مقدمه

در این آموزش قصد داریم طراحی صفحات وب پویا (Dynamic) را با زبان PHP آموزش‌دهیم. هدف زبان PHP، ایجاد صفحات وب پویا، به‌صورت سریع و آسان است. زبان PHP درکنار یک سرویس‌دهنده وب کار می‌کند و تحت سیستم‌های عامل مختلف مثل Windows، UNIX، Linux و... قابل نصب است. یکی‌از مزایای PHP، رایگان‌بودن آن است. برای نصب PHP می‌توانید نسخه موردنظر خود را از سایت www.php.net دریافت‌کنید. برای کار با بانک اطلاعاتی در وب نیز بهترین‌گزینه، MySQL است که آنرا نیز می‌توانید به‌طور رایگان از سایت www.mysql.com دریافت‌کنید. البته گزینه بهتر، نرم‌افزار Wamp Server است که سرویس‌دهنده وب Apache، آخرین نسخه از زبان PHP و آخرین نگارش MySQL را به‌طور رایگان برای شما نصب می‌کند. برای یادگیری PHP به ابزارهای زیر احتیاج داریم:

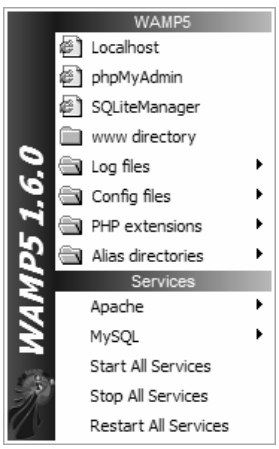
- مرورگر وب (در این آموزش از Internet Explorer استفاده می‌شود)
- سرویس‌دهنده وب (در این آموزش از Apache استفاده می‌شود)
- PHP (در این آموزش از نگارش 5 استفاده می‌شود)
- ویرایشگر متن (در این آموزش از NotePad استفاده می‌شود)
- پایگاه داده‌ها (در این آموزش از MySql نگارش 5 استفاده می‌شود)

آمادگی برای شروع کار

ابتدا مطمئن‌شوید که Wamp Server به‌درستی نصب‌شده و درحال اجرا باشد. برای این‌کار، به System Tray (واقع در کنار ساعت ویندوز) نگاه‌کنید تا یک نیم‌دایره را ببینید. این نیم‌دایره سه وضعیت دارد که فقط یکی از آنها صحیح است:

صحیح	غلط	غلط
		

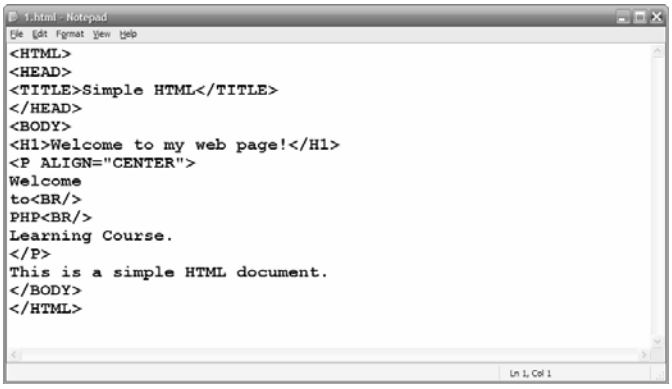
اگر هرکدام از دو وضعیت غلط را مشاهده‌کردید، برروی آن کلیک‌کرده و هرکدام از سرویس‌های پایین منو را که علامت توقف درکنار آن وجود دارد، انتخاب‌کنید و سپس، برروی گزینه Start/Resume Service کلیک‌کنید. اگر این گزینه فعال‌نبود، به‌معنای آن است که سرویس مذکور نصب نشده‌است. لذا برروی Install Service کلیک‌کنید.



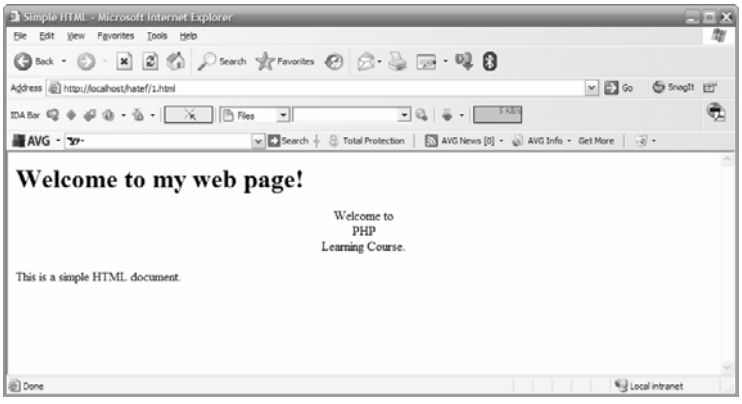
شروع کار

در این آموزش به هیچ عنوان قصد اضافه‌گویی نداریم. بنابراین مستقیماً به‌سراغ اولین مبحث آموزشی می‌رویم. برای شروع کار، کافی‌است NotePad را بازکرده و کد زیر را درون آن تایپ‌کنید (شماره‌های ابتدای خطوط برای توضیح است و نباید آنها را بنویسید:

```
01 <HTML>
02 <HEAD>
03 <TITLE>Simple HTML</TITLE>
04 </HEAD>
05 <BODY>
06 <H1>Welcome to my web page!</H1>
07 <P ALIGN="CENTER">
08 Welcome
09 to<BR/>
10 PHP<BR/>
11 Learning Course.
12 </P>
13 This is a simple HTML document.
14 </BODY>
15 </HTML>
```



سپس فایل را با نام 1.html در پوشه www از مسیر نصب Wamp Server ذخیره‌کنید (اگر تنظیمات نصب را تغییر نداده‌باشید، به‌طور پیش‌فرض مسیر C:\WAMP\www انتخاب می‌شود). برای نمایش صفحه، کافی‌است پنجره مرورگر وب خود (مثلاً Internet Explorer) را باز کنید و در آدرس آن، عبارت localhost/1.html را واردکنید. مشاهده می‌کنید که هرچه در پوشه www قراردهید، ازطریق عبارت localhost قابل دسترسی است (مثلاً برای نمایش فایل C:\WAMP\www\1.html باید عبارت localhost/1.html را وارد کنید). بدین‌ترتیب، صفحه‌ای مشابه پنجره زیر مشاهده خواهید کرد:



این سند وب، نتیجه دستوراتی است که قبلاً نوشتیم.

تشریح کد

همان‌طور که احتمالاً می‌دانید، در زبان HTML هرچه تایپ‌شود، عیناً در خروجی ظاهر می‌گردد؛ مگر اینکه بین علامت‌های < و > محصور گردد که در چنین شرایطی، معنا و مفهوم خاصی خواهد داشت (البته فقط یک‌سری کلمات خاص این ویژگی را دارند). به چنین ساختاری، اصطلاحاً تگ (Tag) می‌گویند. درواقع، دستورات HTML یک‌سری تگ هستند که نحوه نمایش صفحه را تعیین می‌کنند. هر تگ در HTML شامل سه قسمت است:

۱- بازکردن تگ: این قسمت با علامت > و در ادامه، نام تگ و پارامترهای آن (درصورت وجود) و درنهایت، علامت < مشخص می‌شود. قسمت پارامترها اختیاری است و درصورت ذکرنشدن، آخرین مقدار آنها درنظر گرفته می‌شود و اگر قبلاً تغییری در آنها ایجاد نشده‌باشد، همان مقادیر پیش‌فرض خود را در HTML خواهند داشت.

۲- بدنه تگ: درصورتی که یک تگ، تنظیمات خاصی را در HTML فعال‌کند، هر عبارتی که بین قسمت بازکردن و بستن تگ ذکرشود، از آن تنظیمات پیروی خواهد کرد. به این قسمت، بدنه تگ می‌گویند و درواقع، محدوده فعالیت آنرا مشخص می‌کند.

۳- بستن تگ: این قسمت با علامت /> و در ادامه، نام تگ و درنهایت، علامت < مشخص می‌شود. بدیهی‌است که تنظیماتی که تگ مربوطه فعال کرده باشد، بعد از بستن آن، به حالت قبل‌از بازکردن آن‌تگ برمی‌گردد.

در دستور 01، تگ HTML بازشده‌است. تمامی دستورات HTML باید درون این تگ قرارگیرند تا پردازش‌شوند. البته برخی از مرورگرها، تگ‌های HTML را خارج‌از تگ فوق نیز پردازش می‌کنند؛ اما از آنجا که این مسئله، عمومی نیست، پیشنهاد می‌شود همیشه اولین دستور صفحه HTML شما، بازکردن تگ HTML و آخرین دستور نیز بستن تگ مذکور باشد.

در دستور 02 تگ HEAD بازشده‌است. این تگ، قسمت سرصفحه را مشخص می‌کند. ازآنجا که سرصفحه قبل‌از بدنه صفحه پردازش می‌شود، بهتر است تنظیمات کلی صفحه و به‌طورکلی کارهایی که لازم‌است ابتدا انجام‌شود، در سرصفحه قرارگیرد.

در دستور 03، سه‌کار انجام می‌شود. ابتدا تگ TITLE بازشده‌است. این تگ برای مشخص‌کردن عنوان صفحه که در نوار عنوان پنجره ظاهر می‌شود، به‌کار می‌رود. سپس عنوان موردنظر مشخص شده و درنهایت، تگ TITLE بسته‌شده‌است. همان‌طور که در این دستور ملاحظه می‌کنید، نوشتن هر دستور HTML در یک سطر، اجباری نیست؛ اما به خوانایی صفحه کمک زیادی می‌کند.

در دستور 04 تگ HEAD بسته‌شده‌است.

در دستور 05 تگ BODY بازشده‌است. این تگ برای تولید بدنه صفحه به‌کار می‌رود.

در دستور 06، تگ H1 بازشده‌است. این تگ برای مشخص‌کردن تیتر با بیشترین اولویت به‌کار می‌رود (از تگ‌های H1 تا H6 می‌توان برای مشخص‌کردن درجه تیتربندی متن استفاده‌نمود که با زیادشدن عدد مربوطه، سایز قلم کوچک‌تر می‌شود). در ادامه، متن تیتر ذکرشده و سپس، تگ H1 بسته می‌شود.

در دستور 07 از تگ P برای ایجاد یک پاراگراف استفاده شده‌است. در این دستور، مثالی از نحوه مشخص‌کردن پارامترها را ملاحظه می‌کنید. در اینجا، پارامتر ALIGN از تگ P برای مشخص‌کردن نحوه تراز متن به‌صورت وسط‌چین استفاده شده‌است. درطول این آموزش، سعی خواهیم‌کرد با تگ‌های مختلف درزمان استفاده از آنها و همچنین پارامترهایی که مورد استفاده قرار می‌گیرد، آشنا شویم.

در دستورات 08 تا 11 بدنه تگ P مشخص شده‌است. همان‌طور که در اینجا نیز ملاحظه می‌کنید، نوشتن همه بخش‌های یک دستور در یک سطر اجباری نیست. به‌علاوه، همه کارکترهای سرسطر در HTML به‌عنوان فاصله (Space) تعبیر می‌شوند. درحقیقت در HTML برای رفتن به سطر بعد، از یک تگ خاص به‌نام BR باید استفاده‌کنید. این تگ از آن‌جهت خاص است که به‌تنهایی، شامل قسمت‌های بازکردن و بستن است. اگر به‌دقت به آن نگاه‌کنید، خواهید دید که به‌جای < با /> بسته شده‌است. در HTML هرگاه یک تگ، تمامی پارامترهای موردنیاز خود را درون قسمت بازکردن تگ دریافت‌کند و ضمناً شامل قسمت بدنه نباشد، در همان قسمت بازکردن، با قراردادن یک علامت / قبل‌از علامت >، بلافاصله تگ بسته می‌شود.

در دستور 12 تگ P بسته‌شده‌است.

در دستور 13 ادامه بدنه تگ BODY را ملاحظه می‌کنید. همان‌طور که در خروجی قابل‌مشاهده است، این سطر در وسط پنجره ظاهر نمی‌شود؛ زیرا محدوده اعتبار تگ P به‌اتمام رسیده‌است و سطرهای بعد از آن، با همان تنظیمات پیش‌فرض نمایش داده خواهند شد.

در دستور 14 تگ BODY بسته‌شده‌است. بدین‌ترتیب بدنه صفحه به‌اتمام می‌رسد.

در دستور 15 تگ HTML بسته‌شده‌است. درنتیجه، سند HTML نیز به‌اتمام می‌رسد و پس‌از این دستور، هیچ عبارتی ذکر نمی‌شود.

مشاهده مجدد صفحه HTML

در پنجره مرورگر، کلید F5 را فشاردهید (یا از منوی View، گزینه Refresh را انتخاب‌کنید). ملاحظه می‌کنید که مجدداً همان اطلاعات ظاهر می‌شود. یکی‌از بزرگ‌ترین مشکلات HTML، ایستایی (ثابت‌بودن) آن‌است؛ یعنی برای مشاهده صفحه‌ای متفاوت، باید کد آن تغییرکند.

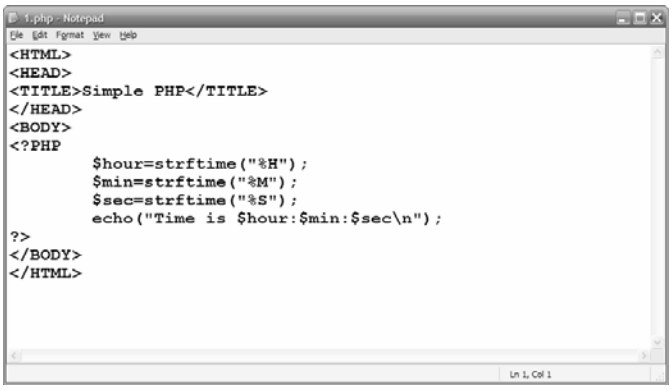
مشاهده کد منبع صفحه HTML

در پنجره مرورگر، روی یک جای خالی از صفحه، کلیک راست کرده و گزینه View Source را انتخاب‌کنید (یا از منوی View، گزینه Source را برگزینید). خواهید دید که پنجره NotePad بازشده و کد منبعی که صفحه خروجی را تولید کرده‌است، به‌شما نشان می‌دهد. واضح‌است که در HTML هیچ امنیتی برای کد شما وجود ندارد. حال در ادامه، مثالی از کد PHP را مشاهده خواهید کرد و تفاوت‌های اصلی آن با HTML مورد بحث قرار خواهد گرفت.

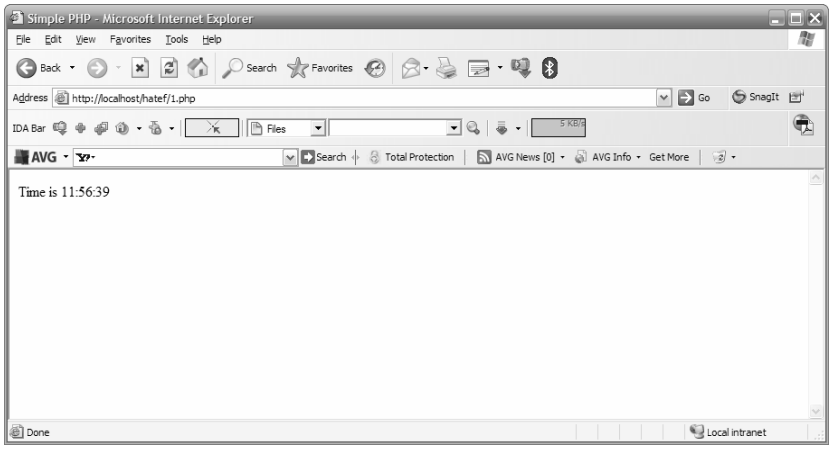
اولین کد PHP

در پنجره NotePad، از منوی File گزینه New را انتخاب‌کنید تا یک سند جدید دراختیار شما قرارداده‌شود. حال دستورات زیر را درون آن تایپ‌کنید (بدون شماره‌های خطوط) و آنرا در همان مسیر قبلی با نام 1.php ذخیره‌کنید:

```
01 <HTML>
02 <HEAD>
03 <TITLE>Simple PHP</TITLE>
04 </HEAD>
05 <BODY>
06 <?PHP
07     $hour=strftime("%H");
08     $min=strftime("%M");
09     $sec=strftime("%S");
10     echo("Time is $hour:$min:$sec\n");
11 ?>
12 </BODY>
13 </HTML>
```



بهتر است قبل‌از هرگونه توضیح، خروجی این صفحه را مشاهده‌کنیم (برای این‌کار، در نوار آدرس مرورگر عبارت localhost/1.php را بنویسید):



همان‌طور که ملاحظه می‌کنید، خروجی این صفحه، نمایش ساعت در صفحه وب است. تنها تفاوت این کد با کد قبل، بدنه صفحه (تگ BODY) است. در اینجا، به‌جای دستورات قبل، یک تگ ویژه را ملاحظه می‌کنید:

```
<?PHP
    ...
?>
```

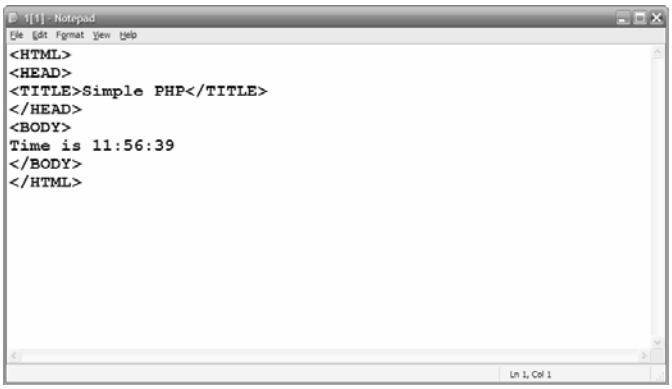
این تگ برای درج دستورات PHP درون کد HTML به‌کار می‌رود. قسمت بازشدن این تگ به‌صورت <?PHP و قسمت بستن آن به‌صورت ?> است. دستورات PHP بین این دو قسمت قرار می‌گیرند. فعلاً دستورات PHP که در این مثال مورد استفاده قرار گرفته‌اند را مورد بررسی قرار نمی‌دهیم (اما در آینده، همه آنرا تشریح خواهند شد).

مشاهده مجدد صفحه PHP

توسط کلید F5 یا انتخاب Refresh از منوی View، صفحه را مجدداً نمایش‌دهید. ملاحظه می‌کنید که ساعت تغییر می‌کند! همان‌طور که می‌بینید، در PHP می‌توان در زمان‌های مختلف درخواست یک صفحه، با توجه به شرایط مختلف، نتایج متفاوتی را نمایش‌داد؛ اما نکته مهم آن‌است که این نتایج متفاوت، بدون تغییر مجدد کد منبع ایجاد می‌شوند. بدیهی‌است که نمایش ساعت فقط یک مثال است. برای مثال، می‌توانید براساس تنظیمات مختلفی که در بانک اطلاعاتی برای هر کاربر ذخیره شده‌است، به‌محض ورود یک کاربر به سیستم، وب‌سایت خود را به‌نحوی که وی قبلاً تنظیم‌کرده‌است، نمایش‌دهید (البته به‌شرطی که صفحه خود را به‌نحوی طراحی کرده‌باشید که تغییرات را بپذیرد و در صفحه اعمال‌کند).

