageType( $label ) {
      if( empty( $this -> access_types[ $label ] ) )
            return false;

      $this -> page_type = $this -> access_types[ $label ];
```

```
        return true;
}
```

لیست ۱۰- ۲۴ تعریف متد ( ) setPageType

چنانکه در این لیست مشاهده می‌کنید، متد ( ) setPageType برچسبی را در قالب یک دنباله کاراکتری به عنوان آرگومان ورودی دریافت می‌کند. در صورتی که این برچسب در آرایه $access_ Types (که خصوصیتی از کلاس Access است) موجود باشد، خصوصیت دیگری از این کلاس با عنوان $page _ type با مقدار متناظر با این برچسب تنظیم می‌گردد.

با در دست داشتن این متدها اکنون در مرحله‌ای از کار هستیم که می‌توانیم در قالب متد ( ) init از یک کلاس مجموعه‌ای از انواع دسترسی‌ها را به واسطه فراخوانی متد ( ) addAccessType تعریف کنیم. بدین ترتیب برنامه‌های اسکریپت مختلف می‌توانند با بهره‌گیری از متد ( ) setPageType اقدام به تعیین انواع دسترسی موردنظر خود نمایند.

درج و دستیابی به اطلاعات کاربران

اکنون که توانستیم انواع دسترسی به سایت را مشخص کنیم به روشی نیاز داریم تا به کمک آن بتوانیم اطلاعات مربوط به کاربران سایت را دریافت کرده به‌گونه‌ای که قادر به تشخیص هر یک از دیگری باشیم.

برای این مثال تصمیم داریم تا از یک بانک اطلاعاتی DBA جهت ثبت اطلاعات کاربران استفاده نماییم. در این بانک اطلاعاتی مواردی همچون نام کاربری (یک دنباله کاراکتری منحصر به‌فرد جهت تشخیص کاربران از یکدیگر)، کلمه عبور، و نوع کاربر را ذخیره خواهیم کرد. نوع کاربر مقداری عددی است که ما آن‌را با انواع دسترسی به صفحات مقایسه خواهیم کرد. این مقایسه از آن جهت صورت می‌گیرد که محدودیت دسترسی کاربر موردنظر به صفحات و اسناد مختلف سایت مشخص شود. متدهایی از کلاس Access را که امکان دستیابی به اطلاعات کاربران سایت و همچنین تغییر آنها را در اختیار قرار می‌دهند، به‌سادگی می‌توان توسط یک کلاس فرزند جهت دستیابی به یک بانک اطلاعاتی دیگر رونویسی (تعریف مجدد) نمود.

اولین متدی که در رابطه با تنظیم و دستیابی اطلاعات کاربران مورد بررسی قرار خواهیم داد متدی با عنوان ( ) user _ add است. تعریف این متد در لیست ۱۱- ۲۴ ارایه شده است.

```
function add_user( $user, $pass, $type ) {
      $res = dba_open( $this -> user_file, "c", "gdbm" ) or
                        die( "no user db" );

      if( dba_fetch( $user, $res ) ) {
            dba_close( $res );
            return false;
      }
```

```
        $add_array = array( 'pass' => $ pass, 'type' => $type
);
        dba_replace( $user, serialize( $add_array ), $res );
        dba_close( $res );
}
```

لیست ۱۱- ۲۴ تعریف متد ( ) user _ add جهت درج اطلاعات کاربران

این متد همان‌گونه که در لیست فوق مشاهده می‌کنید دو دنباله کاراکتری و یک مقدار عددی را به‌عنوان آرگومان‌های ورودی می‌پذیرد. آرگومان‌های اول و دوم متد مورد بحث به ترتیب نام کاربردی و کلمه عبور را مشخص می‌کنند. آرگومان سوم نیز نوع کاربر را به‌منظور تعیین محدودیتهای دسترسی به اسناد مختلف موجود در سایت مشخص می‌کند. این متد از دنباله کاراکتری تعیین شده توسط متد سازنده کلاس مربوطه به‌عنوان مسیر بانک اطلاعاتی موردنظر جهت بازکردن آن بانک اطلاعاتی استفاده می‌کند. درصورتی که مشخصاتی منطبق با نام کاربری ورودی به این متد به‌عنوان آرگومان در بانک اطلاعاتی موجود نباشد، متد ( ) user _ add اقدام به درج اطلاعات این کار به‌عنوان یک کاربر جدید در بانک اطلاعاتی خواهد کرد. نام کاربری فیلدی است که در بانک اطلاعاتی از آن به‌عنوان کلید اصلی یا Primary Key استفاده می‌شود و به همین علت نیز باید کاملاً منحصر به فرد باشد (لازم به ذکر است که کلید اصلی در داخل جدول از بانک اطلاعاتی معنی پیدا می‌کند. اما از آن‌جا که بانکهای اطلاعاتی DBA از یک جدول برای نگهداری داده‌ها استفاده می‌کنند به‌کارگیری اصطلاح کلید اصلی در مورد هر بانک اطلاعاتی از این نوع نیز دارای معنی می‌باشد. کلید اصلی هر جدول از یک بانک اطلاعاتی، فیلدی ازآن بانک است که اطلاعات هر سطر یا هر رکورد از جدول را به‌گونه‌ای منحصر به فرد مشخص می‌کند). سایر داده‌های ورودی نیز به‌ترتیب به بانک اطلاعاتی موردنظر اضافه می‌شوند.

دستیابی به اطلاعات ذخیره شده هر یک از کاربران در بانک اطلاعاتی مورد استفاده فرآیند بسیار ساده‌ای است. این فرآیند که توسط متدی با عنوان ( ) get _ user _ data انجام می‌شود در لیست ۱۲- ۲۴ قابل بررسی است.

```
function get_user_data( $user ) {
        $res = dba_open( $this -> user_file, "c", "gdbm" ) or
                        die( "no user db" );

        $user_data = dba_fetch( $user, $res );
        dba_close( $res );
        if( !$user_data )
                return false;
        return unserialize( $user_data );
}
```

لیست ۱۲- ۲۴ تعریف متد ( ) get _ user _ data

همان‌گونه که در تعریف این متد در لیست مذکور مشاهده می‌کنید متد get _ user _ data ( ) ابتدا سعی می‌کند تا به بانک اطلاعاتی موردنظر متصل شده و سپس با بهره‌گیری از نام کاربری اطلاعات مربوطه را مورد بازیابی قرار دهد. چنانچه این متد فرآیند مذکور را با موفقیت انجام دهد، اطلاعات مورد درخواست را به برنامه فراخواننده باز می‌گرداند. در غیر این‌صورت متد مذکور مقدار false را به برنامه فراخواننده بازخواهد گرداند.

تحمیل کنترل دسترسی

اکنون که امکان تعریف و درج اطلاعات کاربران در بانک اطلاعاتی و همچنین مجال تعیین نوع دسترسی‌کاربران و امکان دستیابی به آنها فراهم شده است به مکانیزمی جهت کنترل دسترسی کاربران نیازمندیم.

برای این منظور از متد خاصی با نام access _ control ( ) که در واقع قلب کلاس Access را تشکیل می‌دهد، نیاز داریم. کد مربوط به این متد را می‌توانید در لیست ۱۳- ۲۴ مشاهده کنید.

```
function access_control() {
      if( $this -> page_type == 0 )
            return true;
      $user = $this -> get_session_var( "user" );
      $pass = $this -> get_session_var( "pass" );

      if( empty( $user ) || empty( $pass )
            $this -> bump( "Not enough user data" );

      $user_array = $this -> get_user_data( $user );

      if( !$user_array )
            $this -> bump( "Unknown user" );

      if( $pass != $user_array[ 'pass' ] )
            $this -> bump( "Incorrect password" );

      if( !$this -> codeAllowed( $user_array[ 'type' ] ) )
            $this -> bump( "Access to this resource
forbidden" );
      return true;
}
```

لیست ۱۳- ۲۴ متد اصلی کلاس Access با عنوان access _ control ( )

چنانکه در تعریف این متد ملاحظه می‌کنید درصورتی که خصوصیت page _ type$ شامل هیچ مقداری نباشد، می‌توان چنین فرض کرد که سند مربوطه کاملاً در دسترس تمامی کاربران قرار دارد.

اطلاعات مربوط به نام کاربری و کلمه عبور با بهره‌گیری از متد ( ) get_ session_ var و ارسال مورد درخواستی به آن تأمین می‌شود. اگر هیچ یک از این اطلاعات در دسترس نباشد، می‌توان چنین نتیجه گرفت که اطلاعات مربوطه از بانک حذف شده و یا پیشتر چنین اطلاعاتی وارد بانک نشده است. به‌هر حال در هریک از این دو حالت با فراخوانی تابع ( ) bump پیغام خطایی مبنی بر فقدان اطلاعات کافی در مورد کاربر مورد نظر بر روی صفحه نقش می‌بندد. اما درصورتی که همه چیز مطابق میل پیش برود و اطلاعات مربوط به نام کاربری مورد بحث و کلمه عبور مربوطه در بانک اطلاعاتی موجود باشند، می‌توان با فراخوانی متد ( ) get _ user _ data و ارسال متغیر $user به آن به‌عنوان آرگومان ورودی اطلاعات مربوط به این کاربر را از بانک اطلاعاتی استخراج نمود (متغیر $user شامل نام کاربری می‌باشد). چنانچه هیچ‌گونه اطلاعاتی به‌واسطه فراخوانی متد ( ) get _ user _ data به برنامه فراخواننده که‌دراینجا متد ( ) access _ control است بازگردانده نشود، می‌توان چنین نتیجه‌گیری کرد که اطلاعات مربوط به کاربر مورد نظر در بانک اطلاعاتی موجود نبوده و بدین سان باردیگر به‌واسطه فراخوانی متد ( ) bump پیغام خطایی روانه صفحه نمایش خواهد شد. در گام بعدی کلمه عبور بازیابی شده از بانک اطلاعاتی با دنباله کاراکتری موجود در متغیر $pass که حاوی کلمه عبور واردشده توسط کاربر است، مورد مقایسه قرار می‌گیرد. درصورتی که این دو مقدار یکسان نباشند از ورود کاربر موردنظر ممانعت به‌عمل می‌آید. درنهایت چنانچه نام کاربری و کلمه عبور وارد شده صحیح باشند، یک بررسی در مورد امکان دستیابی کاربر موردنظر به سند مربوطه انجام خواهد شد. این بررسی بر مبنای عددی که بیانگر نوع کاربر است، صورت می‌پذیرد. متدی که این ارزیابی را انجام می‌دهد ( ) codeAllowed نام داشته و چنین تعریف می‌شود:

```
Function codeAllowed ($num) {
   return ($this → page _ type & $num) ;
}
```

همان‌گونه که در تعریف این تابع مشاهده می‌کنید، از عملگر ویژه‌ای با علامت & استفاده شده است. این عملگر که به عملگر "binary and" معروف است جهت مقایسه مقادیر عددی در مبنای دودویی یا باینری مورد استفاده قرار می‌گیرد. در این مورد مقایسه باینری مابین بیتهای مقدار ذخیره شده در متغیر $page _ type که بیانگر نوع دسترسی است و مقدار آرگومان ورودی به متد یعنی $num صورت می‌پذیرد. در مقایسه بیتی، به این ترتیب هر بیت (که تنها می‌تواند شامل یکی از مقادیر صفر یا یک باشد) از یک متغیر با بیت متناظر خود در متغیر دیگر مورد استفاده قرار می‌گیرد. از آن‌جا که عملگر مقایسه‌ای در اینجا یعنی & ترکیب AND این بیت‌ها را در نظر می‌گیرد، چنانچه هر دو بیت مورد مقایسه یک باشند بیت خروجی متناظر نیز برابر با یک درنظر گرفته می‌شود و بقیه حالات که شامل هر دو صفر یا فقط یکی صفر است، معادل با صفر درنظر گرفته خواهد شد. در چنین شرایطی متد ( ) codeAllowed تنها درصورتی یک مقدار مثبت را به برنامه فراخواننده باز می‌گرداند که نوعی

هم‌پوشانی ما بین مقدار آرگومان $num و خصوصیت $page _ type وجود داشته باشد (منظور از هم‌پوشانی عدم یکسان بودن بیتها است). لازم به توضیح است که بیت (با تلفظ bit)، کوچک‌ترین واحد ذخیره‌سازی اطلاعات در کامپیوترها بوده و همواره شامل یکی از دو مقدار صفر یا یک است.

در صورتی که عدد ورودی تعیین‌کننده نوع کاربر با نوع صفحه مطابقت نکند متد ( ) codeAllowed مقدار صفر را به برنامه فراخواننده بازگردانده و بدین ترتیب موجبات اجرای متد ( ) bump باردیگر فراهم می‌آید. تعریف متد ( ) bump به‌صورت زیر است:

```
Function bump ($msg) {
   $this → redirect ( $this  → login _ page , $msg) ;
}
```

همان‌گونه که از تعریف این متد بر می‌آید، فراخوانی متد ( ) bump در داخل متدی از کلاس فرزند صرفاً موجب فراخوانی متد دیگری با نام ( ) redirect از کلاس پدر خواهد شد. واضح است که در فراخوانی متد ( ) redirect خصوصیت $login _ page مطابق پیش‌فرض با دنباله کاراکتری "login . php" تنظیم شده است که البته این تنظیم اجباری نبوده و می‌توان از هر سند مناسب دیگری به‌جای آن نیز استفاده کرد.

کلاس Access در یک نگاه

اکنون کتابخانه موردنیاز کامل شده و کلیه متدها و قابلیتهای مورد نیاز تحت یک کلاس واحد پیاده‌سازی شده‌اند. لیست ۱۴- ۲۴ کد کاملی از کلاس Access را نشان می‌دهد.

```
see lib/Access.inc.php
```

```
<?php
include_once("Page.inc.php");

class Access extends Page {
    var $access_types = array();
    var $type_pointer = 1;
    var $login_page = "login.php";
    var $page_type = 0;

    function Access( $user_file ) {
        $this->user_file = $user_file;
        Page::Page();
    }

    function addAccessType( $label ) {
        if ( ! empty( $this->access_types[$label] ) )
            return false;
        $this->access_types[$label] = $this->type_pointer;
        $this->type_pointer <<= 1;
        return true;
    }

    function setPageType( $label ) {
        if ( empty( $this->access_types[$label] ) )
            return false;

        $this->page_type = $this->access_types[$label];
        return true;
    }

    function getAccessType( $label ) {
        if ( empty( $this->access_types[$label] ) )
            return false;
        return $this->access_types[$label];
    }

    function codeAllowed( $num ) {
        return ( $this->page_type & $num );
    }

    function set_login_page( $login ) {
        $this->login_page = $login;
    }

    function bump( $msg ) {
        $this->redirect($this->login_page, $msg );
        exit;
    }

        function init() {
                Page::init();
        }

    function access_control() {
        if ( $this->page_type == 0 )
            return true;
```

```
57:         $user = $this->get_session_var( "user" );
58:         $pass = $this->get_session_var( "pass" );
59:         if ( empty( $user ) || empty( $pass ) )
60:             $this->bump("Not enough user data");
61:         $user_array = $this->get_user_data( $user );
62:         if ( ! $user_array )
63:             $this->bump("Unknown user");
64:         if ( $pass != $user_array['pass'] )
65:             $this->bump("Incorrect password");
66:         if ( ! $this->codeAllowed( $user_array['type'] ) )
67:             $this->bump("Access to this resource forbidden");
68:         return true;
69:     }
70:
71:     function remove_user( $user ) {
72:         $res = dba_open( $this->user_file, "c", "gdbm" ) or die("no user
➥db");
73:         $val = dba_delete( $user, $res );
74:         db_close( $res );
75:     }
76:
77:     function add_user( $user, $pass, $type ) {
78:         $res = dba_open( $this->user_file, "c", "gdbm" ) or die("no user
➥db");
79:         if ( dba_fetch( $user, $res ) ) {
80:             dba_close( $res );
81:             return false;
82:         }
83:         $add_array = array( 'pass'=>$pass, 'type'=>$type );
84:         dba_replace( $user, serialize( $add_array ), $res );
85:         dba_close( $res );
86:     }
87:
88:     function get_user_data( $user ) {
89:         $res = dba_open( $this->user_file, "c", "gdbm" ) or die("no user
➥db");
90:         $user_data = dba_fetch( $user, $res );
91:         dba_close( $res );
92:         if ( ! $user_data )
93:             return false;
94:         return unserialize( $user_data );
95:     }
96: }
```

لیست ۱۴- ۲۴ تعریف کامل کلاس Access

اکنون با در دست داشتن این کد کتابخانه‌ای (کد قابل استفاده مجدد یا کدی که می‌توان آن‌را بارها و بارها در برنامه‌های مختلف مورد استفاده قرار داد)، توانایی‌های بسیاری در اختیار ما قرار گرفته است. اما جهت تحصیل آنچه که موردنظر ماست، لازم است تا کدهای کتابخانه‌ای دیگری را نیز توسعه دهیم. قسمت بعدی به پیاده‌سازی کلاس دیگری اختصاص دارد.

پیاده سازی کلاس (چارچوب) مرجعی برای توسعه پروژه‌ها

درصورتی که انواع کاربران و انواع صفحات به‌طور محسوسی با یکدیگر قابل تطبیق باشند، باید آنها را در یک موقعیت واحد قرار داد. این استراتژی فرآیند بازیابی صفحات را افزایش خواهد داد. در برنامه موجود در لیست ۱۵- ۲۴ قصد ما این است که کلاسی ساده را توسعه دهیم که می‌تواند به‌عنوان الگویی برای تعدادی از صفحات در یک پروژه برنامه کاربردی وب مورد استفاده واقع شود. این کلاس مستقیماً توسط برخی از برنامه‌ها مورد نمونه‌گیری واقع خواهد شد. همچنین ممکن است در برخی دیگر از برنامه‌ها به‌عنوان کلاس پدر مورد استفاده قرار بگیرد. با این فلسفه ما نام آن‌را ProjectBase انتخاب می‌کنیم.