مشاهده کد منبع صفحه PHP

برروی صفحه کلیک راست کرده و گزینه View Source را انتخاب‌کنید (یا از منوی View، گزینه Source را برگزینید). پنجره‌ای شبیه تصویر زیر ظاهر می‌شود:



واضح‌است که این کد، کد منبع صفحه نیست؛ زیرا نه از تگ PHP خبری است و نه از دستورات PHP که درون‌آن نوشته‌بودیم. در اینجا، دستورات زیر:

```
<?PHP
    $hour=strftime("%H");
    $min=strftime("%M");
    $sec=strftime("%S");
    echo("Time is $hour:$min:$sec\n");
?>
```

پردازش‌شده و خروجی زیر را تولید کرده‌اند:

```
Time is 11:56:39
```

بنابراین مشاهده می‌کنید که کد شما، کاملاً از دید کاربران نهایی، پنهان است و امنیت آن، حفظ می‌شود.

معنا و هدف PHP

PHP مخفف به‌هم‌ریخته عبارت Hypertext Pre-Processor است (به‌معنای پیش‌پردازنده فوق‌متن). علت استفاده از عبارت فوق‌متن، آن‌است که کلیه دستورات HTML درون فایل‌های PHP قابل‌پردازش است (عکس این‌مطلب وجود ندارد و کدهای PHP درون فایل‌های HTML پردازش نمی‌شوند – منظور، فایل‌هایی است که با پسوند html. ذخیره می‌شوند). منظور از پیش‌پردازنده نیز، آن‌است که صفحات PHP ابتدا در سمت سرور پردازش‌شده و خروجی مناسب با قالب HTML تولیدشده و سپس، خروجی موردنظر به کلاینت ارسال می‌شود. بنابراین، کد PHP اصلی از دید کاربر کاملاً محفوظ خواهد بود.

اولین قانون در PHP

اولین و درعین‌حال، ساده‌ترین قانون PHP که مهم‌ترین قانون نیز می‌باشد، آن‌است که هر دستور، باید با سمی‌کالِن (;) خاتمه‌یابد. برای مثال، دستور زیر موجب بروز خطا خواهد شد:

```
echo("Hello")
```

برای اصلاح آن، باید این‌گونه بنویسید:

```
echo("Hello");
```

توضیحات در PHP

همان‌طور که احتمالاً می‌دانید، PHP یک زبان برنامه‌نویسی است. ساختار این زبان بسیار شبیه به زبان C++ است. یکی‌از امکانات سودمند در هر زبان برنامه‌نویسی، توضیحات هستند که برای کسی که بعداً قصد مطالعه کد را دارد، بسیار مفید است. در PHP برای درج توضیحات تک‌سطری در برنامه می‌توان از علامت‌های # و // استفاده‌نمود. برای مثال، عبارات زیر، هر دو، توضیح هستند و پردازش نمی‌شوند:

```
#This is a comment
//This is also a comment
```

برای توضیحات چندسطری نیز می‌توانید ابتدای توضیحات را با /* و پایان آنرا با */ مشخص‌کنید:

```
/*This is
a multiline
comment*/
```

متغیرها

هر زمان صحبت از زبان‌های برنامه‌نویسی به‌میان می‌آید، بدون‌شک اولین مفهومی که مطرح می‌شود، متغیر است. متغیر مکانی از حافظه است که دارای نام می‌باشد و توسط نام، می‌توان به آن دسترسی‌داشته و درصورت لزوم،آنرا تغییر داد. در PHP متغیرها با استفاده از علامت $ (دلار) تعریف می‌شوند. مثلاً $myVariable یک متغیر به‌نام myVariable است. نام متغیرها در PHP نسبت‌به بزرگی و کوچکی حروف، حساس است. برای مثال، دو متغیر $var و $VAR در PHP با یکدیگر متفاوتند. برای نام‌گذاری متغیر در PHP باید از دو قانون زیر پیروی‌کنیم:

۱- نام متغیر فقط می‌تواند شامل حروف (A-Z و a-z)، اعداد (0-9) و کارکتر خط‌زیر یا Underline (_) باشد.

۲- نام متغیر نمی‌تواند با عدد شروع‌شود.

بنابراین ازبین اسامی زیر، فقط سه مورد اول صحیح است:

```
$name              ✓
$_Name20           ✓
$my_1st_Name       ✓
$2_name            ✘
```

در PHP نوع متغیر تعیین نمی‌شود. درعوض، یک متغیر می‌تواند هر نوع مقداری را بپذیرد و نوع آن، به‌طور خودکار براساس محتوای آن، تغییر خواهد کرد. مثال:

```
$var=5;            ✓
$var="PHP";        ✓
$var=3.14;         ✓
```

هر سه دستور فوق، صحیح است.

مقداردهی به‌روش مقداری و ارجاعی

به دستورات زیر دقت‌کنید:

```
$x=5;
$y=$x;
$x=6;
```

در این مثال، بعد از اجرای دستورات فوق، مقدار $y برابر با 5 خواهد بود؛ زیرا در زمان مقداردهی، یک کپی از مقدار $x درون متغیر $y قرار گرفته‌است و درصورت تغییر مقدار $x، مقدار $y بدون تغییر خواهد ماند.

حال به دستورات زیر توجه‌کنید:

```
$x=5;
$y=&$x;
$x=6;
```

در این مثال، به‌دلیل استفاده از کارکتر & قبل‌از متغیر $x، به‌جای قراردادن یک کپی از مقدار $x درون $y، یک ارجاع به $x درون $y قرار می‌گیرد و درنتیجه، $y به همان محلی از حافظه اشاره می‌کند که $x اشاره دارد. بنابراین، درحقیقت، $y یک اسم مستعار (Alias) برای $x خواهد شد. به چنین شرایطی، مقداردهی ارجاعی می‌گویند. درنتیجه بعد از اجرای دستورات فوق، $y نیز مقدار 6 را در خود خواهد داشت.

انواع متغیرها در PHP

به‌طور کلی، هر متغیر در PHP می‌تواند یکی از انواع داده‌ای زیر را بپذیرد:

```
Integer
Floating-Point
String
Object
Array
```

که به‌ترتیب، برای نگه‌داری اعداد صحیح، اعداد اعشاری، رشته‌ها، اشیاء و آرایه‌ها به‌کار می‌روند.

آشنایی با دستور echo

این دستور، هرچه که به‌عنوان پارامتر دریافت‌کند را در محل قرارگرفتن مکان‌نما، در فایل HTML خروجی (که کاربر ملاحظه خواهد کرد)، می‌نویسد. برای مثال، دستورات زیر را درنظر بگیرید:

```
<HTML>
<HEAD>
<TITLE>Simple PHP</TITLE>
</HEAD>
<BODY>
<?PHP
	$x=5;
	$y=6;
	$z=$x+$y;
	echo("$z\n");
?>
</BODY>
</HTML>
```

کدهای HTML که خارج‌از تگ PHP قراردارند، عیناً در فایل HTML خروجی نوشته می‌شوند؛ اما کدهای PHP ابتدا پردازش‌شده و سپس، هرآنچه دستور echo به خروجی منتقل‌کند، در خروجی نوشته می‌شوند. درنتیجه، درصورتی‌که کد فوق را در یک فایل با پسوند php. ذخیره‌کرده و آنرا در مرورگر مشاهده‌کنید، عدد 11 را در صفحه خواهید دید و اگر برروی صفحه کلیک راست کرده و گزینه View Source را برگزینید، کد زیر را ملاحظه خواهید کرد:

```
<HTML>
<HEAD>
<TITLE>Simple PHP</TITLE>
</HEAD>
<BODY>
11
</BODY>
</HTML>
```

در این کد، کلیه عبارات (تا ابتدای تگ PHP عیناً در کد خروجی ظاهر شده‌اند. سپس کدهای PHP پردازش شده و مجموع متغیر $x با مقدار 5 و متغیر $y با مقدار 6، مقدار متغیر $z یعنی 11 را ساخته‌است. درنتیجه، دستور echo مقدار این متغیر یعنی 11 را در کد خروجی، بعد از دستور بازکردن تگ BODY نوشته‌است. سپس تگ PHP بسته‌شده و مابقی عبارات که HTML هستند، عیناً در کد خروجی ظاهر شده‌اند.

کار با رشته‌ها

رشته‌ها، مجموعه‌ای از کارکترهای متوالی هستند که بین دو گیومه تک (') یا دو گیومه جفت (") قرار می‌گیرند. مثال:

```
$a='This is a text.';
$b="This is a text too.";
```

درصورتی‌که بخواهید کارکتر ' (گیومه تک) را در رشته‌هایی که بین دو گیومه تک قراردارند درج‌کنید، باید یک کارکتر \ قبل‌از آن قراردهید:

```
$text='It\'s mine!';
```

به‌همین‌ترتیب، برای درج کارکتر " (گیومه جفت) در رشته‌های محصورشده بین دو گیومه جفت، باید یک کارکتر \ قبل‌از آن بگذارید:

```
$text="My friend's name is \"Ali\" and he's a good person.";
```

به کارکتر \ اطلاحاً ESCAPE (فرار) می‌گویند؛ زیرا برای فرار از معنای اصلی کارکترهای بعد از آن (و اعطای مفهومی دیگر به آنها) به‌کار می‌رود. در جدول زیر، تعدادی از کدهای ESCAPE را ملاحظه می‌کنید:

کد ESCAPE	نتیجه
\n	حرکت به ابتدای سطر بعد
\r	حرکت به ابتدای سطر جاری
\t	کارکتر TAB (معادل 8 کارکتر SPACE)
\\	کارکتر \
\"	کارکتر " (در رشته‌های محصور بین یک جفت ")
\'	کارکتر ' (در رشته‌های محصور بین یک جفت ')
\$	کارکتر $
\[0-7]	کارکتری که کد ASCII آن در مبنای هشت در جلوی آن نوشته شده‌است
\x[0-F]	کارکتری که کد ASCII آن در مبنای شانزده در جلوی آن نوشته شده‌است

ادغام رشته‌ها

برای ادغام رشته‌ها در PHP از کارکتر نقطه (.) استفاده می‌شود. برای‌مثال:

```
$a="PHP";
$b="Programmer";
$c=$a." ".$b; //$c="PHP Programmer";
```

در PHP می‌توانید رشته‌ها را با اعداد نیز ترکیب‌کنید:

```
$num=5;
$x="Test".$num; //$x="Test5";
```

تفاوت رشته‌های محصور بین گیومه جفت و تک

دستورات زیر را درنظر بگیرید:

```
$x=5;
$text1="X is $x";
$text2='X is $x';
```

در نتیجه دستورات فوق، عبارت X is 5 در متغیر $text1 و عبارت X is $x در متغیر $text2 است. علت این تفاوت در عملکرد، آن‌است که اسامی متغیرها در رشته‌هایی که بین دو گیومه جفت قراردارند، پردازش شده و به‌جای نام آنها، از مقدارشان در عبارت استفاده می‌شود؛ حال‌آنکه در رشته‌های محصور بین دو گیومه تک، هیچ پردازشی روی رشته انجام نمی‌شود و به‌همان‌شکل که نوشته می‌شود، مورد استفاده قرار خواهد گرفت. البته این پردازش، فقط مختص متغیرها است و هیچ‌گونه فراخوانی تابع (در آینده توضیح داده خواهد شد) یا عمل محاسباتی ریاضی پردازش نخواهد شد. برای مثال، نتیجه دستورات زیر:

```
$x=5;
$text1="X=($x+5)";
```

ذخیره عبارت X is (5+5) درون متغیر $text1 است. برای محاسبه صحیح مجموع و درج آن در رشته، باید به‌صورت زیر عمل‌شود:

```
$text1="X=".($x+5);
```

درنتیجه، در انتهای عبارت X=، نتیجه محاسبه ($x+5) به‌صورت یک رشته، ادغام‌شده و عبارت حاصل، درون متغیر $text1 قرار می‌گیرد.

استفاده از رشته به‌عنوان عدد

همان‌طور که در مثال قبل ملاحظه‌کردید، تبدیل عدد به رشته، در زمان نیاز، به‌طور خودکار انجام می‌شود. عکس این‌موضوع نیز صحیح‌است و رشته‌ها نیز درصورتی‌که در عبارات محاسباتی ریاضی مورداستفاده قرارگیرند، به‌عدد تبدیل می‌شوند؛ بدین‌ترتیب که از ابتدای رشته، تا جایی که با یک کارکتر غیرعددی برخورد نشود، جداشده و به‌صورت عدد تعبیر خواهد شد. اگر در ابتدای رشته، عددی وجود نداشته‌باشد، یک ثابت خاص به‌نام NaN (مخفف Not a Number) بازگردانده خواهد شد. مثال:

```
$text="52Ali";
$y=7+$text;        //$y=59;
```

محاسبه طول رشته

توسط تابع strlen می‌توان طول یک رشته را محاسبه‌کرد:

```
$text="Alireza";
$len=strlen($text); //$len=7;
```

عملگرها در PHP

تاکنون با روش تعریف و مقداردهی متغیرها آشنا شدیم. حال قصد داریم روش تغییر و دستکاری آنها را بیان‌کنیم. برای این‌کار، از عملگرها استفاده می‌شود. ساده‌ترین عملگر در PHP، عملگر انتساب است که با نماد = بیان می‌شود. برای مثال، دستور:

```
$x=5;
```

مقدار 5 را به متغیر $x نسبت می‌دهد؛ یا دستور:

```
$y=$x;
```

مقدار متغیر $x را به متغیر $y منسوب می‌کند. از این عملگر برای مقداردهی رشته‌ها نیز می‌توان استفاده‌نمود:

```
$t="Hello!";
```

عملگرهای ریاضی

PHP از پنج‌نوع عملگر ریاضی پشتیبانی می‌کند:

نام عملگر	نماد	مثال	توضیح
جمع	+	$a+$b	جمع $a و $b
تفریق	-	$a-$b	تفریق $b از $a
ضرب	*	$a*$b	ضرب $a در $b
تقسیم	/	$a/$b	تقسیم $a بر $b
باقیمانده	%	$a%$b	باقیمانده تقسیم $a بر $b

برای مثال، درصورتی‌که بخواهیم حاصل‌ضرب متغیرهای $x و $y را در $z ذخیره‌کنیم، باید این‌گونه بنویسیم:

```
$z=$x*$y;
```

هم‌چنین اگر بخواهیم به مقدار قبلی متغیر $a، 5 واحد اضافه‌کنیم، از این دستور استفاده می‌کنیم:

```
$a=$a+5;
```

البته در چنین حالت‌هایی که متغیر سمت‌چپ تساوی، بلافاصله بعد از عملگر انتساب ظاهر می‌شود، ساختار بهینه‌تری وجود دارد:

```
$a+=5;
```

دقت‌کنید که اگر بلافاصله بعد از عملگر انتساب، متغیر سمت چپ تساوی وجود نداشته‌باشد، نمی‌توان از این ساختار استفاده‌کرد:

```
$b=5-$b;      ✓
$b-=5;        ✘      //Result: $b=$b-5; not $b=5-$b;
```

به‌علاوه، برای افزایش و کاهش به‌میزان یک‌واحد نیز روشی باز هم ساده‌تر وجود دارد:

```
$a=$a+1;          $b=$b-1;
$a+=1;            $b-=1;
$a++;             $b--;
++$a;             --$b;
```

نتیجه اجرای تمامی دستورات دو گروه فوق، یکسان‌است. به دو دستور آخر هر گروه دقت‌کنید. درنتیجه اجرای هرکدام از دو دستور آخر گروه اول، مقدار متغیر $a یک‌واحد افزایش‌یافته وبا اجرای هرکدام از دو دستور آخر گروه دوم، مقدار متغیر $b یک‌واحد کاهش می‌یابد. تفاوت این دو دستور، در زمان افزایش یا کاهش است. برای درک بهتر، به مثال زیر دقت‌کنید:

```
$a=5;
$x="A=".$a++;
```

در این دستور، چون ابتدا $a ذکرشده و سپس، عملگر افزایش یک‌واحدی مورد استفاده قرار گرفته‌است، ابتدا مقدار فعلی آن (یعنی 5) در رشته $x قرارگرفته و سپس، یک‌واحد به آن افزوده می‌شود؛ لذا رشته $x حاوی عبارت A=5 خواهد بود. حال اگر همان‌دستور را به‌صورت زیر بنویسیم:

```
$x="A=".++$a;
```

ابتدا به متغیر $a یک‌واحد افزوده‌شده و سپس، در عبارت مورد استفاده قرار می‌گیرد؛ لذا رشته $x حاوی عبارت A=6 خواهد بود.

عملگرهای مقایسه‌ای

از این عملگرها برای مقایسه دو عبارت به‌صورت ریاضی استفاده می‌شود.

عملگر	مثال	توضیح
==	$a==$b	بررسی تساوی مقدار (5 و 5.0 و "5" و... با هم برابرند)
===	$a===$b	بررسی تساوی مقدار و نوع (5 و 5 با هم برابر و با 5.0 و "5" و... متفاوتند)
!=	$a!=$b	بررسی عدم‌تساوی مقدار (5 و 6 با هم متفاوتند ولی 5 و 5.0 و "5" و... با هم برابرند)
!==	$a!==$b	بررسی عدم‌تساوی مقدار و نوع (5 و 6 و 5.0 و "5" و... با هم متفاوتند)
>	$a>$b	بررسی بزرگ‌تربودن
<	$a<$b	بررسی کوچک‌تربودن
>=	$a>=$b	بررسی بزرگ‌تر یا مساوی بودن
<=	$a<=$b	بررسی بزرگ‌تر یا مساوی بودن

عملگرهای منطقی

از این عملگرها برای ترکیب شرایط مختلف جهت بررسی‌کردن استفاده می‌شود:

عملگر	مثال	توضیح
and	$a<5 and $b>2	هر دو شرط باید برقرار باشند
&&	$a<5 && $b>2	هر دو شرط باید برقرار باشند
or	$a<5 or $b>2	کافی‌است یکی‌از دو شرط برقرار باشند
||	$a<5 || $b>2	کافی‌است یکی‌از دو شرط برقرار باشند
xor	$a<5 xor $b>2	فقط باید یکی‌از دو شرط برقرار باشد
^	$a<5 ^ $b>2	فقط باید یکی‌از دو شرط برقرار باشد
!	!$a<5	شرط نباید برقرار باشد

اولویت عملگرها

درصورتی‌که در یک عبارت، از چند عملگر استفاده‌کنید، نتیجه عبارت با توجه به این‌که کدام‌عملگر ابتدا ارزیابی‌شود، متفاوت خواهد بود. برای مثال، در عبارت 2*4+5، اگر ابتدا + عمل‌کند، نتیجه، 18 و درصورتی‌که ابتدا * عمل‌کند، نتیجه 13 خواهد بود. در چنین شرایطی، اولویت عملگرها مشخص‌کننده ترتیب ارزیابی آنهاست. در PHP عملگرها به‌ترتیب زیر ارزیابی می‌شوند (عملگرهای سطرهای بالاتر، اولویت بیشتری دارند و عملگرهای یک‌سطر، دارای اولویت برابر هستند و به همان‌ترتیب نوشته‌شدن در عبارت، ارزیابی می‌شوند:

```
[       (
!       ~       ++      --
*       /       %
+       -       .
<       <=      >       >=
==      !=      ===     !==
&
^
|
&&      and
xor
||      or
?:
=       +=      -=      *=      /=      %=      .=      &=      |=      ^=      ~=
```

کنترل روند اجرای برنامه

تمامی برنامه‌هایی که تا اینجا بررسی‌کردیم، بدین‌صورت بودند که از ابتدا تا انتها، خط‌به‌خط اجرا می‌شدند. با استفاده از ساختارهای کنترلی، می‌توانیم این روش اجرا را تغییر دهیم.