```
see lib/ProjectBase.inc.php
```

```
1: <?php
2: include_once("Access.inc.php");
3:
4: class ProjectBase extends Access {
5:         var $db_file =  "users/user_dir";
6:         var $freelance_user;
7:         var $admin_user;
8:         var $super_super;
9:
10:        function ProjectBase( ) {
11:                Access::Access( $this->db_file );
12:        }
13:
14:        function init() {
15:                Access::init();
16:                $this->addAccessType( "freelance" );
17:                $this->addAccessType( "admin" );
18:                $this->addAccessType( "superuser" );
19:                $this->freelance_user  = ( $this-
➥>getAccessType("freelance"));
20:                $this->admin_user   = ( $this->getAccessType("freelance") |
                                           $this->getAccessType("admin"));
21:
22:                $this->super_user   = ( $this->getAccessType("freelance") |
23:                                           $this->getAccessType("admin") |
24:                                           $this->getAccessType("superuser"));
25:        }
26: }
27: ?>
```

لیست ۱۵- ۲۴ تعریف کلاس ProjectBase

همان‌گونه که مشاهده می‌کنید، کلاس ( ) ProjectBase ابتکار عمل کمی از خود به‌خرج می‌دهد؛ به این معنی که تمام کاری که این کلاس انجام می‌دهد عبارت از معرفی کلاس Access به‌عنوان کلاس پدر و تعریف چند نوع کاربر و چند نوع دسترسی است. این کلاس درحقیقت سه نوع مختلف از دسترسی را با عناوین "freelance"، "admin" و "superuser" به ترتیب در خطوط ۱۶، ۱۷ و

۱۸ از لیست مذکور معرفی می‌کند. این کلاس همچنین متناسب با این سه نوع دسترسی سه نوع کاربر را نیز معرفی می‌کند که در خطوط ۱۹ تا ۲۲ از لیست مورد بحث در قالب خصوصیاتی از کلاس ProjectBase ذخیره می‌شوند. نوع کاربری که با عنوان $freelance_ user تعریف شده است تنها قادر به دستیابی صفحاتی با نوع دسترسی "freelance" خواهد بود. همچنین نوع کاربری که با عنوان $admin_ user مشخص شده است تنها قادر به دستیابی صفحاتی با نوع دسترسی "freelance" یا "admin" خواهد بود. نوع کاربری که با عنوان $super _ user تعریف شده است قادر به دستیابی کلیه صفحات‌موجوددرسایت‌می‌باشد.به‌عبارت‌دیگر این نوع کاربر می‌تواند علاوه بر دستیابی به صفحاتی با نوع دسترسی "freelance" و "admin" صفحات دیگری را که نوع دسترسی آنها به‌صورت"Superuser" تعریف شده نیز مورد دستیابی قرار دهد. تنظیم این سه نوع حقوق دسترسی همان‌گونه که مشاهده می‌کنید با استفاده از عملگر ویژه‌ای با عنوان " binary or" صورت گرفته است. این عملگر بیتهای متناظر از عملوندهای خود را به‌گونه‌ای ترکیب می‌کند که قوانین مربوط به عملیات OR منطقی رعایت شوند. یعنی تنها درصورتی حاصل ترکیب دو بیت برابر با صفر خواهد بود که هر دو بیت ورودی صفر بوده باشند. در سایر موارد، یعنی هر دو یک و تنها یکی از بیتهای صفر، حاصل برابر با یک خواهد بود. طی این قاعده، برای مثال اگر نوع دسترسی "freelance" را با عدد بانیری 1 و نوع دسترسی "admin" را با عدد بانیری 10 مشخص کنیم، حاصل عملیات مورد بحث یعنی "binary or" برابر با عدد باینری 11 خواهد بود.

استفاده از چند کاربر آزمایشی جهت ارزیابی عملکرد سیستم

جهت ارزیابی عملکرد سیستم و اطمینان از صحت آن لازم است تا چند کاربر مختلف را با انواع دسترسی‌های گوناگون ایجاد نماییم. برنامه موجود در لیست ۱۶- ۲۴ این فرآیند را نشان می‌دهد.

```
<?php
include("ProjectBase.inc.php");

$p = new ProjectBase( );
$p->init();
$p->add_user( "matt", "pass", $p->super_user );
$p->add_user( "bob",  "pass", $p->admin_user );
$p->add_user( "mary", "pass", $p->freelance_user );
?>
done
```

لیست ۱۶- ۲۴ افزودن کاربران به سیستم

همان‌گونه که در این لیست مشاهده می‌کنید، اکنون در فاز پیاده‌سازی هستیم. این فرآیند با توجه به کتابخانه‌ای که پیشتر با دقت و موشکافی آن‌را ایجاد کردیم کار بسیار سخت و طاقت‌فرسایی نیست. جهت ارزیابی این کتابخانه و اطمینان از آنچه که انتظار داریم، سه نوع کاربر مختلف با سه نوع

دسترسی متفاوت ایجاد کرده‌ایم. کلمات عبور در این مثال به‌گونه‌ای بسیار ساده که به‌راحتی به‌خاطر سپرده می‌شوند. انتخاب شده‌اند درصورتی‌که قصد ما پیاده‌سازی کامل یک کتابخانه بود، باید تسهیلات و امکاناتی را برای کاربر نوع "superuser" جهت حذف و اضافه کردن کاربران مختلف فراهم می‌کردیم. با این همه فرض می‌کنیم که به چنین قابلیتی نیاز نداریم، هرچند که در عمل این فرض صحیح نیست.

ایجاد تسهیلاتی جهت اتصال یا login

حال که فرآیند ایجاد سیستم موردنظرمان به اتمام رسیده و آماده بهره‌برداری است و همچنین اطلاعات چندین کاربر از انواع مختلف در بانک اطلاعاتی در دست است، می‌توانیم امکانی را جهت اتصال یا login کاربر در اختیار وی قرار دهیم. برنامه موجود در لیست ۱۷- ۲۴ کد مربوط به فرم درخواست کلمه عبور و همچنین پردازش آن‌را نشان می‌دهد.

see example/login.php

```
1: <?php
2: include("ProjectBase.inc.php");
3:
4: class MyPage extends ProjectBase {
5:     function MyPage() {
6:         ProjectBase::ProjectBase();
7:     }
8:     function main() {
9:         if ( $this->checkfflag("subbed") ) {
10:            $user = $this->rdata['user'];
11:            $pass = $this->rdata['pass'];
12:            $user_array = $this->get_user_data( $user, $pass);
13:            if ( ! $user_array )
14:                $this->setMessage("unknown user");
15:            elseif ( $pass != $user_array['pass'] )
16:                $this->setMessage("Incorrect password");
17:            else {
18:                $this->set_session_var("user",  $user );
19:                $this->set_session_var("pass",  $pass );
20:                $this->setMessage("Welcome to the system");
21:                $this->redirect( "welcome.php" );
22:            }
23:        }
24:    }
25: }
26:
27: $p = new MyPage( );
28: $p->init();
29: $p->main();
30: $p->clean_up();
31: ?>
32: <html>
33: <head>
34: <title>login</title>
35: </head>
```

```
36: <body>
37:
38: <?php
39: $p->outputMessage('
40:     <table cellpadding="5">
41:     <tr><td bgcolor="gray">%msg%</td></tr>
42:     </table>');
43: ?>
44:
45: <form>
46:
47: <?php print $p->fflag("subbed") ?>
48: user<br>
49: <input type="text" name="user"><br>
50: pass<br>
51: <input type="password" name="pass"><br>
52: <input type="submit" value="go">
53:
54: </form>
55: </body>
56: </html>
```

لیست ۱۷- ۲۴ کد مربوط به فرآیند اتصال به سیستم یا login

همان‌گونه که در این لیست ملاحظه می‌کنید، در خط ۴۷ از برنامه از متد ( ) fflag و در خط ۹ نیز از متد ( ) checkfflag جهت اطمینان از ارسال فرم login به سرور استفاده کرده‌ایم. همچنین متد ( ) outputMessage را در خط ۳۹ از برنامه به‌منظور قالب‌بندی پیغام بازخورد فراخوانی نموده‌ایم. با این وجود بار اصلی این برنامه به دو روش متدی با نام ( ) main است. درصورتی که فرم مورد بحث ارسال شده باشد، متد ( ) get _ user _ data در خط ۱۲ برنامه جهت بررسی وجود کاربران مورد فراخوانی قرار می‌گیرد و در غیر این‌صورت متد ( ) $setMessag در خط ۱۴ جهت اطلاع کاربر از این موضوع فراخوانی می‌شود.

با فرض اینکه نام کاربر موردنظر در بانک اطلاعاتی موجود باشد، بررسی کلمه عبور مربوطه در خط ۱۵ از برنامه انجام می‌شود. چنانچه کلمه عبور نیز صحیح باشد، کاربر موردنظر می‌تواند به محیط محافظت شده ما قدم بگذارد. این فرآیند با فراخوانی متد ( ) set_ session_ var جهت تنظیم دو متغیر جلسه " user " و " pass " در خطوط ۱۸ و ۱۹ از برنامه انجام می‌شود. از مقادیر این متغیرها در تمامی مدت دستیابی کاربران به اسناد موجود در سایت جهت تعیین حقوق دسترسی استفاده می‌شود.

صفحات محافظت شده

در تمام صفحاتی که در مقابل دستیابی‌های غیر مجاز محافظت می‌شوند، خصوصیت $page_type با فراخوانی متد تعیین نوع صفحه یعنی ( ) setPageType تنظیم می‌شود. همچنین در این‌گونه صفحات متد ( ) access _ control به طور حتم مورد فراخوانی واقع می‌شود. محافظت مینیمال در مورد یک صفحه نمونه در کد لیست ۱۸- ۲۴ قابل بررسی است.

```
<?php
include("ProjectBase.inc.php");
$p = new ProjectBase( );
$p->init();
$p->setPageType("freelance");
$p->access_control();
$p->main();
$p->clean_up();
?>
<html>
<head>
<title>freelance</title>
</head>
<body>
<?php
$p->outputMessage('
        <table cellpadding="5">
        <tr><td bgcolor="gray">%msg%</td></tr>
        </table>');
?>
This is a frelance page<p>

<a href="welcome.php">freelance only</a><br>
<a href="admin_only.php">admin only</a><br>
<a href="super_only.php">super only</a>
</body>
</html>
```

لیست ۱۸- ۲۴ کد مربوط به کمترین محافظت ممکن از یک صفحه از سایت

جهت تعیین اینکه کدام نوع از کاربران می‌توانند به چه صفحاتی از سایت دسترسی داشته باشند، تنها لازم است تا از آرگومان مناسبی هنگام فراخوانی متد ( ) setPageType استفاده کنید.

موارد تکمیل نشده

در درس این ساعت با اینکه فرآیند توسعه کتابخانه مفیدی را شرح دادیم مواردی را می‌توانیم برشماریم که آنها بررسی نکرده و یا به‌طور کامل مورد بررسی قرار ندادیم. در این درس اشاره‌ای به توسعه لایه نمایش نداشتیم؛ همچنین نمی‌توانیم ادعا کنیم که کتابخانه توسعه داده شده و کدهایی که از آن استفاده می‌کنند، کاملاً جامع بودند. چنانچه قصد استفاده از کدهای کتابخانه‌ای توسعه داده شده در درس این ساعت را دارید، لازم است تا چند نکته ای را در ذهن داشته باشید.

فراموش‌نکنیدکه بنا به پیش‌فرض، توابع پشتیبانی از جلسات فایل‌های مربوطه را در فهرستهایی می‌نویسند که عموم می‌توانند آنها را مورد دستیابی قرار دهند. همواره باید اطلاعات مربوط به جلسات را در محل امنی بنویسید.

هیچ مکانیزمی را جهت پایان دادن عادی به جلسات و همچنین هیچ روشی را جهت پایان دادن به جلسات بازدید به‌واسطه وجود تأخیری طولانی مابین دو درخواست متوالی کاربر جهت دریافت سند موردنظر توسعه ندادیم. این دو حالت از مواردی هستند که در برنامه‌های واقعی باید آنها را درنظر داشته باشید.

همچنین کلیه کلمات عبور را درقالب فایل متنی ساده‌ای ذخیره کردیم. چنانچه عامل امنیت برای شما از اهمیت بالایی برخوردار باشد، لازم است تا از تابع ( ) crypt جهت حفاظت از کلمات عبور در بانک اطلاعاتی استفاده نمایید.

به منظور افزایش حفاظ امنیتی می‌توانید از روش مضاعفی برای بالا بردن امنیت دسترسی استفاده نمایید. در این شیوه ابتدا باید آدرس IP کاربر را در اولین ارتباط وی با سایت ثبت کرده و در دسترسی‌های‌بعدی‌او این آدرس را مجدداً مورد بررسی قرار دهید. بدین ترتیب از راهزنی‌هایی که به واسطه قطع و وصل مجدد جلسات ممکن است پیش آید، جلوگیری نموده‌اید.

جمع بندی

در درس این ساعت سعی ما بر این بود تا یک کتابخانه غنی از قابلیتهایی که ممکن است در آینده مورد نیاز باشد، به دست آوریم. در راه توسعه این کتابخانه چنانکه شاهد بودید بسیاری از تکنیکهایی را که طی دروس ساعات گذشته فرا گرفتیم، بارها و بارها مورد استفاده قرار دادیم. مواردی چون کلاسها، اشیا، جلسات و توابع DBA از این قبیل بودند.

علاوه بر این، در درس این ساعت تکنیکهای جدیدی را نیز جهت تعیین اعتبار کاربران و کنترل دسترسی مورد بحث قرار دادیم.

امیدواریم که از مطالعه این کتاب، به همان میزان که مؤلف از نوشتن آن لذت برده است. بهره و لذت کافی را برده باشید.

پرسش و پاسخ

پرسش: چگونه می‌توان علاوه بر مطالعه این کتاب دانش خود را درباره PHP توسعه داد؟

پاسخ: با اتمام این کتاب کار بزرگی انجام دادید.این کتاب شامل اطلاعاتی کافی در مورد نحوه ایجاد برنامه‌هاو محیطهای قابل توسعه می‌باشد. با داشتن این اطلاعات در ذهن و انبوهی از اطلاعات در همین زمینه بر روی اینترنت، تنها تصورتان است که شما را محدود می‌کند ! اگر احساس می‌کنید که این کتاب نقطه آغاز مناسبی برای شما بوده است به احتمال قوی کتابهای مناسب دیگری خواهند

توانست اطلاعات جامع‌تری در اختیارتان قرار دهند. به‌ویژه اگر مایل باشید، می‌توانید کتابهای زیر را که از جمله کتابهای بسیار خوب در این زمینه می‌باشند، تهیه کرده و مورد مطالعه قرار دهید:

- The PHP Developer's Cookbook - by Sterling Hughes
- PHP and MySQL Web Development - by Luke Welling and Laura Thomson

تمرینها

هدف از این بخش ارائه تمرینهایی در قالب آزمون می‌باشد. پاسخ آزمون بلافاصله پس از هر آزمون آمده است. بخش فعالیتها شامل تمرینهایی است که به منظور افزایش قابلیتها و مهارتهای برنامه‌نویسی خواننده طراحی شده و البته فاقد پاسخ است.

آزمون

۱- از کدام تابع می‌توان جهت ترکیب عناصر دو آرایه با یکدیگر استفاده نمود؟
۲- کدام آرایه سیستمی شامل پارامترهای درخواستی از نوع GET می‌باشد؟
۳- چگونه می‌توان در مورد این مطلب که آیا متغیری از برنامه مقداردهی شده است یا خیر، اطمینان حاصل نمود؟
۴- از کدام تابع می‌توان به‌منظور ثبت متغیری به‌همراه یک جلسه استفاده کرد؟
۵- از کدام تابع می‌توان به‌منظور دستیابی به فیلدی از یک بانک اطلاعاتی DBA استفاده کرد؟

پاسخ آزمون

۱- به کمک تابع ( ) array _ merge می‌توان عناصر دو آرایه را با یکدیگر ترکیب کرد.
۲- آرایه سیستمی $HTTP_ GET_ VARS شامل پارامترهای درخواست نوع GET می‌باشد.
۳- با بهره‌گیری از تابعی با عنوان ( ) isset و ارسال نام متغیری به آن می‌توان از این مطلب که آیا متغیر مذکور مقداردهی شده است یا خیر اطلاع حاصل کرد.
۴- به کمک تابعی با عنوان ( ) session _ register می‌توان متغیری را به‌همراه یک جلسه بازدید ثبت نمود.
۵- با استفاده از تابع ( ) dba_ fetch می‌توان به فیلدی از یک بانک اطلاعاتی DBA دسترسی پیدا کرد.

فعالیتها

۱- برنامه‌های ارائه شده در درس این ساعت را مجدداً مورد بازبینی قرار دهید. آیا در میان این برنامه‌ها تکنیک یا مسأله خاصی به چشم می‌خورد که شامل حال پروژه برنامه نویسی شما نیز باشد؟

۲- کلاس Access ارائه شده در درس این ساعت را به‌گونه‌ای توسعه دهید که شامل ویژگی "پایان دادن به جلسات درصورت عدم ارسال درخواست توسط کاربر پس از سپری شدن یک مدت زمان معین" باشد (این ویژگی معمولاً با عنوان session timout شناخته می‌شود). سطح دیگری از امنیت را به سیستم موجود اضافه کنید که آدرس IP کاربر را طی درخواستهای مختلف از یک جلسه بازدید مورد بررسی قرار دهد.

۳- بار دیگر کتاب را از ابتدا تا انتها مورد بازبینی مجدد قرار داده و طی آن نکاتی را که به‌عنوان مطلب مهم علامت‌گذاری کرده‌اید، دوره کنید. درصورتی که کتاب حاضر را تحت عنوان یک دوره آموزشی زبان برنامه‌نویسی PHP مورد استفاده قرار می‌دهید به خاطر داشته باشید که باید نکات فوق را چندین مرتبه بازبینی کنید تا به نتیجه مطلوب برسید.

ضمیمه

فرهنگ واژه‌ها و اصطلاحات تخصصی

فرهنگ واژه‌ها و اصطلاحات تخصصی

anonymous function : تابع بی‌نام ـ تابعی است که حین اجرای یک برنامه اسکریپت ایجاد شده و در قالب متغیری ذخیره می‌شود و یا خود به تابع دیگری ارسال می‌گردد.