ساختار if

این ساختار برای کنترل یک یا چند شرط و اجرای یک یا چند دستور درصورت برقراری یا عدم‌برقراری آن شرط یا شرایط، به‌کار می‌رود. ساختار کلی این دستور به‌صورت زیر است:

```
if( CONDITION(S)1 )
{
      true 1 block
}
elseif( CONDITION(S)2 )
{
      true 2 block
}
...
elseif( CONDITIONS(S)n )
{
      true n block
}
else
{
      false block
}
```

توضیح: درصورتی که شرط یا شرایط 1 برقرار باشند، بلاک true 1 اجرا می‌شود. درغیر این‌صورت، شرط یا شرایط 2 بررسی می‌شوند و درصورت برقرار بودن آنها، بلاک true 2 اجرا می‌گردد. درصورت عدم برقراری این شرط یا شروط نیز شرط یا شروط بعدی تا n (هر اندازه که باشند)، بررسی می‌گردند و درصورت برقراری هرکدام از شرط‌ها، بلاک کد مربوطبه همان شرط اجرا می‌شود. درنهایت، اگر هیچ‌کدام از بلاک‌های if یا elseif برقرار نبودند، بلاک false (کد مربوطبه else) اجرا می‌شود. در ساختار فوق، تمامی قسمت‌های elseif و قسمت else، اختیاری هستند و بنا به نظر برنامه‌نویس، می‌توان از آنها استفاده‌نمود. ضمناً اگر در هرکدام از بلاک‌ها، فقط یک دستور وجود داشته‌باشد، می‌توان از { و } صرف‌نظر نمود؛ هرچند به‌دلیل کاهش خوانایی برنامه و خطر بروز اشکالات احتمالی، این‌کار پیشنهاد نمی‌شود. برای درک بهتر، به مثال زیر دقت‌کنید:

```
if($day==0)
{
      $wday="Saturday";
}
elseif($day==1)
{
      $wday="Sunday";
}
...
$elseif($day==7)
{
      $wday="Friday";
}
else
{
      $wday="ERROR";
}
```

در مثال فوق، مقدار متغیر $day به‌ترتیب با مقادیر 0 تا 7 مقایسه می‌شود و درصورت برابربودن با هرکدام از آنها، نام روز مربوطه در متغیر $wday قرار می‌گیرد. درصورتی‌که مقدار متغیر $day هیچ‌کدام از مقادیر فوق نباشد، عبارت ERROR درون متغیر $wday قرار خواهد گرفت.

ساختار switch

اگر به‌خوبی به ساختار if مثال قبل دقت‌کنید، خواهید دید که در آن، مقدار متغیر $day با مقادیر مختلف بررسی‌شده و درصورت برابربودن با هرکدام از مقادیر، یک بلاک کد اجرا می‌شود. در چنین مواردی (ارزیابی یک متغیر با مقادیر مختلف و انجام کارهای متفاوت براساس مقادیر مختلف آن)، ساختار بهینه‌تری وجود دارد:

```
switch (VARIABLE)
{
	case VALUE 1:
		true 1 block
		break;
	case VALUE 2:
		true 2 block
		break;
	...
	case VALUE n:
		true n block
		break;
	default:
		false block
		break;
}
```

در این ساختار، مقدار VARIABLE به‌صورت بسیار سریع با تمامی caseها مقایسه می‌شود و درصورت برابربودن با هرکدام از آنها، بلاک کد همان‌قسمت تا زمان رسیدن به اولین دستور break; اجرا می‌شود. درصورتی‌که مقدار متغیر با هیچ‌کدام از مقادیر caseها برابر نباشد، بلاک کد مربوطبه قسمت default اجرا خواهد شد. برای درک بهتر، مثال قبل را با این ساختار بازنویسی می‌کنیم:

```
switch($day)
{
	case 0:
		$wday="Saturday";
		break;
	case 1:
		$wday="Sunday";
		break;
	...
	case 7:
		$wday="Friday";
		break;
	default:
		$wday="ERROR";
		break;
}
```

درمورد این ساختار دقت‌کنید که همیشه نمی‌توان از آن به‌عنوان جایگزین if استفاده‌نمود. برای استفاده از این ساختار، باید شرایط زیر فراهم‌باشد:

۱- اجرای بلاک‌های مختلف، بستگی به مقادیر مختلف یک متغیر داشته‌باشد (اگر بخواهیم چند شرط یا چند متغیر را بررسی‌کنیم، باید از if استفاده‌کنیم).

۲- مقادیر کاملاً مشخص و متمایز باشند (برای مثال، اگر بخواهیم درصورت بزرگتربودن از یک مقدار، کارهایی انجام‌دهیم، باید از if استفاده‌شود).

ضمناً در پایان هر case، باید از break; استفاده‌شود؛ درغیر این‌صورت، دستورات آن case تا زمان رسیدن به اولین break; یا رسیدن به انتهای ساختار switch، اجرا خواهند شد. البته این مسئله، در برخی‌موارد، سودمند است. مثال:

```
switch($operator)
{
	case '+':
		$result=$x+$y;
		break;
	case '-':
		$result=$x-$y;
		break;
	case '/':
		if($y!=0)
		{
			$result=$x/$y;
		}
		else
		{
			$result="DIVISION BY ZERO";
		}
		break;
	case '*':
	case 'x':
	case 'X':
		$result=$x*$y;
		break;
	default:
		$result="INVALID OPERATOR";
		break;
}
```

در این مثال، درصورتی‌که $op برابر با یکی‌از کارکترهای *، x یا X باشد، حاصل‌ضرب $x و $y را در $result ذخیره می‌کند.

ساختار :?

معمولاً بسیاری از دستورات if با ساختاری مشابه مثال زیر مورد استفاده قرار می‌گیرند:

```
if($x%2==0)
{
	echo("X is even.<BR/>\n");
}
else
{
	echo("X is odd.<BR/>\n");
}
```

همان‌طور که ملاحظه می‌کنید، بخش اعظمی از کد بلاک‌های true و false مشابه است. بنابراین می‌توانیم از ساختار :? برای خلاصه‌کردن این ساختار بهره‌مند شویم. این ساختار، به‌صورت زیر است:

```
CONDITION(S)?TRUE BLOCK:FALSE BLOCK
```

برای درک بهتر، مثال قبل را با این ساختار بازنویسی می‌کنیم:

```
echo("X is " . ($x%2==0?"even":"odd") . ".<BR/>\n");
```

ساختار while

از این ساختار برای تکرار بخشی از دستورات تا زمانی که یک شرط برقرار است، استفاده می‌شود و ساختار آن، به‌صورت زیر است:

```
while( CONDITION(s) )
{
      code block
}
```

در این ساختار، بلاک code تا زمانی که شرط یا شرایط دستور while برقرار باشند، تکرار می‌شود. بنابراین باید در بدنه بلاک مذکور، شرایطی درنظر گرفته‌شود که بالأخره در مرحله‌ای، شرط یا شرایط while نقض‌شود تا برنامه بتواند از حلقه تکرار، بیرون آید. درغیر این‌صورت، برنامه در یک حلقه بی‌نهایت گرفتار شده و نمی‌تواند کار خود را به‌درستی انجام‌دهد. مثال:

```
$i=1;
while($i<100)
{
      echo("Hello<BR/>\n");
      $i++;
}
```

با اجرای ساختار فوق، صدبار عبارت Hello در کد خروجی نوشته خواهد شد (و درنتیجه، به کاربر نشان‌داده می‌شود). اگر در ساختار قبل، دستور ;$i++ را ننویسیم، برنامه در حلقه گرفتار خواهد شد، زیرا $i همیشه 1 خواهد بود و همیشه، 1 از 100 کوچک‌تر است!

ساختار do...while

ساختار while، شرط خود را در ابتدا بررسی می‌کند و اگر در همان ابتدای کار، شرط برقرار نباشد، حتی یک‌بار هم حلقه اجرا نخواهد شد؛ اما گاهی‌اوقات لازم‌است که بلاک موردنظر، یک‌بار اجرا شود و سپس، شرط بررسی‌گردد و درصورت برقراربودن، مجدداً تکرار گردد. در چنین شرایطی، از ساختار do...while استفاده می‌کنیم:

```
do
{
      code block
}while( CONDITION(S) );
```

در این ساختار، شرط در پایان بررسی می‌شود.

ساختار for

این دستور، کامل‌ترین و درعین‌حال، پیچیده‌ترین ساختار تکرار در PHP است. در این ساختار، سه‌بخش در بدنه حلقه درنظر گرفته شده‌است: مقداردهی اولیه، شرط ادامه حلقه و افزایش/کاهش متغیر یا متغیرهای حلقه. ساختار کلی این دستور به‌صورت زیر است:

```
for( INITIALIZATION ; CONDITION(S) ; INCREMENT/DECREMENT )
{
      code block
}
```

روش اجرای این ساختار بدین‌صورت است که ابتدا قسمت INITIALIZATION اجرا می‌شود. سپس شرط یا شرایط قسمت CONDITIONS(S) بررسی‌شده و درصورت برقراربودن، code block اجرا می‌گردد. پس‌از پایان اجرای code block، قسمت INCREMENT/DECREMENT اجرا می‌گردد و مجدداً قسمت CONDITION(S) بررسی می‌گردد. درصورتی‌که هنوز شرایط برقرار باشند، حلقه مجدداً تکرار می‌گردد و این مراحل، تا زمانی که بالأخره، قسمت CONDITION(S) نقض‌شود، تکرار خواهد شد. برای درک بهتر، به همان مثال while که صدبار پیغام Hello را در کد خروجی می‌نوشت، دقت‌کنید که با ساختار for بازنویسی شده‌است:

```
for($i=1;$i<=100;$i++)
{
	echo("Hello<BR/>\n");
}
```

البته می‌توان هرکدام از قسمت‌های فوق را نادیده‌گرفت؛ مشروط بر آنکه در اجرای حلقه مشکلی پیش نیاید:

```
$i=1;
for(;$i<=100;)
{
	echo("Hello<BR/>\n");
	$i++;
}
```

البته این شکل استفاده از ساختار for، معمول نیست.

دستورات break; و continue; در حلقه‌های تکرار

کاربرد دستور break; در حلقه‌های تکرار بدین‌صورت است که هرگاه برنامه به دستور فوق برسد، بدون توجه به اتمام قانونی حلقه (با نقض‌شدن شرط تکرار حلقه)، از آن خارج می‌شود و برنامه را از اولین دستور بعد از حلقه، ادامه می‌دهد. دستور continue; نیز موجب می‌شود که PHP، تا انتهای بلاک کد را نادیده گرفته و به‌سراغ تکرار بعدی حلقه برود. مثالی از کاربرد دستور break; را در ادامه ملاحظه می‌کنید:

```
$i=1;
while(true)
{
	echo("$i<BR/>\n");
	$i++;
	if($i>100)
	{
		break;
	}
}
```

این حلقه (کمی عجیب)، اعداد از 1 تا 100 را در کد خروجی می‌نویسد. مثالی از کاربرد دستور continue; نیز به‌صورت زیر است:

```
for($i=1;$i<=100;$i++)
{
	if($i%2==0)
	{
		continue;
	}
	echo("$i<BR/>\n");
}
```

این حلقه نیز اعداد فرد از 1 تا 100 را در کد خروجی می‌نویسد. دقت‌کنید که چگونه درصورت زوج‌بودن $i، بقیه بدنه حلقه نادیده گرفته‌شده و به‌سراغ تکرار بعدی حلقه که یک $i فرد است، می‌رود.

آرایه چیست؟

در این جلسه می‌خواهیم روش استفاده از آرایه‌ها را بررسی‌کنیم. آرایه مجموعه‌ای از متغیرها است که دارای یک نام مشترک هستند و توسط اندیس از یکدیگر متمایز می‌شوند. برای مثال، متغیر $a را درنظر بگیرید:

$a						
Index	0	1	2	3	4	5
Value	12	7	4	2	1	290

در مثال فوق، متغیر $a یک آرایه با شش عنصر است که از 0 تا 5 شماره‌گذاری شده‌اند و مقادیر آنها به‌ترتیب برابر 12، 7، 4، 2، 1 و 290 است. حال فرض‌کنیم که می‌خواهیم محتویات خانه پنجم را ده‌برابر کنیم. برای این‌کار باید اینگونه بنویسیم:

```
$a[4]*=10;
```

تعریف یک آرایه در PHP

دقت‌کنید که برای تعریف آرایه در PHP، مشابه متغیرها، نوع آن ذکر نمی‌شود. بنابراین هر عنصر آرایه می‌تواند مقدار متفاوتی از هر نوع داده داشته‌باشد. ضمناً در PHP، طول آرایه نیز تعیین نمی‌شود. برای تعریف یک آرایه، کافی‌است نام آنرا همراه‌با کارکترهای [] ذکرکنیم:

```
$a[]=5;
```

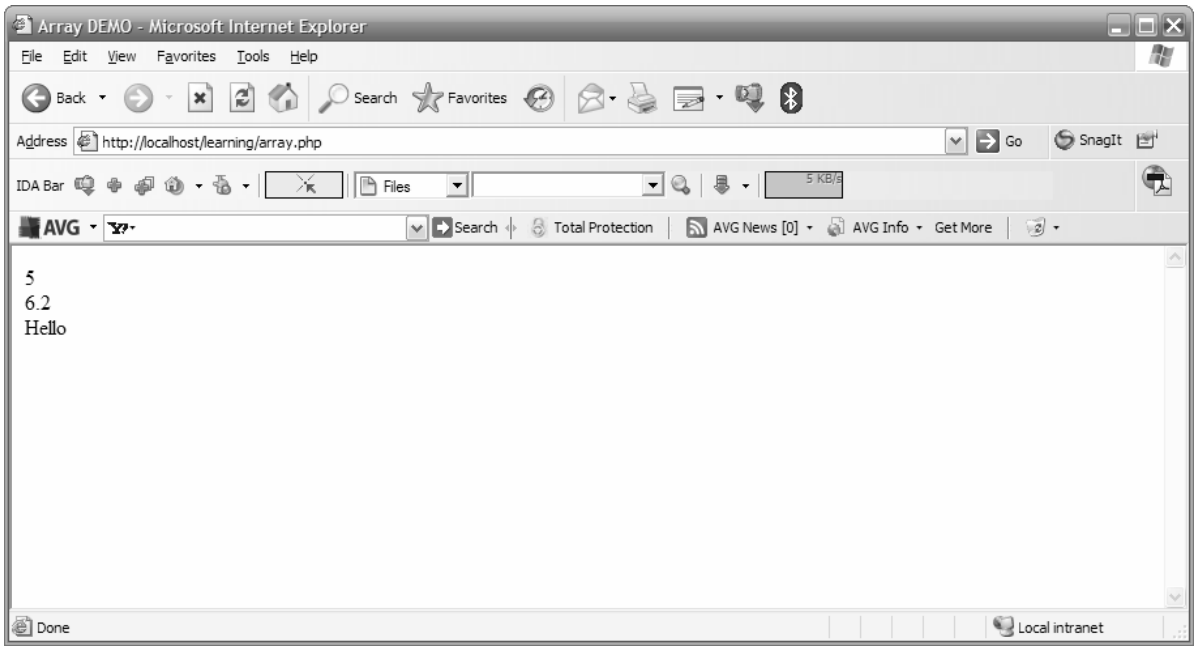
بدین‌ترتیب، یک عنصر با مقدار 5 در آرایه درج می‌شود. درصورتی‌که آرایه قبلاً دارای مقدار باشد، یک عنصر به آن افزوده‌شده و مقدار جدید، در آن ذخیره می‌شود؛ لیکن اگر آرایه موردنظر قبلاً وجود نداشته‌باشد، ایجاد می‌شود و لذا، اندیس عنصر جدید، 0 خواهد بود. برای افزودن عناصر بعدی نیز باید به‌همین‌ترتیب عمل‌کنیم:

```
$a[]=6.2;
$a[]="Hello";
```

برای درک بهتر، به کد زیر دقت‌کنید که ضمن تعریف آرایه فوق، عناصر آنرا در خروجی چاپ می‌کند (شماره خطوط را ننویسید):

```
01 <HTML>
02 <HEAD>
03 <TITLE>Array DEMO</TITLE>
04 </HEAD>
05 <BODY>
06 <?php
07      $a[]=5;
08      $a[]=6.2;
09      $a[]="Hello";
10      for($i=0;$i<3;$i++)
11      {
12          echo("$a[$i]<br/>\n");
13      }
14 ?>
15 </BODY>
16 </HTML>
```

خروجی برنامه فوق بدین‌صورت خواهد بود:



همان‌طور که ملاحظه می‌کنید، در PHP علاوه‌بر آنکه تعداد عناصر آرایه قابل تغییر است، نوع هر عنصر نیز می‌تواند با سایر عناصر متفاوت باشد. این امر، کاملاً طبیعی است؛ زیرا در PHP، نوع عناصر مشخص نمی‌شود و مقادیر آنها می‌تواند دارای هر نوع دلخواه باشد.

روش‌های دیگر تعریف آرایه

روشی که بیان‌کردیم، یکی‌از روش‌های ساده تعریف آرایه است که در آن، هیچ کنترلی برروی اندیس عناصر وجود ندارد و فقط می‌توان به‌ترتیب عناصر را درج‌کرده و سپس، مقدار هر عنصر را با ذکر اندیس آن، مورد دسترسی قرارداد. در ادامه، با چند روش برای مقداردهی عناصر یک آرایه آشنا می‌شویم که امکانات بیشتری را دراختیارمان قرار می‌دهند.

درج عناصر با ذکر شماره اندیس (اندیس عددی)

برای اینکه دقیقاً یک عنصر را در محل دلخواه از آرایه درج‌کنیم، به روش زیر عمل می‌کنیم:

```
$a[1]="PHP";
$a[3]=5;
$a[0]=6.2;
```

حال اگر توسط حلقه تکرار زیر، عناصر این آرایه را نمایش‌دهیم:

```
for($i=0;$i<4;$i++)
{
    echo("A[$i]=$a[$i]<br/>\n");
}
```

خروجی زیر تولید خواهد شد:

```
A[0]=6.2
A[1]=PHP
A[2]=
A[3]=5
```

همان‌طور که ملاحظه می‌کنید، عنصر A[2] مقدار ندارد و لذا، درمقابل آن، چیزی نوشته نمی‌شود. با استفاده از روش فوق، می‌توانیم هر عنصر را دقیقاً در همان اندیس دلخواه، درج‌کنیم.

استفاده از اندیس غیرعددی

در PHP، برای عناصر مختلف یک آرایه، اجباری برای استفاده از اندیس عددی وجود ندارد. برای استفاده از اندیس‌های غیرعددی (مثلاً رشته‌ها)، به‌روش زیر عمل می‌کنیم:

```
$stateCapital["SistanAndBaluchestan"]="Zahedan";
$stateCapital["Tehran"]="Tehran";
$stateCapital["Fars"]="Shiraz";
```

درصورت استفاده از چنین آرایه‌ای، دیگر نمی‌توان از حلقه for برای نمایش مقادیر آن، استفاده‌کرد. هرچند روش‌های دیگری برای این‌کار وجود دارد که در ادامه همین جلسه، با آنها آشنا خواهیم شد. برای نمایش مقادیر آرایه فوق، به‌صورت زیر عمل می‌کنیم:

```
echo($stateCapital["SistanAndBaluchestan"]."<br/>\n");
echo($stateCapital["Fars"]."<br/>\n");
echo($stateCapital["Tehran"]."<br/>\n");
```

که خروجی دستورات فوق به‌صورت زیر خواهد بود:

```
Zahedan
Shiraz
Tehran
```

استفاده از تابع array

یک روش آسان برای تعریف یک آرایه و مقداردهی عناصر آن به‌طور هم‌زمان، استفاده از تابع array است. خروجی این تابع، یک آرایه است و مقادیری که باید در عناصر آرایه درج‌شوند را به‌عنوان پارامتر دریافت می‌کند:

```
$cities=array("Zahedan","Shiraz","Tehran");
```

بدین‌ترتیب، آرایه‌ای به‌صورت زیر تعریف خواهد شد:

```
$cities[0]="Zahedan";
$cities[1]="Shiraz";
$cities[2]="Tehran";
```

در روش‌های قبل مشاهده‌کردیم که فقط در اولین روش (ذکرنکردن اندیس)، شماره‌گذاری عناصر یک آرایه از صفر شروع می‌شود و در سایر روش‌ها، می‌توانیم شماره‌گذاری عناصر را از اندیسی به‌جز صفر شروع‌کنیم:

```
$a[1]=5;
$a[2]="Ali";
//OR
$family["Ali"]="Hoseini";
$family["Reza"]="Kamali";
```

برای اینکه در تابع array، اندیس هر عنصر را ذکرکنیم، باید قبل از هر مقدار، اندیس آنرا به‌همراه علامت => بنویسیم. برای درک بهتر، دستورات فوق را با استفاده از تابع array بازنویسی می‌کنیم:

```
$a=array(1=>5,2=>"Ali");
//OR
$family=array("Ali"=>"Hoseini","Reza"=>"Kamali");
```

در این ساختار، اگر از اندیس عددی استفاده‌شود، می‌توانیم اندیس یک یا چند عنصر را ذکر نکنیم. در این‌صورت، اندیس آن عنصر، یک‌واحد بیشتر از اندیس عنصر قبل خواهد بود و اگر عنصر مربوطه، اولین عنصر باشد، اندیس آن برابر با صفر خواهد شد:

```
$numbers=array(10=>5,7,4,25=>12,9);
```

نتیجه اجرای دستور فوق، تعریف آرایه $numbers به‌صورت زیر است:

```
$numbers[10]=5;
$numbers[11]=7;
$numbers[12]=4;
$numbers[25]=12;
$numbers[26]=9;
```

استفاده از تابع range

این تابع، دو پارامتر از ورودی گرفته و محدوده بین آنها را به‌صورت یک آرایه، باز می‌گرداند. برای درک بهتر، به مثال زیر دقت‌کنید:

```
$years=range(2001,2010);
```

نتیجه اجرای این دستور، چنین است:

```
$years[0]=2001;
$years[1]=2002;
.
.
.
$years[8]=2009;
$years[9]=2010;
```

البته پارامترهای ورودی، می‌توانند از نوع کارکتر نیز باشند (البته فقط یک کارکتر، نه یک رشته). مثال:

```
$reverseSmallLetters=range("z","a");
```

که نتیجه آن، چنین است:

```
$reverseSmallLetters[0]="z";
$reverseSmallLetters[1]="y";
$reverseSmallLetters[2]="x";
.
.
.
$reverseSmallLetters[24]="b";
$reverseSmallLetters[25]="a";
```

نمایش محتویات آرایه‌ها

برای نمایش آرایه‌ها، روش‌های مختلفی وجود دارد. یکی‌از این روش‌ها که قبلاً با آن آشنا شدید، استفاده از حلقه for است که فقط درصورتی کاربرد دارد که اندیس عناصر آرایه، از نوع عددی باشد:

```
$a[1]=5;
$a[2]=8;
$a[3]=4;
for($i=1;$i<=3;$i++)
{
      echo("$a[$i]<br/>\n");
}
/*Output:
5
8
4
*/
```

نمایش آرایه به‌کمک تابع print_r

این تابع، یک آرایه را به‌عنوان پارامتر دریافت‌کرده و عناصر آنرا به‌ترتیب در خروجی می‌نویسد:

```
$c["SistanAndBaluchestan"]="Zahedan";
$c["Fars"]="Shiraz";
$c["Tehran"]="Tehran";
print_r($c);
/*Output:
Array
(
      [SistanAndBaluchestan] => Zahedan
      [Fars] => Shiraz
      [Tehran] => Tehran
)
*/
```

نمایش آرایه به‌کمک تابع var_dump

این تابع نیز ساختاری مشابه تابع print_r دارد، اما نتایج کامل‌تری ارائه می‌دهد:

```
$c["SistanAndBaluchestan"]="Zahedan";
$c["Fars"]="Shiraz";
$c["Tehran"]="Tehran";
var_dump($c);
/*Output:
array(3) {
      ["SistanAndBaluchestan"]=>
            string(7) "Zahedan"
      ["Fars"]=>
            string(6) "Shiraz"
      ["Tehran"]=>
            string(6) "Tehran"
}
*/
```

نکته: دقت‌کنید که اگر از PHP در وب استفاده می‌کنید، خروجی تولیدشده، توسط HTML نمایش داده خواهد شد. درنتیجه، شکل منظمی را که در مثال‌های فوق ملاحظه می‌کنید، نخواهد داشت و تماماً در یک سطر طولانی نوشته می‌شود. برای آنکه HTML را وادار کنید خروجی PHP را به‌همان‌شکل که دریافت می‌کند، نمایش‌دهد و از قالب HTML برای این‌کار استفاده نکند، از تگ pre استفاده‌کنید؛ بدین‌ترتیب که تگ مذکور را قبل‌از دستورات print_r یا var_dump، بازکرده و بعد از آن دستورات، ببندید:

```
echo("<pre>\n");
print_r($c);
echo("</pre>\n");
//OR
echo("<pre>\n");
var_dump($c);
echo("</pre>\n");
```

نمایش آرایه با استفاده از ساختار تکرار foreach

ساختار تکرار دیگری به‌نام foreach در PHP وجود دارد که بدون اهمیت به نوع اندیس عناصر آرایه، کل عناصر آنرا پیمایش می‌کند و کارهای خواسته‌شده در بدنه حلقه را برروی همه آنها انجام می‌دهد. برای درک بهتر این ساختار، نحوه عملکرد آنرا در مثال زیر مشاهده‌کنید:

```
$capital["S_AND_B"]="Zahedan";
$capital["Fars"]="Shiraz";
$capital["Tehran"]="Tehran";
foreach($capital as $city)
{
      echo("$city<br/>\n");
}
/*Output:
Zahedan
Shiraz
Tehran
*/
```

در این ساختار، به‌ترتیب عناصر آرایه $capital با نام موقت $city پیمایش می‌شوند و دستور echo برروی آنها اعمال می‌گردد. از آنجا که هربار، $city برابر با یکی از عناصر آرایه $capital خواهد بود، همه عناصر آرایه مذکور، در خروجی ظاهر می‌گردند. این ساختار، از آنجا که با اندیس آرایه کار نمی‌کند، بدون اهمیت به نوع اندیس یک آرایه (عددی یا رشته‌ای)، قابل استفاده است.

ویرایش آرایه‌ها

برای تغییر مقدار یک عنصر از آرایه، کافی‌است توسط اندیس، به آن عنصر اشاره‌کرده و مقدار جدید را در آن ذخیره‌کنیم:

```
01 $stateCount["Iran"]=25;
02 echo($stateCount["Iran"]."<br/>\n");
//Output: 25
03 $ stateCount ["Iran"]=29;
04 echo($stateCount["Iran"]."<br/>\n");
//Output: 29
```

دستور 03، مقدار اندیس "Iran" را در آرایه $stateCount، از 25 به 29 تغییر می‌دهد.

به‌علاوه، در PHP می‌توانید یک کپی سریع از تمامی مقادیر آرایه را درون یک آرایه جدید قراردهید:

```
$arrayCopy=$myArray;
```

با اجرای دستور فوق، یک کپی از آرایه $myArray به‌نام $arrayCopy با تعداد عناصر یکسان، اندیس‌های مشابه و مقادیر مساوی برای عناصر آرایه‌ها، ایجاد خواهد شد.

حذف عناصر از آرایه

ممکن‌است گاهی اوقات نیاز به حذف کامل یک عنصر از آرایه داشته‌باشید. برای مثال، آرایه زیر را درنظر بگیرید:

```
$colors=array("red","green","blue","pink","yellow");
```

این آرایه، شامل پنج مقدار است. حال فرض‌کنید که دیگر به رنگ صورتی علاقه‌ای ندارید و می‌خواهید آنرا از آرایه خود حذف‌کنید. شاید در نگاه اول، دستور زیر مناسب باشد:

```
$colors[3]="";
```

اما دقت کنید که اگرچه دستور فوق، عنصر چهارم آرایه $colors را برابر با یک رشته خالی قرار می‌دهد؛ اما آنرا از آرایه حذف نمی‌کند (هنوز هم آرایه‌ای با پنج عنصر دارید که یکی‌از عناصر آن، یک رشته خالی است). برای حذف کامل یک عنصر از آرایه، از دستور unset استفاده می‌کنیم:

```
unset($colors[3]);
```

اکنون اگر آرایه خود را با هرکدام از ساختارهایی که ذکرشد، نمایش‌دهیم، نتیجه زیر حاصل خواهد شد:

```
$colors[0]="red"
$colors[1]="green"
$colors[2]="blue"
$colors[4]="yellow"
```

دقت‌کنید که با استفاده از دستور unset، شماره اندیس عناصر تغییر نمی‌کند و فقط، عنصر موردنظر از آرایه حذف می‌شود. ضمناً درصورت حذف تمامی عناصر یک آرایه با دستور unset، خود آرایه ازبین نمی‌رود و کماکان یک آرایه خالی دراختیار خواهید داشت. برای حذف آرایه، کافی‌است آنرا همانند سایر عناصر آن، حذف‌کنید:

```
unset($colors);
```

نکته: دستور unset مخصوص آرایه‌ها نیست و برای حذف متغیرهای معمولی نیز قابل استفاده است.

استفاده از توابع در PHP

دومین مفهوم مهم در هر زبان برنامه‌نویسی (بعد از متغیر)، تابع است. برنامه‌نویس خوب، کسی‌است که برنامه را با استفاده از توابع مختلف می‌نویسد. توابع به برنامه‌نویس در درک بهتر برنامه و اشکال‌زدایی و تغییر آن، کمک می‌کنند. برای مثال، حالتی را درنظر بگیرید که قرار است اطلاعات خاصی در خروجی نشان‌داده شود. حال اگر این اطلاعات را بخواهیم در چند قسمت از صفحه، نمایش دهیم، باید در مکان‌های موردنظر، دستورات محاسبه و نمایش اطلاعات موردنظر را درج‌کنیم. این مسأله، ممکن‌است ما را با مشکل مواجه‌کند؛ زیرا احتمال بروز خطا، به‌تعداد دفعات تکرار کد مربوطه، افزایش خواهد یافت. مثلاً اگر کد موردنظر دارای خطا باشد و در پنج‌قسمت از برنامه، آن کد را کپی کرده‌باشیم، باید هر پنج قسمت، اصلاح‌گردد. ضمناً اگر بخواهیم شیوه نمایش اطلاعات را تغییر دهیم و یا هرگونه ویرایش دیگری رو کد موردنظر داشته‌باشیم، باید هر پنج‌قسمت اصلاح‌گردد. راه‌حل مناسب برای این‌کار، استفاده از توابع است؛ بدین‌صورت که کد مربوطه را درون یک تابع نوشته و هر زمان به آن نیاز داشتیم، تابع موردنظر را فراخوانی کنیم. این‌کار علاوه‌بر افزایش خوانایی، موجب سهولت در اصلاح و تغییر نیز می‌شوند؛ زیرا با ویرایش تابع، تمامی قسمت‌هایی از برنامه که آنرا فراخوانی کرده‌اند نیز به‌طور خودکار، اصلاح شده یا تغییر خواهند یافت. به‌علاوه، کد برنامه نیز کوتاه‌تر می‌شود.

تعریف تابع

برای تعریف یک تابع در PHP از کلمه کلیدی function استفاده می‌کنیم. سپس، نام تابع و علامت پرانتز باز (() و در ادامه، اسامی پارامترهای ورودی تابع را می‌نویسیم که با کاما (,) از هم جدا می‌شوند (البته درصورت وجود پارامتر ورودی) و درنهایت، علامت پرانتز بسته (() را ذکر می‌کنیم. بدین‌ترتیب، عنوان تابع ایجاد می‌شود. حال کافی‌است بین دو علامت آکولاد باز ({) و آکولاد بسته (}), بدنه تابع را بنویسیم. مثال:

```
function greeting($name,$type)
{
	$greeting="";
	switch($type)
	{
		case 0:
			$greeting="Welcome";
			break;
		case 1:
			$greeting="GoodBye";
			break;
		default:
			$greeting="";
			break;
	}
	echo("<P ALIGN=\"CENTER\">$greeting $name.</P>\n");
}
```

در مثال فوق، تابع greeting، دو پارامتر ورودی به‌نام $name و $type دریافت می‌کند و آنرا درون یک پاراگراف و در وسط سطر، براساس $type، عبارت Welcome یا GoodBye ویا هیچ‌کدام را همراه‌با عبارت $name، درج می‌کند. به‌عنوان مثالی از فراخوانی تابع فوق، به کد زیر دقت‌کنید:

```
greeting("PHP User",0);
//Output: "Welcome PHP User."
greeting("PHP User",1);
//Output: "GoodBye PHP User."
```

اگر یک تابع، پارامتر ورودی نداشته‌باشد، بعد از نوشتن پرانتز باز، بلافاصله پرانتز بسته را درج می‌کنیم.

نکته: دقت کنید که اگر متغیری را درون بدنه یک تابع تعریف‌کنید، خارج از آن تابع، نمی‌توانید به مقدار آن دسترسی داشته‌باشید. به متغیرهای داخلی توابع، متغیرهای محلی (Local Variables) می‌گویند.

تعیین خروجی یک تابع

توابع نه‌تنها می‌توانند کارهای خاصی را انجام‌دهند، بلکه این امکان نیز وجود دارد که پس‌از انجام کارهای خود، یک‌مقدار را نیز به‌عنوان نتیجه، بازگردانند. به این مقدار، خروجی یا مقدار بازگشتی تابع می‌گویند. برای تعیین مقدار بازگشتی تابع، از دستور return و در ادامه، مقداری که باید بازگردانده‌شود، استفاده می‌کنیم:

```
function sum($a,$b)
{
	return $a+$b;
	//OR return($a+$b);
}
```

نکته: دقت‌کنید که return یک تابع نیست، بلکه یک کلمه کلیدی است؛ لذا قراردادن عبارت بعد از آن درون پرانتز، کاملاً اختیاری است. ضمناً به‌محض اجراشدن دستور return، برنامه از تابع خارج‌شده و مقدار مشخص‌شده را باز می‌گرداند. به‌عبارت دیگر، دستورات بعد از return، اجرا نخواهند شد. می‌توانید از این امر، به‌نفع خود استفاده‌کنید:

```
function divide($x,$y)
{
	if($y==0)
	{
		return "Error";
	}
	return $x/$y;
}
```

در مثال فوق، اگر $y برابر با صفر باشد، عبارت Error بازگردانده‌شده و دیگر دستور return $x/$y; اجرا نمی‌شود. ضمناً اگر نخواهید تابع شما مقداری را بازگرداند، می‌توانید از دستور return استفاده نکنید؛ هرچند هنوز هم شکل خاصی از دستور return وجود دارد که برای خروج ضروری از تابع، می‌توانید از آن استفاده‌کنید:

```
function displayDivide($x,$y)
{
	if($y==0)
	{
		return;
	}
	echo(($x/$y)."<br/>\n");
}
```

محدوده متغیر (Variable Scope)

در PHP، هر متغیری که خارج از توابع تعریف‌شود، متغیر سراسری است و در تمام بدنه برنامه معتبر می‌باشد. درعوض، متغیرهای درون توابع، محلی هستند و فقط درون تابعی که در آن تعریف شده‌اند، اعتبار دارند. درصورتی‌که درون یک تابع، متغیری هم‌نام با یکی از متغیرهای سراسری برنامه تعریف‌کنید، دو متغیر با آن نام وجود خواهد داشت: یک متغیر سراسری و یک متغیر محلی که هرکدام، مقادیر خودشان را خواهندداشت.

برای درک بهتر، به مثال زیر دقت‌کنید:

```
$x=5;
function test()
{
	$x=7;
	echo("$x<br/>\n");
}
test();
//Output: 7
echo("$x<br/>\n");
//Output: 5
```

برای دسترسی به هرکدام از متغیرهای سراسری در بدنه یک تابع، باید متغیری هم‌نام با همان متغیر، به‌کمک کلمه کلیدی global تعریف‌کنید تا متغیر درون تابع، همان متغیر سراسری باشد (نه یک متغیر محلی). مثال:

```
$x=5;
function test()
{
	global $x;
	$x=7;
	echo("$x<br/>\n");
}
test();
//Output: 7
echo("$x<br/>\n");
//Output: 7
```

متغیرهای static

به این تابع دقت‌کنید:

```
function test()
{
	$callCount=0;
	$callCount++;
	echo("This function is called $callCount time(s).<br/>\n");
}
```

با کمی دقت متوجه می‌شویم که به‌جای $callCount در دستور echo، همیشه عدد یک نوشته می‌شود. اگر بخواهیم تعداد فراخوانی‌های این تابع را به‌دست آوریم، هیچ راهی به‌جز استفاده از متغیرهای static نداریم. متغیرهای static دقیقاً همانند متغیرهای معمولی هستند؛ اما با خروج از تابع، آخرین مقدار خود را حفظ می‌کنند و مجدداً مقداردهی اولیه نخواهند شد:

```
function test()
{
	static $callCount=0;
	$callCount++;
	echo("This function is called $callCount time(s).<br/>\n");
}
```

در کد فوق، به‌دلیل آنکه متغیر $callCount از نوع static تعریف شده‌است، در اولین فراخوانی تابع test مقدار صفر را در خود ذخیره‌کرده و سپس یک‌واحد به آن افزوده می‌شود و تعداد دفعات اجرای تابع، یک‌بار ذکر خواهد شد. نکته مهم، فراخوانی‌های بعدی تابع است که در آنها، دیگر $callCount با صفر مقداردهی اولیه نمی‌شود و آخرین مقدار خود را خواهد داشت و هربار، یک‌واحد به آن افزوده می‌شود. درنتیجه، دستور echo، تعداد دفعات فراخوانی تابع را به‌طور دقیق ذکر خواهد کرد.