argument : آرگومان ـ مقداری است که به‌عنوان ورودی به یک تابع هنگام فراخوانی آن تابع ارسال می‌شود. آرگومان‌های یک تابع، همواره در داخل یک جفت پرانتز به تابع موردنظر ارسال می‌شوند. هنگام تعریف یک تابع اسامی آرگومان‌ها در قالب لیستی که عناصر آن با علامت کاما از یکدیگر جدا شده‌اند در درون جفت پرانتز مشخص می‌شوند. این آرگومان‌ها به‌عنوان متغیرهای محلی در دسترس تابع قرار می‌گیرند.

array : آرایه ـ لیستی از متغیرهاست. آرایه خود متغیری است که شامل عناصری شاخص‌گذاری شده (با عدد یا با دنباله‌ای از کاراکترها) می‌باشد. با بهره‌گیری از آرایه‌ها می‌توان چندین مقدار مختلف را تحت یک نام واحد ذخیره و بازیابی کرده و یا پردازش دیگری روی آنها انجام داد.

associative array : آرایه انجمنی ـ آرایه‌ای است که شاخصهای آن دنباله‌های کاراکتری هستند.

Atom : اتم ـ در قالب عبارت منظم، اتم به الگویی اطلاق می‌شود که در درون یک جفت پرانتز واقع شده باشد (معمولاً به آن الگوی فرعی یا subpattern نیز گفته می‌شود). پس از تعریف اتم موردنظر می‌توان آن‌را به‌منزله یک کاراکتر ساده و یا یک کلاس کاراکتری فرض کرد.

Boolean : نوع داده منطقی ـ یکی از انواع داده‌ها در زبان برنامه‌نویسی PHP است. متغیرهایی با این نوع داده تنها می‌توانند شامل یکی از دو مقدار true یا false باشند.

Bounds : محدودیت تطبیق : عددی است که تعداد دفعاتی را که یک کاراکتر یا مجموعه‌ای از کاراکترهای مشخص باید در یک عبارت منظم مورد تطبیق واقع شوند، مشخص می‌کند.

break Statement : عبارت break ـ عبارتی است که شامل واژه کلیدی break می‌باشد. این عبارت موجب توقف آنی ساختار تکرار for یا while و خروج از آن شده و هیچ حلقه دیگری مورد پردازش قرار نمی‌گیرد.

cast : تغییر نوع داده ـ فرآیندی است که طی آن نوعی از داده‌ها به نوع دیگر تبدیل می‌شود.

Class : کلاس ـ مجموعه‌ای از توابع (که در قالب کلاس با عنوان متد شناخته می‌شوند) و متغیرها (که در قالب کلاس با عنوان خصوصیت شناخته می‌شوند). تعریف کلاس با واژه کلیدی class انجام می‌شود. کلاسها در واقع الگوهایی برای ایجاد اشیا می‌باشند.

color resource : مرجع رنگ ـ مقدار ویژه‌ای از نوع داده "resource" است. این مقدار ویژه توسط تابعی با نام ( ) imagecolorallocate به برنامه فراخواننده بازگردانده می‌شود. مقدار بازگشتی از این تابع را بدین ترتیب می‌توان در اختیار توابع دیگری که قابلیت استفاده از رنگها را دارند، قرار داد.

comment : توضیح ـ متنی در یک برنامه است که توسط مفسر زبان برنامه‌نویسی مربوطه نادیده گرفته می‌شود. از توضیحات معمولاً در مواردی چون افزایش خوانایی برنامه استفاده می‌شود.

constant : مقادیر ثابت ـ مقادیری هستند که توسط تابع ( ) difine در برنامه تعریف می‌شوند. این مقادیر در طول اجرای برنامه تغییر نکرده و ثابت باقی می‌مانند. مقادیر ثابت دارای حوزه سراسری هستند. تنها از مقادیر عددی و دنباله‌های کاراکتری می‌توان به‌عنوان مقادیر ثابت استفاده کرد.

continue statement : عبارت continue ـ عبارتی است که در آن از واژه کلیدی continue استفاده شده باشد. این عبارت موجب می‌شود تا ساختار تکرار for یا while مربوطه بلافاصله و بدون درنگ اجرای حلقه جاری را متوقف کرده و عملیات را به‌ازای حلقه بعدی از سر بگیرد. این فرآیند با ارزیابی مجدد عبارت شرطی و از سرگیری عملیات موردنظر ساختار (در صورتی که عبارت شرطی مذکور به صورت true ارزیابی شود) انجام می‌گیرد (در ساختار تکرار نوع for پیش از ارزیابی عبارت شرطی مورد بحث عبارت کاهش یا افزایش اجرا می‌شود).

conversion specification : مشخصات تبدیل ـ بخشی از یک دنباله کاراکتری است که فرآیند کنترل قالب‌بندی را به‌عهده دارد. هر مشخصه تبدیل با یک علامت درصد (%) آغاز شده و چگونگی قالب‌بندی آرگومان مربوطه در ساختار فراخوانی توابع ( ) printf و ( ) sprintf را تعریف می‌کند. بهره‌گیری از مشخصه‌های تبدیل مشمول هیچ‌گونه محدودیتی نبوده و به هر تعداد دلخواه می‌توان از مشخصه‌های تبدیل در دنباله‌های کاراکتری کنترل‌کننده شیوه قالب‌بندی استفاده کرد؛ به‌شرطی که از تعداد متناظری آرگومان در تابع ( ) printf یا ( ) sprintf بهره گرفت.

Cookie : کوکی ـ حجم کوچکی از داده‌هاست که به‌واسطه درخواست سندی از یک وب سرور یا یک برنامه اسکریپت بر روی کامپیوتر درخواست کننده ذخیره می‌شود.

data type : نوع‌داده‌ها ـ انواع مختلفی از داده‌ها که ذخیره‌سازی آنها بر روی کامپیوتر مستلزم صرف‌حافظه‌به میزانهای مختلف است. همچنین فرآیند کار بر روی آنها به شیوه‌های مختلفی از یکدیگر صورت می‌پذیرد. یک نوع داده نام طبقه‌ای از یک سیستم دسته‌بندی نوع داده‌ها و اطلاعات ذخیره شده بر روی کامپیوتر است. زبان برنامه‌نویسی PHP از هشت نوع داده مختلف با عناوین زیر پشتیبانی به‌عمل می‌آورد: integer، double، string، Boolean، object، array، resource و NULL.

DBA : لایه مفهومی بانک اطلاعاتی ـ مجموعه‌ای از توابع ویژه‌ای است که جهت بهره‌برداری از طیف گسترده‌ای از سیستمهای بانک اطلاعاتی مبتنی بر فایل تدارک دیده شده‌اند.

DBA resource : مقدار ویژه‌ای از نوع داده resource ـ این مقدار ویژه توسط تابع dba _ open ( ) به برنامه فراخواننده بازگردانده می‌شود. از این مقدار ویژه سپس در توابع مختلف DBA که جهت بهره‌برداری از بانکهای اطلاعاتی تدارک دیده شده‌اند می‌توان استفاده کرد.

DBM : مدیر بانک اطلاعاتی ـ قابلیتی است که سیستمهای شبه DBM و شبه DBA به‌واسطه آن امکان ذخیره، پردازش و بازیابی عناصری در قالب یک زوج نام و مقدار را در اختیارقرار می‌دهند.

DOM : مدل پردازش اسناد که مبتنی بر اشیاست ـ شیوه‌ای از دستیابی به اسناد XML است که مستلزم پیمایش درختی از گره‌ها که بر مبنای سلسله مراتب پدر ـ فرزند توسعه یافته‌اند می‌باشد.

Double : نوع داده double ـ نوعی از داده‌های پشتیبانی شده در زبان برنامه‌نویسی PHP است که با عنوان float نیز شناخته شده و می‌توان از آن جهت ذخیره یک عدد اعشاری استفاده کرد. طبق آنچه که در دیکشنری کامپیوتر آمده است " هر عددی که متشکل از یک بخش مانتیس (بخشی که بعد از نقطه اعشار واقع می‌شود)، یک توان و یک بخش پایه باشد که یک double محسوب می‌شود." برای منظوری که ما در این کتاب به‌دنبال آن هستیم، می‌توان یک عدد double را به‌عنوان عددی فرض کرد که شامل بخشی از یک عدد صحیح می‌باشد. به‌عبارت دیگر، عددی که شامل یک نقطه اعشار باشد.

DTD : توصیف نوع سند یا Document Type Definition ـ مجموعه‌ای از قوانین که ساختار یک سند از نوع XML را به‌طور کامل توصیف می‌کند. برای مثال مشخص می‌کند که از کدام نشانه‌ها و بر مبنای چه ترتیبی در سند XML موردنظر استفاده شده است. برنامه‌ای با عنوان XML Parser توصیف DTD مربوطه را مورد بازخوانی قرار داده و ساختار سند XML را با مجموعه قوانین موجود در آن تطبیق می‌دهد.

else statement : عبارت else ـ عبارتی است که تنها به‌همراه عبارت if می‌توان از آن استفاده نمود. این عبارت متشکل از واژه کلید else و یک عبارت PHP (یا مجموعه‌ای از عبارات PHP) است. این عبارت تنها هنگامی اجرا خواهد شد که عبارت شرطی موجود در بخش if از یک ساختار تصمیم‌گیری if / else به صورت false ارزیابی گردد.