پارامترهای مقداری و ارجاعی

درصورتی‌که پارامترهای یک تابع را به روشی که تاکنون بیان‌کردیم، تعریف‌کنید، با ارسال هر متغیر برای تابع مربوطه، فقط یک کپی از مقدار آن، درون پارامتر ذخیره خواهد شد و درصورت تغییر پارامتر درون بدنه تابع، متغیر اصلی بدون تغییر خواهد ماند. مثال:

```
$x=5;
function test($num)
{
	$num++;
}
test($x);
echo("$x<br/>\n");
//Output: 5
```

در اینجا، فقط یک کپی از مقدار $x درون $num قرارگرفته و سپس، یک‌واحد افزایش داده شده‌است؛ اما متغیر $x بدون تغییر باقی خواهد ماند. به این‌گونه پارامترها، مقداری می‌گویند. حال به کد زیر دقت‌کنید:

```
$x=5;
function test(&$num)
{
	$num++;
}
test($x);
echo("$x<br/>\n");
//Output: 6
```

در اینجا، به‌دلیل استفاده از کارکتر & قبل‌از پارامتر $num، این پارامتر یک نام مستعار برای متغیرهایی خواهد شد که برای تابع، ارسال می‌شوند (در اینجا، $x). بنابراین دستور $num++; دقیقاً معادل دستور $x++; است؛ زیرا $num درحقیقت یک نام مستعار برای $x است. به این‌گونه پارامترها، ارجاعی می‌گویند.

در PHP این امکان وجود دارد که از یک تابع، در برخی موارد به‌صورت مقداری و در موارد دیگر، به‌صورت ارجاعی استفاده‌شود. برای این‌کار، به‌جای تعریف پارامتر ازنوع ارجاعی، در زمان فراخوانی تابع، متغیر موردنظر را به‌صورت ارجاعی برای آن ارسال می‌کنیم:

```
$x=5;
$y=5;
function test($num)
{
	$num++;
}
test($x);
text(&$y);
echo("$x<br/>\n");
//Output: 5
echo("$y<br/>\n");
//Output: 6
```

در اینجا، $x به‌صورت مقداری و $y به‌صورت ارجاعی به تابع test ارسال شده‌است. بنابراین دستور $num++; مقدار $x را افزایش نمی‌دهد؛ اما مقدار $y دراثر اجرای دستور فوق، افزایش خواهد یافت.

ارتباط با کاربر

تاکنون هر دستوری که نوشته‌ایم، کارهای خاصی را انجام می‌داد؛ اما کاربر (کسی که صفحه وب را مشاهده می‌کند)، هیچ کنترلی برروی برنامه نداشت. درواقع تاکنون برنامه‌های ما به همان صورت ایستا (Static) که در HTML وجود داشت، طراحی می‌شدند (باز هم برای تغییر خروجی، باید کد را تغییر می‌دادیم). حال می‌خواهیم روش ارتباط با کاربر را توضیح‌دهیم تا بتوانیم از کاربر اطلاعات دلخواه را دریافت‌کرده و براساس آن، صفحاتی طراحی‌کنیم که بدون نیاز به بازنویسی کد، اطلاعات مختلفی را (بسته به ورودی‌های کاربر) نمایش‌دهد.

فرم‌ها در زبان HTML

برای برقراری ارتباط با کاربر، از فرم‌ها استفاده می‌شود. یک فرم می‌تواند شامل موارد مختلف همچون کادرهای متن معمولی، کادرهای متن رمز عبور (که در آنها به‌جای متن تایپ‌شده، کارکتر * یا ● نمایش داده می‌شود)، کادرهای انتخاب (که کاربر می‌تواند گزینه‌های دلخواه خود را فعال ☑ یا غیرفعال ☐ نماید)، دکمه‌های انتخاب (که کاربر فقط می‌تواند یکی از گزینه‌های موجود را انتخاب‌کند ⊙ و بقیه گزینه‌ها با انتخاب هر گزینه، از انتخاب خارج می‌شوند ○)، کادرهای متن مخفی، دکمه‌های ارسال اطلاعات و... باشد. برای ایجاد یک فرم، از تگ form استفاده می‌شود. این تگ، دارای دو پارامتر مهم به‌نام‌های action و method است. پارامتر action مشخص‌کننده صفحه‌ای است که بعد از تکمیل اطلاعات فرم، مقادیر واردشده توسط کاربر به آن صفحه ارسال می‌شود. پارامتر method نیز روش ارسال اطلاعات را مشخص می‌کند که می‌تواند یکی‌از مقادیر get یا post را بپذیرد. برای درک بهتر، به مثال زیر دقت‌کنید:

```
<FORM ACTION="result.php" METHOD=GET>
...
</FORM>
```

بدین‌ترتیب، فرمی تشکیل می‌شود که درصورت کلیک‌کردن کاربر برروی دکمه ارسال اطلاعات، کلیه مقادیر واردشده یا انتخاب‌شده توسط وی، به صفحه result.php ارسال خواهد شد. درمورد تفاوت روش‌های ارسال get و post بهتر است پس‌از آشنایی با تعدادی از کنترل‌های ورودی کاربر، صحبت کنیم. تمامی کنترل‌های ورودی کاربر، به‌کمک تگ input ایجاد می‌شوند و خاصیت‌های مختلف این تگ، نوع کنترل ورودی را تعیین می‌کند.

کادر متن

برای ایجاد یک کادر متن که کاربر بتواند متن دلخواه خود را درون آن بنویسد، از تگ input بدین‌صورت استفاده می‌شود که خاصیت type آن با مقدار text و خاصیت name آن با نام دلخواه (برای دسترسی به مقدار آن در صفحه مقصد) تنظیم می‌گردد. ضمناً خاصیت value نیز در این کنترل وجود دارد که تنظیم‌کردن آن اختیاری است و درصورت تنظیم‌کردن با هر رشته دلخواه، درزمان ظاهرشدن فرم، از قبل درون آن نوشته می‌شود. مثال:

```
<INPUT TYPE="text" NAME="uname" VALUE="alireza"/>
```

alireza

بدین‌ترتیب، یک کادر متن به‌نام uname (این نام در برنامه به‌کار می‌رود و کاربر آنرا نمی‌بیند) برای واردکردن عبارت دلخواه توسط کاربر، ایجاد خواهد شد.

نکته: از آنجا که تگ input تمامی پارامترهای موردنیاز خود را بعد از کارکتر > و تا قبل از رسیدن به کارکتر <، دریافت می‌کند، لذا به‌جای بستن آن با </INPUT>، باید از /> به‌جای < استفاده‌کنیم.

کادر متن رمز عبور

این کادر متن، دقیقاً مشابه کادر متن معمولی است؛ با این تفاوت که هرچه کاربر درون آن تایپ‌کند، مخفی‌بوده و به‌جای کارکترهای تایپ‌شده، کارکتر * یا ● (براساس نوع مرورگر و سیستم‌عامل) نمایش داده می‌شود. برای ایجاد این کادر متن، مشابه کادر متن معمولی عمل می‌کنیم و فقط پارامتر type را با password (به‌جای text) مقداردهی می‌کنیم. در اینجا نیز خاصیت value اختیاری است و اگر تنظیم‌شود، در زمان ظاهرشدن فرم، از قبل درون آن نوشته خواهد شد. مثال:

```
<INPUT TYPE="password" NAME="pass" VALUE="1234"/>
```

بدین‌ترتیب، یک کادر متن رمز عبور به‌نام pass (پنهان از دید کاربر) برای واردکردن رمز عبور دلخواه دراختیار کاربر قرار می‌گیرد.

کادر انتخاب

از این کنترل در مواقعی استفاده می‌شود که بخواهیم چند گزینه در اختیار کاربر قراردهیم تا بتواند یک یا چند مورد از بین آنها انتخاب‌کند. برای این‌کار، باید خاصیت type را برروی checkbox و خاصیت name را با نام دلخواه تنظیم‌کنیم. درصورت تمایل، می‌توان خاصیت value را نیز تنظیم‌نمود که مشخص‌کننده مقداری است که درصورت انتخاب‌شدن گزینه موردنظر، به صفحه مقصد ارسال خواهد شد (اگر این خاصیت تنظیم نشود، نام کنترل به‌عنوان مقدار خاصیت value درنظر گرفته خواهد شد. اگر بخواهیم در لحظه ظاهرشدن فرم (بازشدن صفحه)، گزینه موردنظر انتخاب شده باشد، باید خاصیت checked را با مقدار checked تنظیم‌کنیم.

مثال:

```
<INPUT TYPE="checkbox" NAME="article" VALUE="art"/>
<INPUT TYPE="checkbox" NAME="book" VALUE="book" CHECKED="checked"/>
```

دکمه انتخاب

از این کنترل در مواقعی استفاده می‌کنیم که قصد داشته‌باشیم چند گزینه در اختیار کاربر قراردهیم؛ اما کاربر در هر لحظه، فقط بتواند یک گزینه را انتخاب‌کند. در استفاده از این کنترل باید دقت‌کنید که خاصیت name همه کنترل‌هایی که مربوط‌به‌هم هستند، باید یکسان باشد تا در هر لحظه، فقط یکی از آنها قابل انتخاب باشد. برای تعریف این کنترل، خاصیت type باید برروی radio و name با نام دلخواه و value با عبارتی که درصورت انتخاب دکمه مربوطه، به‌عنوان مقدار برای صفحه مقصد ارسال خواهد شد، تنظیم‌شود. در اینجا نیز خاصیت checked درصورتی‌که با مقدار checked تنظیم‌شود، موجب انتخاب‌شدن دکمه مربوطه در زمان ظاهرشدن فرم خواهد شد:

```
<INPUT TYPE="radio" NAME="gender" VALUE="Male" CHECKED="checked"/>
<INPUT TYPE="radio" NAME="gender" VALUE="Female"/>
```

کادر متن مخفی

این کنترل دقیقاً مشابه کنترل کادر متن است، با این تفاوت که کاربر آنرا نمی‌بیند و نمی‌تواند محتویات آنرا تغییر دهد! شاید در نگاه اول، کاربرد این کنترل مشخص نباشد؛ اما شرایطی را درنظر بگیرید که قصد دارید مقدار خاصی را بدون اینکه کاربر بداند، برای صفحه مقصد ارسال‌کنید. در چنین حالتی، می‌توانید از کادر متن مخفی استفاده‌کنید. برای این‌کار، خاصیت type را با مقدار hidden و خاصیت value را با مقدار موردنظر تنظیم‌نمایید. مثال:

```
<INPUT TYPE="hidden" NAME="count" VALUE="3"/>
```

در مثال فوق، هر زمان که اطلاعات به صفحه مقصد ارسال‌شود، مقدار 3 نیز با نام count برای آن‌صفحه، ارسال خواهد شد.

دکمه ارسال اطلاعات

این کنترل، کلید اصلی فرم‌ها است و وظیفه بازکردن صفحه مقصد و ارسال مقادیر تایپ‌شده یا انتخاب‌شده توسط کاربر، به آن صفحه را برعهده دارد. برای استفاده از این دکمه، خاصیت type را باید برروی submit، name را با نام دلخواه و value را با عبارتی که قصددارید روی دکمه نمایش داده‌شود، تنظیم‌کنید. مثال:

```
<INPUT TYPE="submit" NAME="submitbutton" VALUE="Send"/>
```

Send

بدین‌ترتیب، با کلیک برروی این دکمه، صفحه مقصد بازشده و کلیه اطلاعاتی که کاربر وارد کرده‌است، برای وی ارسال خواهد شد.

نکته: انواع دیگری از کنترل‌های ورودی نیز وجود دارند که به‌دلیل کاربرد کمتر، از توضیح درباره آنها خودداری می‌کنیم.

خاصیت ID و تگ LABEL

تمامی کنترل‌های فوق، دارای یک خاصیت اختیاری به‌نام id هستند که می‌تواند با عبارت دلخواه تنظیم‌شود. این خاصیت برخلاف خاصیت name که در صفحه مقصد برای دسترسی به کنترل مربوطه (و مقدار آن)، به‌کار می‌رود، در صفحه مبدأ (صفحه‌ای که حاوی فرم است)، برای دسترسی به کنترل کاربرد دارد. یکی‌از مهم‌ترین کاربردهای این خاصیت، استفاده از تگ LABEL است. تگ LABEL دو کاربرد اساسی دارد: نمایش متن توضیحی درکنار کنترل و همچنین، اختصاص کلید میانبر صفحه‌کلید به کنترل. شاید بگویید: متن توضیحی را می‌توان به‌سادگی درکنار کنترل و خارج‌از تگ INPUT نوشت. برای مثال:

```
Username: <INPUT TYPE="text" NAME="uname"/>
```

Username:

این گفته، کاملاً صحیح است، اما تفاوت درج متن توضیحی به‌کمک LABEL با درج متن به‌روش تایپ مستقیم، در آن‌است که متن توضیحی به‌کمک LABEL، کاملاً به کنترل متصل است و اگر کاربر برروی آن کلیک‌کند، کنترل متناظر با آن، انتخاب می‌شود. این امر برای کنترل‌هایی که ذاتاً کوچک هستند (مثل کادرها و دکمه‌های انتخاب) بسیار سودمند است؛ زیرا باعث سهولت انتخاب آنها توسط ماوس می‌شود. برای این‌که یک متن را به یک کنترل نسبت‌دهیم، از تگ LABEL و تنظیم خاصیت for با مقدار خاصیت id کنترل مربوطه، استفاده می‌کنیم:

```
<LABEL FOR="un">Username: </LABEL><INPUT TYPE="text" NAME="uname" ID="un"/>
```

کاربرد دوم تگ LABEL نیز بسیار سودمند است. اگر بازدیدکننده صفحه وب شما، علاقمند به استفاده از صفحه‌کلید باشد، معمولاً به‌دنبال کلیدهای میانبر برای انتخاب گزینه‌ها خواهد بود و تمایل زیادی به استفاده از ماوس نخواهدداشت. به‌کمک خاصیت accesskey تگ LABEL، می‌توانید یک کلید میانبر برای کنترل خود اختصاص‌دهید. برای مثال:

```
<INPUT TYPE="checkbox" NAME="married" ID="maritalstatus"/>
<LABEL FOR="maritalstatus" ACCESSKEY="m"><U>M</U>arried</LABEL>
```

☐ Married

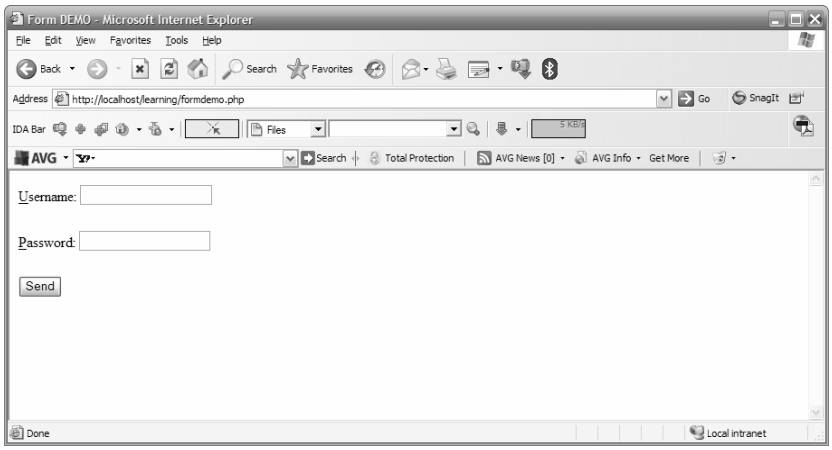
در مثال فوق، دقت‌کنید که برای انتخاب گزینه Married، می‌توانید هم برروی کادر انتخاب و هم برروی نوشته Married کلیک‌کنید. همچنین، امکان انتخاب این گزینه توسط کلیدهای Alt+M وجود دارد. البته کلید میانبر، به‌دلیل خاصیت accesskey تگ LABEL ایجاد شده‌است و برای اطلاع کاربر از این ویژگی، حرف M از کلمه Married، به‌کمک تگ U به‌صورت زیرخط‌دار نمایش داده شده‌است.

تفاوت متدهای GET و POST در فرم

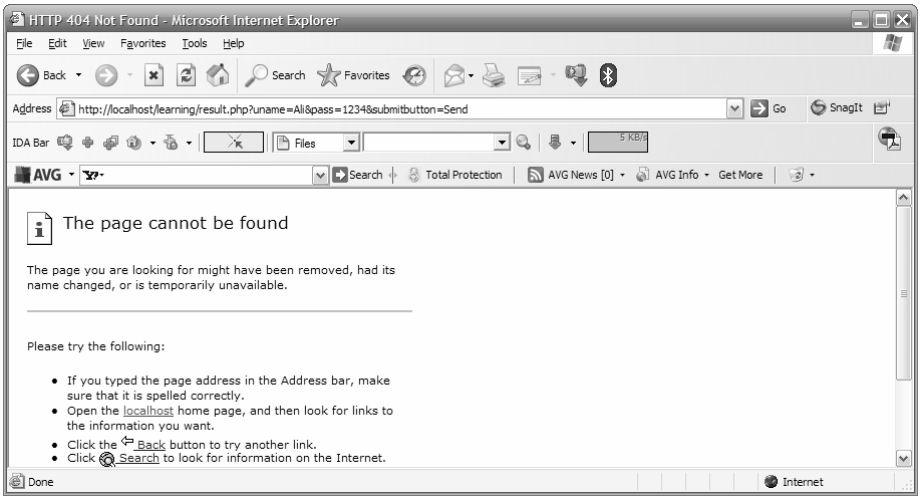
حال که به‌اندازه کافی برای ایجاد یک فرم، اطلاعات دراختیار داریم، اجازه‌دهید که تفاوت متدهای GET و POST را در عمل مشاهده‌کنیم. به فرم زیر دقت‌کنید:

```
<FORM ACTION="result.php" METHOD=GET>
<LABEL FOR="un" ACCESSKEY="u"><U>U</U>sername: </LABEL>
<INPUT TYPE="text" NAME="uname" ID="un"/>
<BR/><BR/>
<LABEL FOR="pw" ACCESSKEY="p"><U>P</U>assword: </LABEL>
<INPUT TYPE="password" NAME="pass" ID="pw"/>
<BR/><BR/>
<INPUT TYPE="submit" NAME="submitbutton" VALUE="Send"/>
</FORM>
```

نتیجه اجرای این کد، صفحه‌ای به‌صورت زیر خواهد بود:



درصورتی‌که کاربر برروی دکمه Send کلیک‌کند، صفحه result.php بازشده و هرآنچه کاربر وارد کرده‌است، برای آن، ارسال می‌شود. برای مثال، در کادر Username عبارت Ali و در کادر Password عبارت 1234 را واردکرده و برروی دکمه Send کلیک می‌کنیم. پنجره مرورگر مطابق تصویر زیر تغییر می‌کند:



علت نمایش خطای 404 آن‌است که صفحه result.php وجود ندارد و مرورگر نمی‌تواند آنرا بازکند (این‌امر کاملاً طبیعی است: ما هنوز صفحه مقصد را طراحی نکرده‌ایم!).

فعلاً با محتویات صفحه کار نداریم. به آدرس صفحه دقت‌کنید:

```
http://localhost/learning/result.php?uname=Ali&pass=1234&submitbutton=Send
```

اگر به‌دقت به این آدرس دقت‌کنید، بخش‌های مختلفی را در آن ملاحظه خواهید کرد:

```
http://localhost/learning/result.php
?
uname=Ali
&
pass=1234
&
submitbutton=Send
```

بخش اول، آدرس صفحه مقصد را مشخص می‌کند. در ادامه، یک علامت سؤال درج‌شده و مقادیر واردشده توسط کاربر، همراه‌با نام آنها ذکر می‌شوند (اگر بیش‌از یک مقدار توسط کاربر وارد شده‌باشد، با علامت & از یکدیگر جدا می‌شوند). درنهایت، دکمه‌ای که کاربر برروی آن کلیک‌کرده‌است، مشخص می‌شود. دقت‌کنید که می‌توانید بیش‌از یک دکمه Submit داشته‌باشید که خاصیت name آنها مشترک و خاصیت value آنها متفاوت باشد. بدین‌ترتیب، در صفحه مقصد مشخص خواهد شد که کاربر برروی کدام دکمه کلیک کرده‌است و می‌توان براساس انتخاب کاربر، کارهای مختلفی انجام‌داد.

نکته اصلی در آدرس فوق آن‌است که امنیت در روش GET بسیار ضعیف است؛ زیرا هرآنچه کاربر وارد کرده‌است، مستقیماً در آدرس صفحه قابل مشاهده است. درحقیقت در متد GET، مقادیر واردشده توسط کاربر، ازطریق آدرس صفحه ارسال می‌شود. این مسأله، درصورتی‌که اطلاعات ارسال‌شده محرمانه باشد (مثل رمز عبور کاربر)، باعث ازبین‌رفتن امنیت می‌شود و در چنین مواردی، بهتر است از متد POST استفاده‌شود که در ادامه، توضیح داده خواهد شد.

قبل از آنکه به‌سراغ متد POST برویم، خوب است با مزایای روش GET نیز آشنا شویم. فرض‌کنید که یک وبلاگ طراحی کرده‌اید که شامل مطالب متنوع درمورد نجوم، کامپیوتر، الکترونیک، هواشناسی، سرگرمی و... است. حال اگر مطلب خاصی داشته‌باشید که بخواهید لینک مشاهده آنرا برای یکی از دوستانتان ارسال‌کنید، باید صفحه وبلاگ خود را به‌نحوی طراحی‌کنید که درصورت دریافت یک پارامتر (مثلاً blogid)، مستقیماً همان مطلب را نمایش دهد و کاربر مجبور نباشد دربین مطالب وبلاگ (یا در آرشیو)، به‌دنبال مطلب موردنظر بگردد. در چنین مواردی، باید از متد GET استفاده‌کنید؛ زیرا به‌سادگی، امکان مشخص‌کردن blogid را در آدرس صفحه، به‌شما می‌دهد. برای مثال، اگر شماره مطلب موردنظر شما، ۱۵۳ باشد، کافی‌است آدرس زیر را برای دوستتان ارسال‌کنید (به‌جای URL باید آدرس صفحه خود را بنویسید):

```
URL?blogid=153
```

بدین‌ترتیب، دوست شما با کلیک‌کردن برروی لینک فوق، مستقیماً به متن شماره ۱۵۳ هدایت می‌شود. به این‌عمل، نشانه‌گذاری (Bookmark) می‌گویند و فقط با استفاده از متد GET امکان‌پذیر است (علت عدم امکان Bookmark با استفاده از متد POST را در ادامه خواهیم گفت). یکی‌دیگر از کاربردهای متد GET، بالابردن آمار بازدید سایت ازطریق موتورهای جستجو است. برای درک این قابلیت، فرض‌کنید همان وبلاگ را طوری طراحی‌کرده‌اید که اگر پارامتر category را دریافت‌کند (به‌صورت یک رشته)، مطالب مرتبط با همان category را نمایش‌دهد. برای مثال، اگر کاربر آدرس زیر را بازکند:

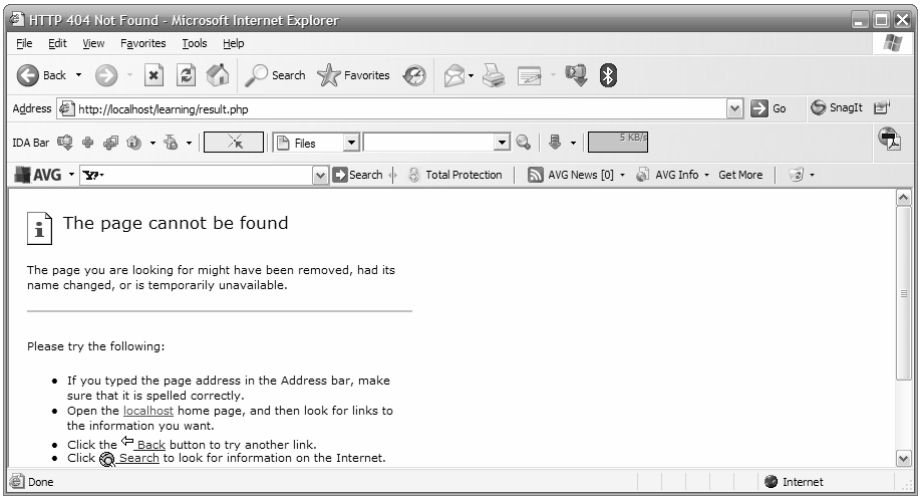
```
URL?category=computer
```

تمامی مطالب مرتبط با کامپیوتر که در وبلاگ شما موجود است، نمایش‌یابد. با استفاده از متد GET، اگرچه URL همه صفحات یکسان است؛ اما ازآنجا که ازطریق تنظیم پارامتر category، آدرسی در وبلاگ شما یافت می‌شود که کلمه computer در آن ذکر شده‌باشد، درصورتی‌که کاربر در موتورهای جستجو به‌دنبال عبارت computer بگردد، سایت شما نیز در نتایج جستجو ظاهر خواهد شد.

حال ببینیم متد POST چه تفاوتی دارد. در اینجا، همان فرم را با روش POST ملاحظه می‌کنید:

```
<FORM ACTION="result.php" METHOD=POST>
<LABEL FOR="un" ACCESSKEY="u"><U>U</U>sername: </LABEL>
<INPUT TYPE="text" NAME="uname" ID="un"/>
<BR/><BR/>
<LABEL FOR="pw" ACCESSKEY="p"><U>P</U>assword: </LABEL>
<INPUT TYPE="password" NAME="pass" ID="pw"/>
<BR/><BR/>
<INPUT TYPE="submit" NAME="submitbutton" VALUE="Send"/>
</FORM>
```

حال مجدداً در فرم ظاهرشده (که ازنظر ظاهری دقیقاً مشابه فرم قبل است)، در کادر Username عبارت Ali و در کادر Password عبارت 1234 را وارد نموده و برروی دکمه Send کلیک می‌کنیم:



این‌بار نیز با همان خطا مواجه می‌شویم؛ اما فعلاً آدرس صفحه برای ما اهمیت دارد:

```
http://localhost/learning/result.php
```

مشاهده می‌کنید که در اینجا، مقادیر واردشده توسط کاربر، در آدرس ذکر نمی‌شوند؛ بلکه هم‌زمان با درخواست صفحه مقصد از سرور، پارامترهای واردشده توسط کاربر نیز برای سرور ارسال می‌گردد. درنتیجه، این روش برای ارسال اطلاعات محرمانه، بهتر است. بنابراین، پیشنهاد می‌کنیم عادت‌کنید همیشه از متد GET استفاده‌کنید، مگر اینکه یکی از دو محدودیت زیر را داشته‌باشید:

- درصورت استفاده از متد GET، آدرس تولیدشده بیشتر از ۲۵۵ کارکتر گردد (مرورگر پشتیبانی نمی‌کند).
- اطلاعات ارسال‌شده، محرمانه باشد.

بنابراین، در مثال فوق، با توجه به محدودیت دوم، از متد POST استفاده می‌کنیم.

پردازش اطلاعات دریافت‌شده در صفحه مقصد

تا اینجا، آموختیم که چگونه اطلاعات را ارسال‌کنیم؛ اما بخش مهم کار مانده‌است: پردازش ورودی کاربر. حال قصد داریم صفحه result.php را طراحی‌کنیم که اطلاعات کاربر را دریافت‌کرده و پردازش مناسب را برروی آنها انجام می‌دهد. برای مثال، قصد داریم صفحه result.php را به‌نحوی طراحی‌کنیم که اگر کاربر عبارت Ali را در کادر Username و عبارت 1234 را در کادر Password وارد کرده‌باشد، با پیغام Welcome!!! و درغیر این‌صورت، با پیغام Invalid User! و یک پیغام به‌صورت زیر مواجه‌گردد:

```
Click here to retry.
```

که اگر برروی کلمه here کلیک‌کند، به صفحه فرم بازگردد تا بتواند مجدداً تلاش‌کند.

دسترسی به ورودی‌های کاربر در PHP

در PHP برای دسترسی به آنچه کاربر وارد کرده‌است، روش‌های مختلفی وجود دارد. یکی‌از این روش‌ها، تعریف این پارامترها به‌صورت متغیرهای سراسری در برنامه است. برای این‌کار، باید تنظیمات PHP را که در فایل php.ini قرار دارد، تغییر داده و خاصیت register_globals به‌صورت زیر تنظیم‌کنیم:

```
register_globals=On
```

درنتیجه، متغیرهایی هم‌نام با خاصیت name کنترل‌های صفحه مبدأ، در صفحه مقصد تعریف خواهد شد که مقادیر واردشده توسط کاربر، درون آنها ذخیره شده‌است:

```
$uname="Ali";
$pass="1234";
```

استفاده از این روش، بنا به دلایل زیر به هیچ‌عنوان توصیه نمی‌شود:

- سطح دسترسی همه متغیرها سراسری می‌شود و ممکن‌است برخی متغیرها رونویسی شوند.
- همه سرورها بنا به دلایل امنیتی، اجازه دسترسی به فایل php.ini را نمی‌دهند.

روش دوم، استفاده از آرایه‌های سراسری ازپیش تعریف‌شده در PHP است. سه‌مورد از این آرایه‌ها که به مبحث این جلسه مربوط است، به‌قرار زیر هستند:

- $_GET : این آرایه، حاوی پارامترهایی است که با متد GET به صفحه مقصد ارسال شده‌اند.
- $_POST : اگر پارامترها با متد POST به صفحه مقصد ارسال‌شوند، در این آرایه قرار می‌گیرند.
- $_REQUEST : این آرایه، پارامترهای ارسال‌شده با هر دو روش GET و POST را در خود ذخیره می‌کند که به‌دلیل پایین‌آمدن امنیت، آنرا مورد استفاده قرار نمی‌دهیم (ممکن‌است برای امنیت بیشتر، از متد POST استفاده کرده‌باشیم؛ ولی با استفاده از این آرایه، صفحه مقصد، پارامترهای نوع GET را نیز دریافت می‌کند و لذا، امکان امتحان‌کردن مقادیر مختلف با واردکردن آنها در آدرس وجود دارد).

همه آرایه‌های فوق، از اندیس رشته‌ای برای مشخص‌کردن مقادیر استفاده می‌کنند که اندیس هر عنصر، همان خاصیت name کنترل مربوطه در صفحه مبدأ است. برای مثال، پارامترهای واردشده در فرم قبل را توسط دستورات زیر می‌توانیم نمایش‌دهیم:

```
echo($_POST["uname"]."<BR/>\n");
echo($_POST["pass"]."<BR/>\n");
```

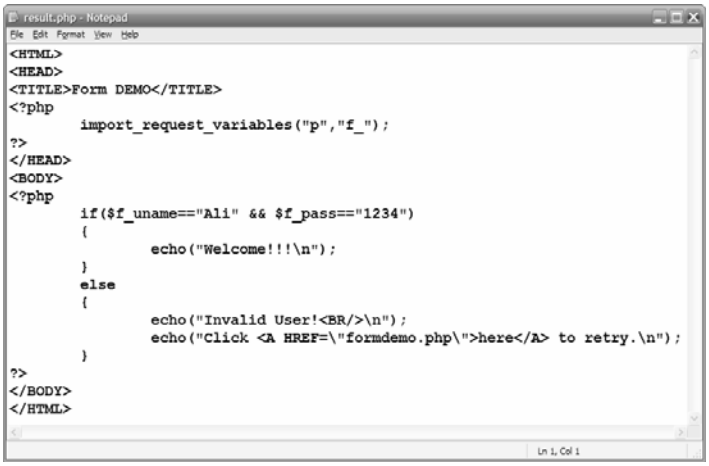
تنها ایراد این روش، طولانی‌شدن نام متغیرها برای دسترسی به پارامترها است ($_POST به ابتدای همه اسامی اضافه می‌شود).

نکته: PHP درمورد نام توابع و متغیرها نسبت‌به بزرگی و کوچکی حروف حساس‌است. آنها را به همان‌شکلی که در این آموزش ملاحظه می‌کنید، وارد نمایید.

نهایتاً روش سوم که درحقیقت، بهترین روش است و از این به‌بعد، صرفاً آنرا مورد استفاده قرار خواهیم‌داد، استفاده از تابع import_request_variables است. وظیفه این تابع، دریافت پارامترهایی که برای صفحه ارسال شده‌اند و سپس، تبدیل آنها به متغیرهایی با پیشوند مشخص (برای پرهیز از تداخل با متغیرهای دیگر) است. این تابع، دو پارامتر از ورودی دریافت می‌کند که هردو، از نوع رشته‌ای هستند. در رشته اول، کارکترهای مختلفی که هرکدام بیانگر نوع خاصی از ارسال پارامترها هستند، ذکر می‌شود که دو مورد از آنها که مورد بحث این جلسه است، g (برای دریافت پارامترهای GET) و p (برای دریافت پارامترهای POST) می‌باشد. دقت‌کنید که می‌توان بیش‌از یک کارکتر را در این رشته مشخص‌نمود. برای‌مثال، اگر بخواهیم صفحه مقصد، پارامترهای GET و POST را بپذیرد، رشته اول را به‌صورت "gp" یا "pg" می‌نویسیم (که به همان‌دلیلی که درمورد $_REQUEST ذکرشد، پیشنهاد نمی‌شود). پارامتر دوم، رشته‌ای است که پیشوند موردنظر را برای رشته‌ها تعیین می‌کند. مثلاً اگر پارامتر دوم را "f_" بگذاریم، برای دسترسی به پارامترهای فرم قبل، از متغیرهای $f_uname و $f_pass استفاده‌می‌کنیم.

با توجه به توضیحات فوق، کد فایل result.php باید به‌صورت زیر باشد (شماره خطوط برای توضیح است):

```
01 <HTML>
02 <HEAD>
03 <TITLE>Form DEMO</TITLE>
04 <?php
05     import_request_variables("p","f_");
06 ?>
07 </HEAD>
08 <BODY>
09 <?php
10     if($f_uname=="Ali" && $f_pass=="1234")
11     {
12         echo("Welcome!!!\n");
13     }
14     else
15     {
16         echo("Invalid User!<BR/>\n");
17         echo("Click <A HREF=\"formdemo.php\">here</A> to retry.\n");
18     }
19 ?>
20 </BODY>
21 </HTML>
```



همان‌طور که در کد فوق ملاحظه می‌کنید، بهتر است تابع import_request_variables در قسمت HEAD صفحه قرار داده‌شود تا وقتی به قسمت BODY می‌رسیم و قصد استفاده از متغیرها را داریم، مطمئن‌باشیم که قبلاً متغیرها تعریف شده‌اند (قسمت HEAD قبل‌از BODY پردازش می‌شود).

در مثال فوق، پارامتر uname (که درحقیقت متن واردشده در کادر متن فرم است)، در متغیر $f_uname و همچنین پارامتر pass (که متن واردشده در کادر متن رمز عبور فرم است)، در متغیر $f_pass ذخیره می‌شود (به‌دلیل استفاده از تابع import_request_variables). سپس، مقدار این متغیرها به‌ترتیب با عبارات "Ali" و "1234" مقایسه می‌شود و درصورت برابربودن، پیغام Welcome!!! و درغیر این‌صورت، پیغام Invalid User! همراه‌با یک لینک برای بازگشت به صفحه فرم، به کاربر نشان‌داده می‌شود. دقت‌کنید که اگر کاربر نام را به‌صورت "ali" واردکند، باز هم با پیغام خطا مواجه می‌شود (زیرا مقایسه رشته‌ها نیز نسبت‌به حروف کوچک و بزرگ حساس‌است). در اکثر فرم‌های استاندارد، نام کاربری نسبت‌به بزرگی و کوچکی حروف حساس نیست. برای این‌که در فرم ما نیز این ویژگی درنظر گرفته‌شود، باید ابتدا متغیر $f_uname را به حروف کوچک یا بزرگ تبدیل‌کرده و سپس، آنرا به‌ترتیب با "ali" یا "ALI" مقایسه‌کنیم. درنتیجه، دستور if را باید به یکی‌از دو شکل زیر بنویسیم:

```
if(strtolower($f_name)=="ali" && $f_pass=="1234")
if(strtoupper($f_name)=="ALI" && $f_pass=="1234")
```

نکته: تابع strtolower یک رشته را به حروف کوچک و تابع strtoupper، آنرا به حروف بزرگ تبدیل می‌کند.