entity body : بدنه سند ـ بخش اصلی هر سند است که توسط وب سرور به برنامه کلاینت بازگردانده می‌شود. این بخش همچنین ممکن است به‌عنوان بخشی از یک درخواست از نوع POST از جانب برنامه کلاینت به سرور ارسال شود.

Escape : گریز ـ مکانیزمی‌است‌که‌طی آن کاراکترهایی با معنای خاص را می‌توان در یک دنباله کاراکتری یا یک عبارت منظم با قرار دادن علامت \ قبل از آنها در معنای واقعی‌شان مورد استفاده قرار داد.

Expression : عبارت- هرترکیبی از توابع، مقادیر و عملگرها که معادل یک مقدار خاص درنظر گرفته می‌شود. به‌عنوان یک قاعده کلی، عبارت هر آن چیزی است که بتوان از آن به‌جای یک مقدار استفاده کرد.

Field width specifier : شاخص پهنای میدان ـ شاخصی در قالب یک مشخصه تبدیل است که در فضای مورد نیاز جهت قالب‌بندی خروجی را تعیین می‌کند.

file resource : مرجع فایل ـ مقدار خاصی از نوع داده "resource" است که توسط تابع ( )fopen به برنامه فراخواننده بازگردانده می‌شود. از این مقدار سپس می‌توان در سایر توابعی که می‌توانند فایل‌ها را مورد پردازش قرار دهند، بهره‌گرفت.

Float : نوع داده float ـ معادلی است برای نوع داده double.

for statement : عبارت for ـ ساختار تکراری است که بر مبنای مقداردهی اولیه یک متغیر شمارنده (عبارت مقداردهی اولیه)، ارزیابی متغیر شمارنده (عبارت ارزیابی) و تغییر مقدار متغیر شمارنده (عبارت کاهش یا افزایش) تشکیل می‌یابد. ویژگی این عبارت آن‌است که تمام ملزومات ساختار در یک خط از برنامه شکل می‌گیرد. به‌شرطی که عبارت ارزیابی معادل مقدار true باشد، برنامه به اجرای دستورالعملهای بدنه حلقه ادامه می‌دهد.

format control string : دنباله کاراکتری تعیین کننده نوع قالب‌بندی ـ اولین آرگومان از تابع ( )printf یا ( )sprintf است. این آرگومان شامل مشخصه‌های تبدیل است. مشخصه‌های مذکور نحوه چگونگی قالب‌بندی سایر آرگومان‌های این توابع را تعیین می‌کنند.

function : تابع ـ بلوکی از کد که بلافاصله مورد اجرا واقع نشده بلکه اجرای آن مستلزم فراخوانی نام نسبت داده شده به بلوک مذکور است. توابع به دو دسته سیستمی و غیرسیستمی که توسط برنامه‌نویس توسعه داده می‌شوند، قابل تقسیم می‌باشند. برخی از توابع به‌گونه‌ای طراحی می‌شوند که اطلاعاتی را به‌عنوان آرگومان ورودی دریافت می‌کنند. بیشتر توابع مقداری را به‌عنوان نتیجه عملیات به برنامه فراخواننده خود باز می‌گردانند.

GET Request : درخواست نوع GET ـ نوعی از درخواست که برنامه کلاینت به برنامه سرور (وب سرور) ارسال می‌کند. در این نوع درخواست اطلاعات در قالب ضمیمه‌ای از آدرس URL مربوطه به سرور ارسال می‌شود.

global statement : عبارت global ـ ترکیبی است که از واژه کلیدی global که به دنبال آن نام یک یا چند متغیر ذکر می‌شود. این واژه کلیدی موجب می‌شود تا متغیر یا متغیرهای فوق به جای حوزه محلی در یک حوزه سراسری که کل برنامه را شامل می‌شود، قابل دسترسی باشند.

header section : بخش هدر ـ بخشی از یک درخواست یا پاسخ HTTP است که به‌دنبال آن خط درخواست یا خط پاسخ واقع می‌شود. بخش هدر شامل اطلاعاتی در قالب اسامی و مقادیر مربوطه

است به‌گونه‌ای که هر نام و مقدار مربوطه در یک خط از این بخش واقع می‌شود. هر نام از مقدار مربوط به آن توسط یک علامت کولون جدا می‌شود.

htaccess file . : فایل htaccess . ـ سندی است که توسط وب سرور Apache مورد بازخوانی قرار می‌گیرد. این سند شامل گزینه‌هایی جهت پیکربندی و نحوه عملکرد وب سرور است. مدیر وب سرور قادر است تا گزینه‌هایی را که کاربران مجاز به تنظیم مقدار آنها هستند، مشخص نماید. فرآیند تنظیم بر مبنای ساختار فهرستها انجام می‌گیرد. درصورتی که تنظیم مربوطه به‌طور مناسبی انجام شده باشد گزینه‌هایی که از این سند تنظیم شده باشند بر روی فهرست جاری و تمام زیر فهرستهای آن تأثیرگذارخواهند بود. اسناد مورد بحث ممکن است شامل گزینه‌های PHP نیز باشند. چنین گزینه‌هایی با پیشوندهایی چون php _ flag یا php _ value مشخص می‌شوند. از اسناد htaccess . علاوه بر این می‌توان جهت تنظیم گزینه Add _ type نیز استفاده کرد. گزینه مذکور پسوند فایل‌هایی را که باید توسط پردازشگر PHP مورد پردازش قرار بگیرند، مشخص می‌کند.

HTTP : قرار داد انتقال فرامتن یا HyperText Transfer Protocol ـ مجموعه‌ای از قوانین که فرآیند ارسال درخواست توسط برنامه کلاینت و ارسال پاسخ مربوطه توسط سرور را تعریف می‌کند.

if statement : عبارت if ـ عبارتی است که در آن از واژه کلیدی if به منظور ارزیابی یک یا چند عبارت شرطی استفاده می‌شود. دستورالعمل یا دستورالعملهای موجود در بدنه عبارت if تنها درصورتی اجرا می‌شوند که شرط یا شرطهای مورد ارزیابی معادل با مقدار true باشند.

image resource : مرجع تصویر ـ مقدار ویژه‌ای از نوع داده "resource" است. این مقدار ویژه به‌واسطه فراخوانی تابع ( ) imagecreate به برنامه فراخواننده بازگردانده می‌شود. از این مقدار بازگشتی سپس می‌توان در تابعی که عملیات را در مورد پردازش تصاویر انجام می‌دهند، استفاده نمود.

Inheritance : وراثت ـ رابطه ویژه‌ای مابین اشیای موجود در برنامه که در برنامه‌نویسی به شیوه شیئ‌گرا مورد استفاده قرار می‌گیرد. این رابطه بر این موضوع دلالت می‌کند که یک کلاس نمونه می‌تواند شامل متغیرها و متدهای یک کلاس دیگر باشد مشروط بر آنکه چنین رابطه‌ای مابین آن دو کلاس برقرار باشد. رابطه وراثت در تعریف کلاس فرزند مشخص شده و پیاده‌سازی آن در زبان PHP با استفاده از واژه کلیدی extends صورت می‌پذیرد.

Integer : نوع عدد صحیح ـ یکی از انواع داده‌های مجاز در زبان PHP است. متغیری از این نوع داده می‌تواند اعداد صحیح مثبت و منفی و عدد صفر را ذخیره نماید.

iteration : تکرار ـ اجرایی واحد از یک یا چند دستورالعمل موجود در حلقه‌ای از یک ساختار تکرار است. حلقه‌ای که به تعداد پنج مرتبه اجرا می‌شود، شامل پنج مرتبه تکرار است.

Link resource : مرجع پیوند ـ مقدار ویژه‌ای از نوع داده "resource" است. این مقدار ویژه به‌واسطه فراخوانی تابع ( ) mysql _ connect به برنامه فراخواننده بازگردانده می‌شود. از این مقدار بازگشتی سپس می‌توان در سایر توابع MySQL جهت کار با بانک اطلاعاتی مربوطه استفاده کرد.

multidimensional array : آرایه چندبعدی ـ آرایه‌ای است که عناصر آن خود از نوع آرایه هستند.

NULL : پوچ ـ یک نوع داده خاص است. این نوع داده شامل مقداری با عنوان NULL است که بیانگر یک متغیر مقداردهی نشده می‌باشد، یعنی متغیری که شامل هیچ مقداری نیست.

object : شئ ـ نمونه‌ای از یک کلاس است که به عنوان یک موجودیت فیزیکی در حافظه ذخیره می‌شود. به‌عبارت دیگر یک شئ قابلیت نهفته در یک کلاس است. اشیا با استفاده از واژه کلیدی new از کلاسهای مربوطه نمونه‌گیری می‌شوند، بدین صورت که ابتدا واژه کلیدی مذکور و سپس نام کلاس موردنظر جهت نمونه‌گیری ذکر می‌شود. پس از نمونه‌گیری شیئی از یک کلاس می‌توان کلیه خصوصیات و متدهای آن‌را مورد دستیابی قرار داد. Object در زبان برنامه‌نویسی PHP نوعی داده محسوب می‌شود.

operand : عملوند ـ مقداری است که به‌همراه یک عملگر مورد استفاده قرار می‌گیرد. معمولاً به‌ازای هر دو عملوند از یک عملگر استفاده می‌شود.

operator : عملگر ـ یک نشانه یا مجموعه‌ای از نشانه‌های متوالی است که درصورت استفاده به‌همراه مقادیر (عملوندها) منجر به انجام عملی بر روی آنها شده و مقدار جدیدی را تولید می‌کند.

padding specifier :شاخص پرکننده ـ این شاخص در قالب یک مشخصه تبدیل مورد استفاده قرار گرفته و تعداد کاراکترهایی را که خروجی باید به خود اختصاص دهد، مشخص می‌کند این شاخص همچنین کاراکتری را مشخص می‌کند که در صورت عدم تأمین تعداد کاراکترهای لازم در خروجی باید به عنوان کاراکترهای پرکننده مورد استفاده قرار بگیرند.

padding modifier : تغییردهنده رفتار الگو ـ کاراکتری است که پس از آخرین علامت جداکننده در یک عبارت منظم سازگار با perl واقع شده و موجب اصلاح رفتار آن می‌شود.

POST request : درخواست نوع POST ـ درخواستی است که توسط برنامه کلانیت به برنامه سرور ارسال می‌گردد. در این نوع درخواست اطلاعات موردنظر را می‌توان در قالب بدنه درخواست به سرور ارسال نمود.

precision specifier : مشخصه دقت ـ بخشی از یک مشخصه تبدیل است که تعداد ارقام اعشاری مربوط به عددی اعشاری را جهت نمایش در خروجی مشخص می‌کند.

predefined variables : متغیرهای سیستمی ـ متغیرهایی هستند که به‌طور خودکار حین اجرای برنامه مقداردهی شده و توسط موتور PHP در اختیار برنامه اسکریپت قرار می‌گیرند. این‌گونه متغیرها دارای حوزه سراسری بوده و شامل متغیرهای سرور همچون $HTTP_ REFERER می‌شوند.

query string : دنباله پرس و جو ـ مجموعه‌ای از زوجها شامل اسامی و مقادیر مربوطه که به‌عنوان بخشی از یک درخواست نوع GET به انتهای URL ضمیمه می‌شوند. در چنین ترکیبی اسامی و مقادیر مربوطه با بهره‌گیری از علامت = از یکدیگر جدا می‌شوند. همچنین هر زوج با استفاده از علامت & از زوج بعدی جدا می‌شود. دنباله پرس و جو خود توسط علامت ? از آدرس URL جدا می‌گردد. اسامی و مقادیر مربوطه در این ترکیب‌بندی به شیوه خاصی کدگذاری می‌شوند به‌گونه‌ای که کاراکترهایی که معنای خاصی برای سرور دارند، با مقادیر معادل خود جایگزین می‌گردند.

Reference : مرجع ـ مفهومی است که به اشاره چندین متغیر به یک مقدار واحد دلالت دارد. بنا به پیش‌فرض، ارسال آرگومان‌ها به توابع مربوطه و همچنین فرآیند نسبت‌دهی در زبان برنامه‌نویسی PHP از طریق مقدار انجام می‌گیرد. این بدان مفهوم است که در عملیات نامبرده از کپی مقادیر به‌جای خود آنها استفاده می‌شود. در فرآیند ارسال آرگومان‌ها به توابع و همچنین نسبت‌دهی به شیوه مرجع، متغیر جدید به همان مقداری که متغیر اصلی اشاره می‌کند اشاره خواهد داشت. بدین ترتیب هرگونه تغییری در متغیر جدید موجب اعمال همان تغییر در متغیر اصلی خواهد شد.

regular expresion : عبارت منظم ـ یک روش توانمند و بسیار کارآمد جهت ارزیابی و تغییر متون.

request headers : هدرهای درخواست ـ مجموعه‌ای از زوجها شامل اسامی و مقادیر مربوطه است که توسط برنامه کلاینت جهت دراختیار گذاشتن اطلاعاتی درمورد خود برنامه کلانیت و نوع درخواست به برنامه سرور ارسال می‌شود.

request line : خط درخواست ـ اولین خط درخواست ارسالی از جانب برنامه کلانیت به برنامه سرور است. این خط شامل نوع درخواست ارسالی (معمولاً یکی از انواع GET، POST یا HEAD)، آدرس سند مورد نظر و درنهایت نسخه قرار داد HTTP مورد استفاده (HTTP / 1. 0 یا HTTP / 1. 1) می‌باشد.

resource : مرجع ـ یکی از انواع داده‌ها در زبان PHP است. مراجع امکاناتی را جهت دستیابی و کار با موجودیتهای خارجی (خارج از برنامه) همچون بانکهای اطلاعاتی، فایل‌ها و تصاویر گرافیکی در اختیار قرار می‌دهند.

response headers : هدرهای پاسخ ـ مجموعه‌ای از زوجهای شامل اسامی و مقادیر مربوطه که توسط برنامه سرور در پاسخ به درخواست برنامه کلاینت برای آن ارسال می‌شود. این مجموعه شامل اطلاعاتی درباره برنامه سرور و داده‌های ارسالی توسط آن است.

result identifier : مشخصه مجموعه نتایج پرس و جو از بانک اطلاعاتی ـ مراجعه شود به result resource.

result resource : مرجع مجموعه نتایج حاصل از پرس و جو ـ مقدار ویژه‌ای از نوع داده "resource" است. این مقدار به‌واسطه فراخوانی تابع ( ) mysql _ query به برنامه فراخواننده بازگردانده می‌شود. سپس می‌توان از این مقدار بازگشتی در سایر توابع مربوط به MySQL جهت کار بر روی مجموعه نتایج حاصل از پرس و جو از بانک اطلاعاتی استفاده نمود.

server variables : متغیرهای سرور ـ متغیرهای سیستمی که PHP آنها را در اختیار برنامه‌نویس قرار می‌دهد. اینکه کدام متغیرها در دسترس برنامه قرار می‌گیرند، موضوعی است که به وب سرورمورد استفاده بستگی دارد. با این وجود متغیرهای متداولی همچون $HTTP _ USER_ AGENT و $REMOTE _ ADDR همواره در دسترس قرار دارند.

SQL : زبان پرس و جوی ساخت یافته یا Structured Query Language ـ یک دستور زبان استاندار جهت پرس و جو از بانکهای اطلاعاتی مختلف.

Statement : عبارت ـ دستورالعملی است که جهت اجرا به پردازشگر PHP ارسال می‌شود. عبارت در یک برنامه PHP معادل جمله در زبان انگلیسی (یا هر زبان دیگری) است. هر عبارتی از برنامه با یک علامت سمی‌کولون (;) خاتمه پیدا می‌کند (جملات در زبان انگلیسی با یک علامت نقطه پایان می‌پذیرند). عباراتی که سایر عبارات را دربر می‌گیرند و نیز عباراتی که بلوکی از کد را خاتمه می‌دهند از این قاعده مستثنی هستند. در بیشتر موارد عدم استفاده از علامت سمی‌کولون جهت پایان دادن به یک عبارت موجب به اشتباه افتادن پردازشگر PHP و آشکارشدن یک خطا می‌شود.

Static statement : عبارت Static- عبارتی است شامل واژه کلیدی Static که به دنبال آن معرفی یک متغیر و نسبت دهی مقداری به آن انجام شده باشد. از این نوع عبارات در رابطه با توابع استفاده می‌شود. هر تغییری که به متغیر Static داده می‌شود مابین فراخوانی‌های یک تابع ثبت خواهد شد.

Status line : خط وضعیت ـ اولین پاسخ سرور به یک درخواست ارسالی توسط کلاینت است. خط وضعیت شامل نسخه مورد استفاده از قرار داد HTTP توسط برنامه سرور (یکی از دو مورد HTTP/1. 0 یا HTTP/ 1. 1) ، کد مشخصه پاسخ و بالاخره پیغامی است که مفهوم این کد را بیان می‌کند.

String : دنباله کاراکتری ـ یک نوع داده که شامل مجموعه‌ای از کاراکترها می‌باشد.

Subclass : کلاس فرزند ـ کلاسی است که متغیرها و متدهای یک کلاس دیگر موسوم به کلاس پدر را به ارث می‌برد.

ternory operator : عملگر سه‌تایی ـ مقداری را به برنامه باز می‌گرداند که از یکی از دو عبارتی که توسط علامت کولون (:) از یکدیگر جدا شده‌اند، حاصل شده است. اینکه کدام‌یک از این دو عبارت در تولید مقدار بازگشتی مؤثرند به نتیجه یک عبارت شرطی بستگی دارد. این عبارت شرطی پیش از دو عبارت مذکور واقع شده و توسط علامت ? از آنها جدا می‌شود.

timestamp : برچسب زمان ـ تعداد ثانیه‌های سپری شده از نیمه شب اول ماه ژانویه سال ۱۹۷۰ برمبنای مرجع GMT است. از عدد حاصل در محاسبات مربوط به تاریخ استفاده می‌شود.