کار با فایل‌ها در PHP

اطلاعات یک وب‌سایت، نقشی اساسی در پویا بودن آن ایفا می‌کنند و روش ذخیره‌سازی آنها، اهمیت به‌سزایی دارد. در جلسات گذشته، با یکی از روش‌های ذخیره‌سازی اطلاعات (یعنی متغیرها) آشنا شدیم. این روش، تا حدود زیادی مناسب است؛ اما در برخی موارد، ممکن‌است نیاز به ذخیره اطلاعات به‌مدت طولانی داشته باشیم. در چنین مواردی، استفاده از متغیرها روش قابل اطمینانی به‌نظر نمی‌رسد؛ زیرا با بستن صفحه یا نمایش مجدد آن (Refresh)، مقادیر قبلی متغیرها ازبین می‌رود. در این جلسه با یکی از روش‌های ذخیره‌سازی طولانی‌مدت اطلاعات یعنی استفاده از فایل‌ها آشنا خواهیم‌شد. البته روش دیگری نیز برای این‌کار وجود دارد که در جلسات آینده با آن نیز آشنا خواهیم شد (استفاده از بانک اطلاعاتی).

مراحل کار با فایل‌ها

برای انجام هرگونه کار با فایل‌ها، باید سه مرحله اصلی زیر را انجام‌دهیم:

- بازکردن فایل و تعیین روش دسترسی به فایل (خواندن، نوشتن و...)
- انجام اعمال موردنظر برروی فایل (خواندن از فایل، نوشتن در فایل)
- بستن فایل

بازکردن فایل

طبیعتاً قبل از انجام هر کاری برروی یک فایل، باید آنرا باز کنیم. برای بازکردن یک فایل، از دستور fopen استفاده می‌کنیم که دارای ساختار زیر است:

```
$variableName = fopen("filePath","mode");
```

در این ساختار، به‌جای filePath مسیر فایل را مشخص می‌کنیم. اگر فایل مذکور در همان مسیر قرارگیری فایل PHP باشد، فقط ذکر نام فایل کافی‌است؛ اما اگر در مسیر دیگری قرار داشته باشد، باید نام فایل را همراه‌با ذکر مسیر نسبی آن، معین کنیم. در قسمت mode، یکی از عبارات زیر را مورد استفاده قرار می‌دهیم:

مقدار	مفهوم
r	خواندن (درصورت عدم وجود فایل، پیغام خطا تولید خواهد شد).
r+	خواندن و نوشتن (درصورت عدم وجود فایل، پیغام خطا تولید خواهد شد).
w	نوشتن (درصورت وجود فایل، حذف‌شده و مجدداً ایجاد خواهد شد).
w+	خواندن و نوشتن (درصورت وجود فایل، حذف‌شده و مجدداً ایجاد خواهد شد).
a	نوشتن در انتهای فایل (درصورت وجود، به انتهای آن اضافه می‌شود و درصورت عدم وجود، ایجاد خواهد شد).
a+	خواندن و نوشتن در انتهای فایل (درصورت وجود، به انتهای آن اضافه می‌شود و درصورت عدم وجود، ایجاد خواهد شد).

درصورتی‌که همه‌چیز به‌درستی پیش‌رود، فایل مذکور با نوع دسترسی مشخص‌شده، باز می‌شود و یک کنترل‌کننده (Handle) برای دسترسی به آن درطول برنامه، بازگردانده خواهد شد که در متغیر معرفی‌شده در قسمت $variableName، قرار خواهد گرفت. بنابراین، در ادامه برنامه، ازطریق همان متغیر، می‌توان اعمال دلخواه را برروی فایل مربوطه، انجام‌داد. برای درک بهتر، به مثال‌های زیر دقت‌کنید:

مسیر فایل موردنظر	:	C:\WAMP\www\test.txt
مسیر فایل PHP	:	C:\WAMP\www\file.php
مثال دستور بازکردن فایل:		$fp=fopen("test.txt","r");
مسیر فایل موردنظر	:	C:\WAMP\www\text\test.txt
مسیر فایل PHP	:	C:\WAMP\www\file.php
مثال دستور بازکردن فایل:		$fp=fopen("text/test.txt","r");
مسیر فایل موردنظر	:	C:\WAMP\www\test.txt
مسیر فایل PHP	:	C:\WAMP\www\code\file.php
مثال دستور بازکردن فایل:		$fp=fopen("../test.txt","r");

نکته: برای دسترسی به پوشه والد، از .. استفاده می‌شود. ضمناً برای جداکردن مسیرها به‌جای \ باید از / استفاده‌کنید. به‌علاوه، در همه مثال‌های فوق، فایل test.txt در حالت خواندن باز شده است.

درصورتی‌که فایل موردنظر وجود نداشته‌باشد یا بنا به هر دلیلی، PHP نتواند آن فایل را باز کند (برای مثال، به‌دلیل تنظیمات امنیتی سرور)، هشداری مشابه پیغام زیر در صفحه خروجی ظاهر خواهد شد:

Warning: fopen(test.txt): failed to open stream: No such file or directory in **C:\WAMP\www\file.php** on line **5**

که در پیغام مذکور، به‌جای نام و مسیر فایل و شماره خط، نام و مسیر فایل تولیدکننده خطا و شماره خطی که در آن، اقدام به بازکردن فایل شده‌است، ظاهر خواهد شد.

توجه‌کنید که پیغام‌های خطا موجب توقف اجرای کد نمی‌شوند و دستورات بعدی کد پردازش خواهند شد. درنتیجه، در دستورات بعد، هر دستوری که اقدام به خواندن از فایل (یا نوشتن در فایل) نماید، موجب بروز خطا خواهد شد. برای پرهیز از چنین وضعیتی، بهتر است فایل‌ها را با چنین ساختاری باز نماییم:

```
$fp=fopen("test.txt","r") or die("Error: Can't open the file.");
```

در این ساختار، وجود قسمت پررنگ‌شده، موجب می‌شود که درصورت عدم موفقیت در بازکردن فایل، دستور die فراخوانی‌شده و ضمن نمایش پیغام خطای مشخص‌شده، به اجرای کد خاتمه‌دهد. بدیهی‌است که دستورات بعد از die اجرا نخواهند شد.

بازکردن فایل‌های سایت‌های دیگر

برای بازکردن فایل‌های یک سایت دیگر، می‌توانید مشابه مثال زیر عمل‌کنید:

```
$file=fopen("http://www.ncis.ir/index.php","r");
```

دقت‌کنید که فایل‌های سایت‌های دیگر را می‌توان فقط در وضعیت خواندن باز نمود.

نوشتن در فایل

بعد از بازکردن یک فایل در هرکدام از وضعیت‌های w، w+، a و a+، می‌توانیم عمل نوشتن را در فایل موردنظر، انجام‌دهیم. برای این‌کار، دستورات مختلفی وجود دارد که یکی‌از آنها، دستور fwrite است:

```
fwrite($file,stringToSave);
```

که به‌جای $file باید متغیر مورداستفاده در دستور fopen و به‌جای stringToSave باید رشته موردنظر را برای ذخیره‌سازی بنویسیم. برای مثال، به دستورات زیر توجه‌کنید:

```
$file=fopen("sample.txt","a") or die("Error: Can't open the file.");
fwrite($file,"Hello!\n");
```

این دستورات موجب بازکردن فایل sample.txt در وضعیت نوشتن در انتهای فایل می‌شوند (اگر فایل موجود باشد، اطلاعات جدید به انتهای آن اضافه می‌شود و درصورت عدم وجود فایل، ایجاد خواهد شد). سپس عبارت Hello! همراه‌با یک کارکتر New Line (\n که موجب رفتن به سطر بعد می‌شود)، در انتهای فایل بازشده، درج خواهد شد.

دستور fwrite دارای یک پارامتر اختیاری عددی نیز می‌باشد که در انتهای همه پارامترها ذکر می‌شود و درصورت تعیین‌شدن، از رشته مشخص‌شده، به‌اندازه عددی که تعیین می‌شود، در فایل نوشته خواهد شد. مثال:

```
fwrite($file,"Hello User!\n",5); //Writes 5 characters (Hello) in the file.
```

نکته: اگر طول رشته مشخص‌شده، کمتر از تعداد کارکتر معین‌شده در پارامتر سوم باشد، رشته مربوطه به‌طور کامل در فایل نوشته خواهد شد.

دستور دیگری نیز به‌نام fputs برای نوشتن در فایل وجود دارد که ازنظر عملکرد و پارامترهای ورودی، دقیقاً مشابه دستور fwrite است؛ لذا از توضیح آن خودداری می‌کنیم و شما می‌توانید به‌دلخواه، از هرکدام استفاده‌کنید.

خواندن از فایل

برای خواندن از فایل، دستورات مختلفی در PHP وجود دارد که هرکدام، روش متفاوتی را برای دریافت اطلاعات از فایل، به‌کار می‌برند. یکی‌از این دستورات، دستور fgets است که ساختار کلی زیر را دارد:

```
$line = fgets($file);
```

که در این ساختار کلی، با هربار اجرای دستور فوق، یک خط از $file خوانده‌شده و درون رشته $line ذخیره می‌شود. برای خواندن تمامی فایل و نمایش آن در خروجی، می‌توانید از ساختاری مشابه مثال زیر استفاده‌کنید:

```
while(!feof($file))
{
      $line=fgets($file);
      echo("$line<BR/>\n");
}
```

همان‌طور که ملاحظه می‌کنید، در این ساختار از یک دستور به‌نام feof برای کنترل شرط حلقه while استفاده شده‌است. این دستور، یک فایل را مورد بررسی قرار می‌دهد و درصورت رسیدن به انتهای فایل، مقدار TRUE باز می‌گرداند. بنابراین در حلقه while، تا زمانی که به انتهای فایل نرسیده‌ایم (به‌دلیل استفاده از !)، یک خط از فایل خوانده‌شده و درون متغیر $line قرار می‌گیرد. سپس توسط دستور echo، خط خوانده‌شده از فایل، در خروجی نوشته می‌شود. البته دستور fgets دارای یک پارامتر اختیاری از نوع عددی نیز می‌باشد که درصورت ذکرشدن، هرباره از یک سطر به‌اندازه یک‌واحد کمتر از عدد مشخص‌شده از فایل می‌خواند و درصورتی‌که قبل‌از خواندن تعداد کارکترهای مشخص‌شده، به انتهای سطر برسد، عمل خواندن را متوقف‌کرده و تا همان‌قسمت را بر می‌گرداند. مثال:

```
$str=fgets($file,5);
```

در مثال فوق، با هربار اجرای دستور fgets، چهار کارکتر از فایل خوانده‌شده و در متغیر $str ذخیره می‌شود. اگر قبل‌از رسیدن به چهار کارکتر، سطر تمام‌شود، کارکترهای خوانده‌شده، در متغیر $str ذخیره خواهد شد.

خواندن یک فایل و ذخیره در آرایه

در اغلب موارد، ممکن‌است نیاز به خواندن یک فایل و ذخیره هر سطر از آن در یک آرایه از نوع رشته باشد. در چنین مواردی، از ساختاری مشابه مثال زیر استفاده می‌شود:

```
$file=fopen("test.txt","r");
while(!feof($file))
{
	$content[]=fgets($file);
}
```

بدین‌ترتیب، محتویات فایل خوانده‌شده، به‌صورت سطر به سطر در یک آرایه از نوع رشته ذخیره می‌شود. البته PHP یک ساختار بهینه‌تر برای انجام عمل فوق دارد:

```
$content=file("test.txt");
```

دستور فوق، عمل بازکردن فایل، خواندن سطر به سطر آن و ذخیره در آرایه $content و بستن فایل را به‌طور خودکار انجام می‌دهد (درمورد بستن فایل در ادامه همین جلسه، توضیح خواهیم داد).

البته استفاده از دستور file درصورتی‌که حجم فایل شما (در این مثال، test.txt) بسیار زیاد باشد، سرعت اجرای اسکریپت شما را به‌شدت کاهش خواهد داد؛ بنابراین در چنین شرایطی، بهتر است از همان حلقه while و دستور fgets استفاده‌کنید.

خواندن فایل و ذخیره در رشته

برخی اوقات ممکن‌است نیاز به ذخیره محتویات یک فایل در یک رشته طولانی (به‌جای یک آرایه رشته‌ای شامل سطرهای مختلف فایل) داشته‌باشید. در این‌گونه موارد، به‌جای تابع file، از تابع file_get_contents استفاده‌کنید. روش کاربرد این تابع دقیقاً مشابه تابع file است؛ با این تفاوت که خروجی آن، به‌جای یک آرایه که هر عنصر آن، یکی‌از سطرهای فایل است، یک رشته طولانی شامل کلیه محتویات فایل خواهد بود:

```
$contents=file_get_contents("test.txt");
```

خواندن یک کارکتر از فایل

برای این‌کار از تابع fgetc استفاده می‌شود که ساختار آن به‌صورت زیر است:

```
$char=fgetc($file);
```

این تابع، با هر بار فراخوانی، یک کارکتر را از فایل خوانده و آنرا به‌صورت یک رشته تک‌کارکتری باز می‌گرداند.

مثال زیر، محتویات فایل 1.txt را به‌کمک تابع fgetc، در فایل 2.txt کپی می‌کند:

```
$in=fopen("1.txt","r");
$out=fopen("2.txt","w");
while(!feof($in))
{
	$char=fgetc($in);
	fwrite($out,$char);
}
```

بستن فایل

از آنجا که هر فایل برای نگه‌داری کنترل‌کننده خود از متغیرها استفاده می‌کند، برای صرفه‌جویی در حافظه و پرهیز از خراب‌شدن احتمالی فایل، باید بعد از اتمام کار با هر فایل، آنرا با دستور fclose ببندیم. مثال:

```
fclose($file);
```

بنابراین، باید به انتهای تمامی مثال‌های فوق، دستور fclose را برای بستن فایل، اضافه‌نماییم.

کار با بانک‌های اطلاعاتی

اگر نیاز به ذخیره اطلاعات پیچیده‌تری داشته‌باشید، بخواهید امنیت آنها را تضمین‌کنید و یا کاربران زیادی بتوانند به‌طور همزمان به داده‌ها دسترسی داشته‌باشند، بانک اطلاعاتی عملکرد بسیار بهتری نسبت‌به فایل‌های متنی ساده خواهد داشت. به‌علاوه، اگر با یک نرم‌افزار مدیریت بانک اطلاعاتی آشنا باشید، کارکردن با بانک‌های اطلاعاتی به‌سادگی استفاده از فایل‌های متنی خواهد بود.

درک نرم‌افزار بانک اطلاعاتی

بانک اطلاعاتی، درحقیقت یک رابط مدیریت فایل است که اطلاعات را به‌صورت ساخت‌یافته ذخیره می‌کند؛ درنتیجه می‌توانید به‌سرعت اطلاعات دلخواه خود را در زمان نیاز، بیابید. بانک اطلاعاتی می‌تواند کوچک‌باشد (مثلاً اسامی، نشانی‌ها و شماره‌های تلفن همه دوستان شما را در خود ذخیره‌کند)؛ یا به‌صورت انبوهی از اطلاعات باشد که با ساختاری بسیار پیچیده، مدیریت می‌شود (مثل بانک اطلاعاتی وب‌سایت Google که مشخصات و محتویات سایت‌های اینترنت را در خود نگه‌داری می‌کند).

هر بانک اطلاعاتی، بدون اهمیت به اندازه و نوع آن، نیاز به یک نرم‌افزار مدیریت دارد تا داده‌ها را به‌شکل صحیح ذخیره‌نماید و در هنگام نیاز به اطلاعات، آنها را با سرعت مناسب از بانک اطلاعاتی استخراج کند.

انواع نرم‌افزارهای مدیریت بانک اطلاعاتی

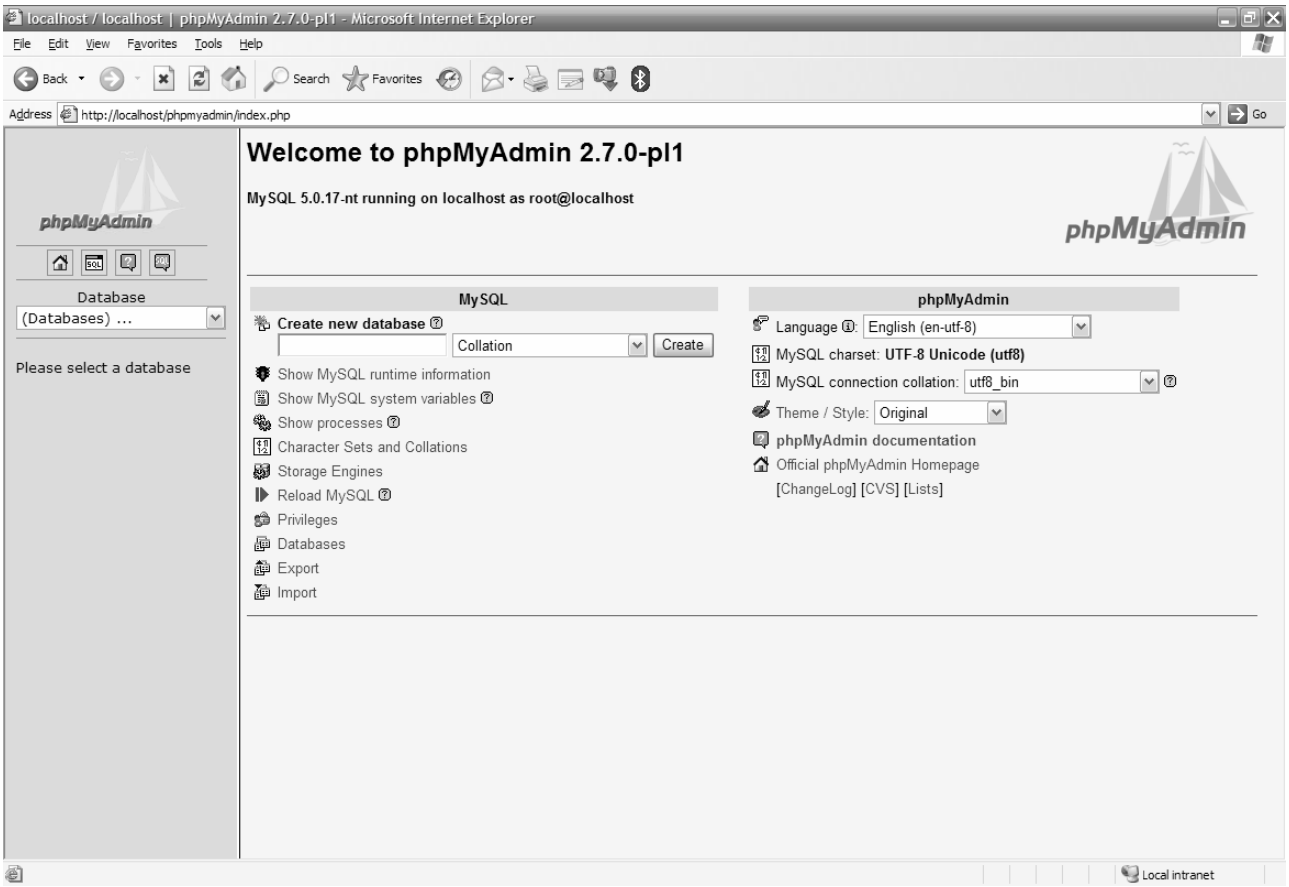
نرم‌افزارهای زیادی برای مدیریت بانک اطلاعاتی طراحی شده‌اند و هرکدام، نقاط ضعف و قوت خاص خود را دارا می‌باشند و به تمامی آنها، لقب DBMS (مخفف DataBase Management System) داده می‌شود. چند نمونه از DBMSهای معروف را مشاهده می‌کنید:

- ✓ IBM DB2
- ✓ Informix
- ✓ Ingres
- ✓ Microsoft Access
- ✓ Microsoft SQL Server (MS SQL)
- ✓ mSQL
- ✓ MySQL
- ✓ Oracle
- ✓ PostgreSQL
- ✓ SQLite
- ✓ Sybase

البته DBMSهای دیگری نیز همچون dBase، ODBC، FoxPro و... نیز وجود دارند که به‌دلیل عدم امکان استفاده در وب، از آنها صرف‌نظر می‌کنیم. در میان DBMSهای فوق، MySQL محبوبیت بیشتری دارد. به‌دلیل امکانات جانبی و سرعت و سهولت استفاده و مهم‌تر از همه، سازگاری کامل با PHP (بدون نیاز به نصب Plug-inهای جانبی)، در این آموزش از MySQL استفاده می‌کنیم.