type specifier : مشخص نوع ـ بخشی از یک مشخص تبدیل است که نوع داده‌ای را که باید در خروجی نمایش داده شود، مشخص می‌کند.

variable : متغیر ـ ظرفی است برای نگهداری مقداری از یک نوع داده خاص که می‌تواند از نوع عددی، دنباله‌ای از کاراکترها، یک شئ خاص، یک آرایه و یا یک مقدار blooean باشد. محتوای متغیرها می‌تواند حین اجرای برنامه تغییر نماید.

while statement : عبارت while ـ یک ساختار تکرار که متشکل از یک عبارت شرطی و یک عبارت (یا مجموعه‌ای از عبارتها) می‌باشد. تا زمانی‌که عبارت شرطی موردنظر به‌صورت true ارزیابی شود عبارت یا عبارتهای موجود در بدنه این ساختار به طور مکرر اجرا خواهد شد.

XML : زبان نشانه‌گذاری قابل توسعه یا Exteusible Markup Language ـ مجموعه‌ای از قوانین است که جهت تعریف و پردازش زبانهای نشانه‌گذاری وضع می‌شود. این‌گونه زبانها معمولاً جهت ایجاد ساختاری از داده‌ها به‌منظور استفاده اشتراکی از آنها، قالب‌بندی آنها جهت نمایش و یا ارسال دستورالعملها به یک مفسر خاص مورد استفاده قرار می‌گیرند.

XSLT : سیستم تبدیل زبان آرایشی قابل توسعه یا Extensible Markup Language Transformations ـ یک سیستم الگوسازی جهت اسناد XML که فرآیند تبدیل اسناد از قالب XML به قالبی دیگر همچون HTML یا WML را به آسانی در اختیار می‌گذارد.

مترجمان : علی ناصح

محمد ناصح

فهرست انتشارات مؤسسه فرهنگی هنری دیباگران تهران

کتابهای کامپیوتر

	کـامـپیوتـر	مؤلف/مترجم
۱	مبانی کامپیوتر	مهندس سعید سعادت
۲	آموزش سریع اپراتوری کامپیوتر	مهندس سعید سعادت
۳	اصول ومفاهیم در فناوری اطلاعات	خسرو مهدی پور
۴	فرهنگ تشریحی لغات و اصطلاحات کامپیوتری مایکروسافت	سعید ظریفی
۵	گواهینامه بین المللی کاربری کامپیوتر(ICDL)(سطح ۱و۲)	مهندس مرتضی متواضع
۶	تمرین و آزمون گواهینامه بین المللی کاربری کامپیوتر(ICDL)(سطح ۱و۲)	مهندس علی اکبر متواضع
۷	آموزش ICDL به زبان ساده (در هفت مهارت)	مهندس علی اکبر متواضع
۸	اپراتوری مقدماتی کامپیوتر	فاطمه کبیری فر - جمشیدصفایی فر
۹	خودآموز سیستم عامل MS-DOS (مقدماتی و پیشرفته)	مهندس سعید سعادت
۱۰	مبانی کامپیوتر (کاردانش)	سعید سعادت و افروز کاشف‌الحق
۱۱	NC و ویروسهای کامپیوتری (کاردانش)	واحد تحقیقات و انتشارات
۱۲	الگوریتم و فلوچارت (کاردانش)	واحد تحقیقات و انتشارات
۱۳	Excel 97 (کاردانش)	مهندس علی اکبر متواضع
۱۴	سیستم عامل DOS (کاردانش)	مهندس سعید سعادت
۱۵	NU (کاردانش)	افروز کاشف الحق و سعید سعادت
۱۶	AutoCAD (کاردانش)	مهندس علیرضا پارسای
۱۷	PowerPoint 97 (کاردانش)	جمیله جامی
۱۸	Q Basic (کاردانش)	واحد تحقیقات و انتشارات
۱۹	ویندوز ۹۸ (۱) (کاردانش)	سعید سعادت و افروز کاشف الحق
۲۰	ویندوز ۹۸ (۲) (کاردانش)	سعید سعادت وافروز کاشف الحق
۲۱	زبان تخصصی رایانه ۱ (کاردانش)	خسرو مهدی پور عطایی
۲۲	زبان تخصصی رایانه ۲ (کاردانش)	خسرو مهدی پور عطایی
۲۳	Word 97 (کاردانش)	مهندس مرتضی متواضع
۲۴	نشر رومیزی MS-Word (مقدماتی و پیشرفته)	مهندس فاطمه کبیری فر
۲۵	برنامه نویسی پاسکال مقدماتی (کمک آموزشی)	مهندس مجتبی الله وردی
۲۶	برنامه نویسی پاسکال پیشرفته (کمک آموزشی)	مهندس مجتبی الله وردی
۲۷	برنامه نویسی Q Basic (کمک آموزشی)	مهندس مجتبی الله وردی
۲۸	Word 97 (کمک آموزشی)	فهیمه وفقی
۲۹	Excel 97 (کمک آموزشی)	فهیمه وفقی
۳۰	راهنمای AutoCAD14 (کمک آموزشی)	نلی آقایی
۳۱	ویندوز ۹۸(۱)(کمک آموزشی)	مهندس مجتبی الله وردی
۳۲	ویندوز ۹۸(۲)(کمک آموزشی)	مهندس مجتبی الله وردی
۳۳	NU (کمک آموزشی)	مهندس مجتبی الله وردی
۳۴	PowerPoint 97(کمک آموزشی)	جمیله جامی
۳۵	راهنمای جامع برنامه نویسی پاسکال	مهرداد اسماعیلی
۳۶	همگام با دلفی ۶ (جلد اول و دوم)	مهرداد اسماعیلی
۳۷	گزارش سازی در دلفی ۶	مهرداد اسماعیلی
۳۸	آموزش Installshield برای دلفی ۶	مهرداد اسماعیلی

۳۹	پرسشهای چهار گزینه‌ای دلفی ۶	مهرداد اسماعیلی
۴۰	تکنیکهای صوتی و تصویری در دلفی ۶	مهندس محمد عادلی نیا- زینب ستوده‌فر
۴۱	برنامه نویسی دلفی ۷(مقدماتی و پیشرفته)	مهرداد اسماعیلی
۴۲	آموزش سریع Windows 98	مهندس سعید سعادت
۴۳	آموزش گام به گام Windows 98	مهندس افروز کاشف الحق
۴۴	مرجع کامل Windows 98	مهندس علی اکبر متواضع
۴۵	خودآموز تصویری Windows 98 (جلد اول و دوم)	مهندس سیما مجاهد - افشین پزشکی
۴۶	آموزش گام به گام Windows 2000	مهندس سعید سعادت
۴۷	خودآموز سریع Windows 2000 حرفه‌ای	مهندس سهیلا کاویان - خسرو مهدی پور
۴۸	مرجع الفبایی Windows 2000	مسعود پاک نظر
۴۹	ایستگاه کاری Windows NT	علی رحیمی فر - مسعود عسگری
۵۰	خودآموز Windows XP	مهندس سعید سعادت - خسرو مهدی پور
۵۱	خودآموز گام به گام windows XP	اردوان نایینی علی اکبری
۵۲	خودآموز آسان Windows XP	فاطمه کبیری فر - جمشید صفایی فر
۵۳	شبکه سازی خانگی با windows XP	مهندس علی اکبر متواضع
۵۴	عیب یابی در windows XP	مهندس علی اکبر متواضع
۵۵	مرجع الفبایی Windows XP	مهندس سیما مجاهد - افشین پزشکی
۵۶	صوت و تصویر برتر در windows XP	پیمان سمرقندی
۵۷	خودآموز گام به گام Office 2002 XP	مهندس علی اکبر متواضع
۵۸	آموزش گام به گام word 2002 XP	مهندس مرتضی متواضع
۵۹	آموزش گام به گام Excel 2002 XP	مهندس علی اکبر متواضع
۶۰	آموزش گام به گام FrontPage2002 XP	مهندسین امیرحسین رضوی-ملیحه دهقان
۶۱	آموزش گام به گام Access 2002 XP	پدرام پاک گوهر
۶۲	آموزش گام به گام Outlook 2002 XP	خسرو مهدی پور عطایی
۶۳	خودآموز سریع PowerPoint 2002 XP	علی رحیمی فر
۶۴	آموزش گام به گام Office 2000	مهندس مرتضی متواضع
۶۵	آموزش گام به گام word 2000	مهندس مرتضی متواضع
۶۶	آموزش گام به گام Excel 2000	مهندس علیرضا پارسای
۶۷	آموزش گام به گام Access 2000	مهندس علی ناصح
۶۸	آموزش گام به گام Outlook 2000	خسرو مهدی پور عطایی
۶۹	آموزش گام به گام FrontPage 2000	پیمان سمرقندی
۷۰	آموزش گام به گام PowerPoint 2000	بیتا خاقانی
۷۱	خودآموز گام به گام Microsoft Project 2000	مهندس مرتضی متواضع
۷۲	پرسش و پاسخ Office 2000	مهندس مرتضی متواضع
۷۳	ناگفته‌های Excel 2000	مهندس علی‌اکبر متواضع
۷۴	پرسش و پاسخ Frontpage 2000	پیمان سمرقندی
۷۵	مدیریت پایگاه داده‌ها در Access 2000	رضا حاتمیان
۷۶	خودآموز آسان Excel 2000	مهندس علی اکبر متواضع
۷۷	خودآموز آسان Word 2000	مهندس مرتضی متواضع
۷۸	خودآموز Photoshop 7 در ۲۴ ساعت	مهندسین امیرحسین رضوی-ملیحه دهقان

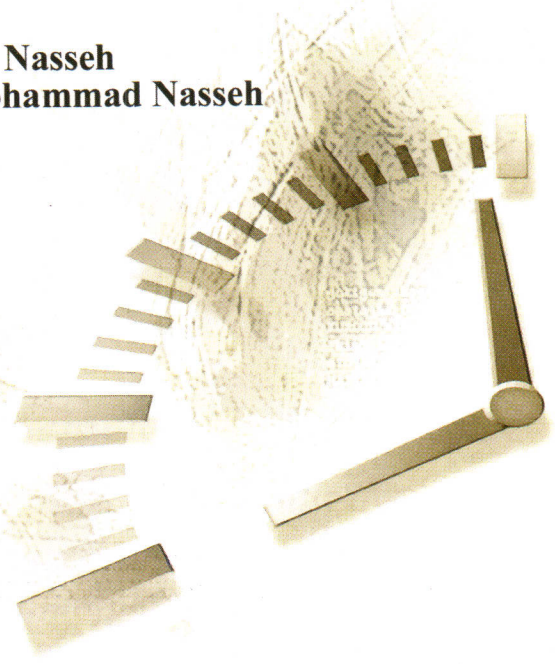
نشانی : سعادت آباد ـ میدان کاج ـ سرو شرقی ـ روبه‌روی خ علامه ـ شماره ۹۷

TEH DIBAGARAN ARTISTIC AND CULTURAL INSTITUTE

RWTUV
TÜV CERT EN ISO 9001
ISO 9001 Certificated
In Publication & Cover Design

Teach Yourself PHP in 24 Hours

Translated by: Ali Nasseh
Mohammad Nasseh



مؤسسه فرهنگی هنری دیباگران تهران
ساختمان مرکزی: سعادت‌آباد- میدان‌کاج- سرو شرقی
روبه‌روی خیابان علامه- پلاک ۹۷
تلفن:۷-۲۰۹۸۴۴۶ دورنگار:۲۰۹۸۴۴۸
E-mail : publishing@mftmail.com
URL : www.mftsite.com

ISBN 964-354-426-5
9 789643 544263