نصب DBMS

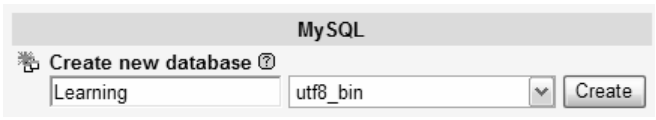
اولین مرحله برای کار با بانک اطلاعاتی، نصب نرم‌افزار مدیریت آن است که به‌دلیل استفاده از WampServer، نرم‌افزار MySQL به‌طور خودکار نصب خواهد شد و لذا، از آموزش نصب آن خودداری می‌کنیم. گام بعدی، استفاده از یک نرم‌افزار است که بتوانیم ازطریق آن، به‌سهولت با بانک اطلاعاتی کار کنیم. علت این مسئله، آن‌است که MySQL فقط ازطریق یک محیط متنی (بدون استفاده از گرافیک) با بانک اطلاعاتی کار می‌کند و این مسئله، طبیعتاً خوشایند نیست. یکی‌از نرم‌افزارهای سودمند مدیریت بانک اطلاعاتی، phpMyAdmin است به‌طور خودکار همراه‌با WampServer نصب می‌شود. برای اجرای آن، باید ابتدا برروی آیکُن WampServer کلیک‌کرده و سپس، گزینه phpMyAdmin را انتخاب کنید تا محیط آن مطابق تصویر زیر ظاهر شود:



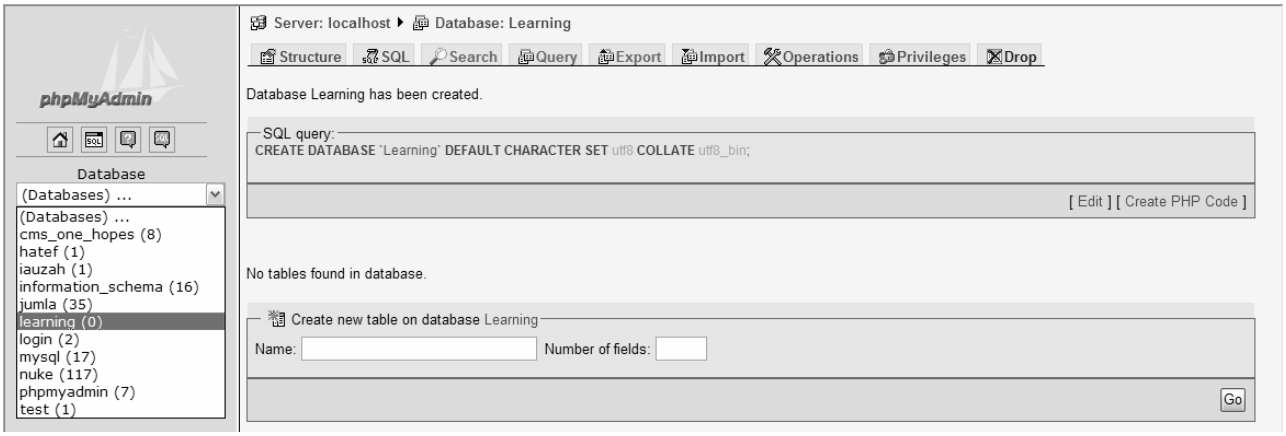
برای یادگیری بهتر روش کار، قصد داریم ساخت بانک اطلاعاتی و جداول داخلی آنرا با phpMyAdmin انجام‌داده و مابقی کارها را (شامل درج داده‌های جدید، خواندن اطلاعات از بانک اطلاعاتی، ویرایش و حذف رکوردها و...)، توسط PHP انجام‌دهیم.

ایجاد بانک اطلاعاتی

برای انجام این کار، در قسمت Create new database نام بانک اطلاعاتی موردنظر را (مثلاً Learning) نوشته و از قسمت Collation، سیستم کدگذاری کارکترها را انتخاب می‌کنیم (پیشنهاد می‌شود از utf8_bin استفاده‌کنید تا بانک اطلاعاتی شما از تمامی زبان‌های یونیکُد پشتیبانی نماید):



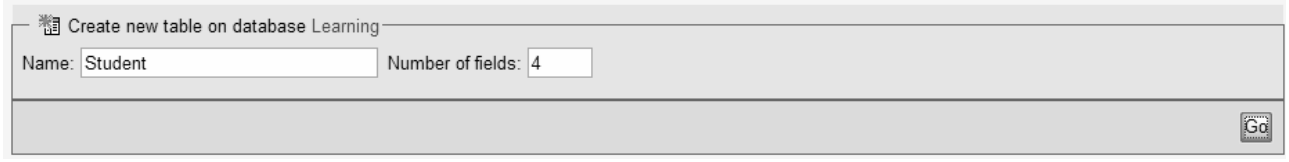
حال برروی دکمه Create کلیک‌کنید تا بانک اطلاعاتی شما ساخته‌شود. بدین‌ترتیب، با ساخته‌شدن بانک اطلاعاتی، نام آن به فهرست بانک‌های اطلاعاتی در سمت چپ صفحه، اضافه می‌شود:



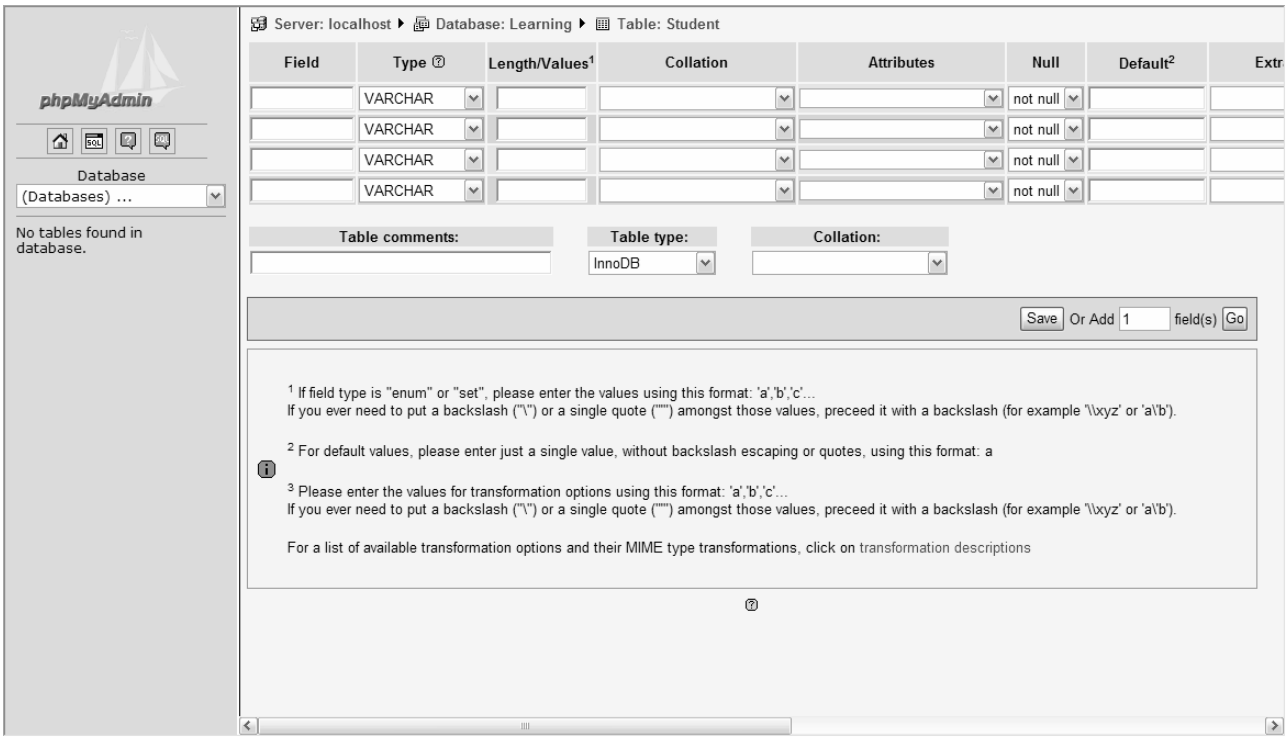
اگر به قسمت SQL query دقت‌کنید، خواهید دید که دستور MySQL موردنیاز برای انجام این‌کار، نوشته شده‌است. درنتیجه با استفاده از ابزار phpMyAdmin، حتی اگر با ساختار دستورات MySQL آشنایی نداشته‌باشید؛ ازآنجاکه پس‌از انجام هر عمل، دستور MySQL مربوط به آن، در این قسمت درج خواهد شد، به‌سرعت با این دستورات آشنایی پیدا خواهید کرد (حتی یک لینک به‌نام Create PHP Code در این قسمت وجود دارد که به‌طور خودکار برای شما اسکریپتی ایجاد خواهد کرد که با درج آن در صفحه موردنظر خود، دستور یا دستورات MySQL موجود در قسمت SQL query توسط اسکریپت شما اجرا خواهد شد).

ایجاد جدول

در قسمت Create new table on database که در ادامه این عبارت، نام بانک اطلاعاتی (مثلاً Learning) ذکر می‌شود، نام دلخواه خود را برای جدول جدید در قسمت Name و تعداد فیلدهای جدول موردنظر را در قسمت Number of fields بنویسید و برروی دکمه Go کلیک‌کنید:



بدین‌ترتیب با صفحه‌ای مطابق تصویر زیر مواجه خواهید شد:



در این صفحه، نام هر فیلد را در ستون Field، نوع آنـرا در قـسمت Type، حـداکثر طـول مجـاز را در قـسمت Length/Values (برای فیلـدهای متنـی) و نـوع کدگـذاری را در قـسمت Collation (پیـشنهاد: utf8_bin) تنظیم کنید. درصورتی که می‌خواهید فیلد خاصی اجازه خـالی‌بـودن در رکوردهـا را داشـته‌باشـد، در سـتون Null مربوطبه همان فیلد، not null را به null تغییر دهید. پیشنهاد می‌شود به بقیه ستون‌ها کـاری نداشـته‌باشـید. در قسمت Table type گزینه InnoDB و در قـسمت Collation مقابـل آن نیـز (مربـوطبـه کدگـذاری جـدول)، utf8_bin را برگزینید و نهایتاً برروی دکمه Save کلیک‌کنید.

در این مثال، قصد داریم چهار فیلد به‌صورت زیر داشته‌باشیم (هیچ‌کدام از فیلدهای زیر نمی‌توانند خالی‌باشند):

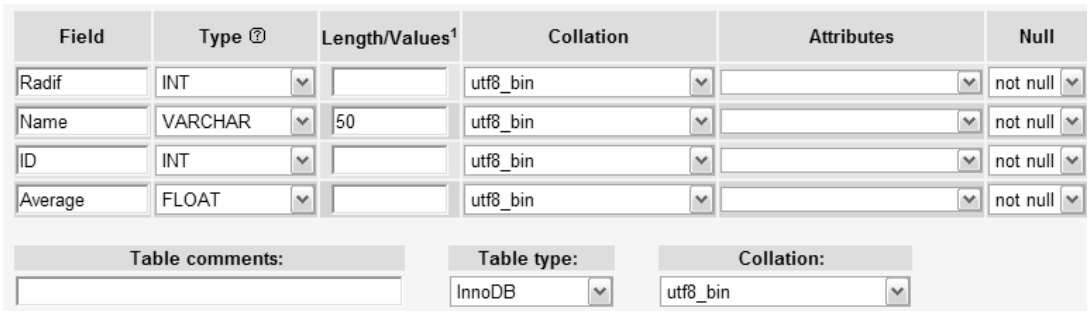
۱– Radif (ردیف) از نوع عددی صحیح

۲– Name (نام) از نوع متنی با حداکثر طول ۵۰ کارکتر

۳– ID (شماره دانشجویی) از نوع عددی صحیح

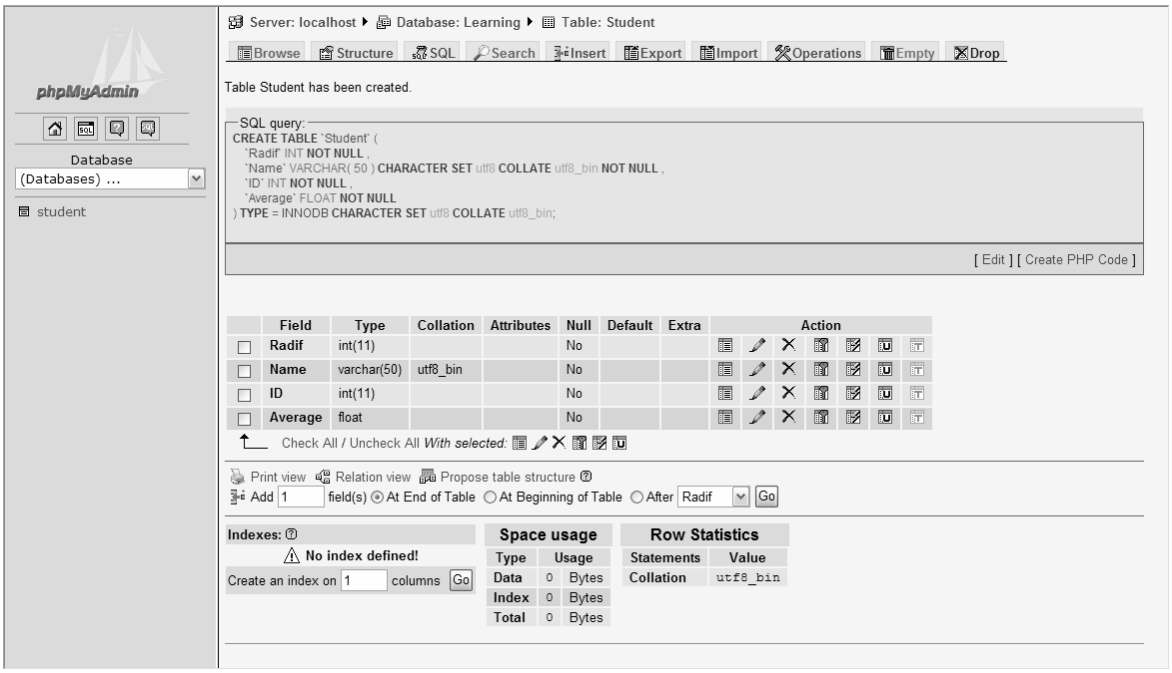
٤– Average (معدل) از نوع عددی اعشاری

بنابراین جدول فوق را به‌صورت زیر تنظیم می‌کنیم:

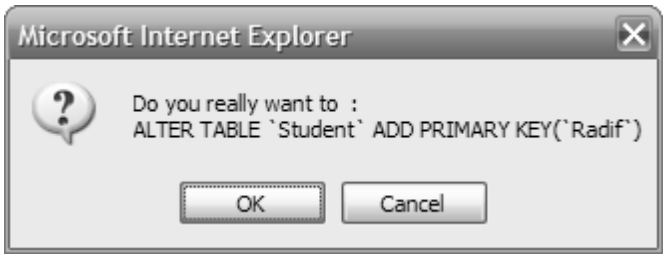


درنهایت، برروی دکمه Save کلیک می‌کنیم تا جدول موردنظر، ایجاد شود.

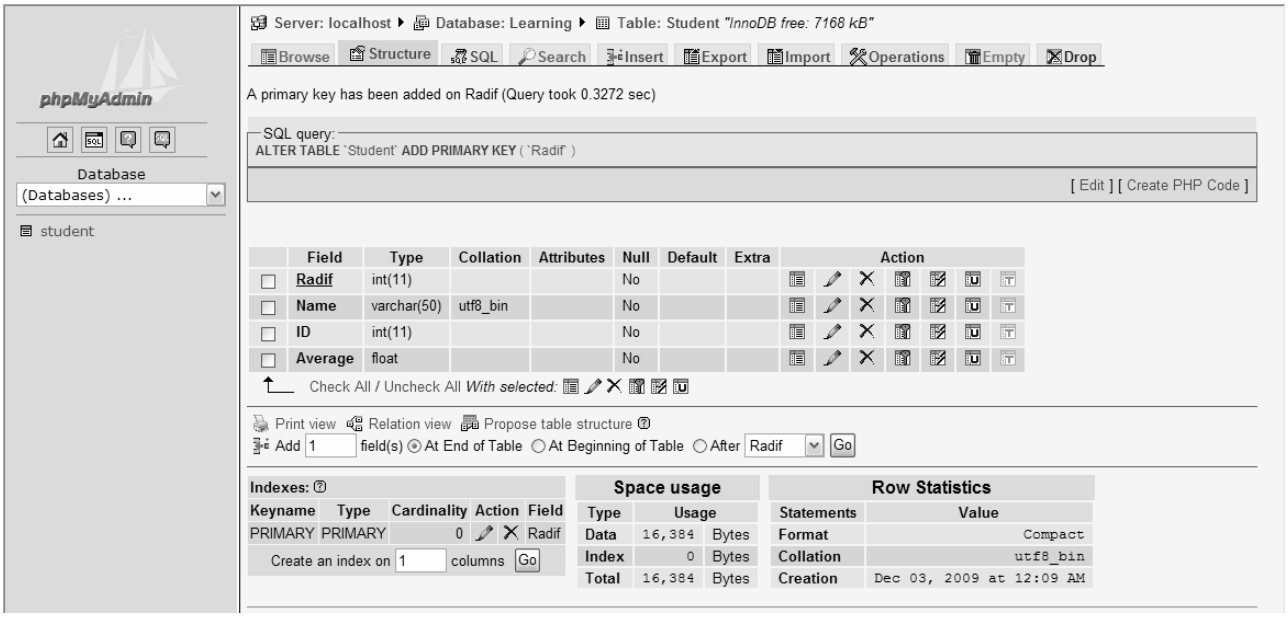
با ساخته‌شدن جدول، phpMyAdmin اطلاعات زیر را نشان خواهد داد:



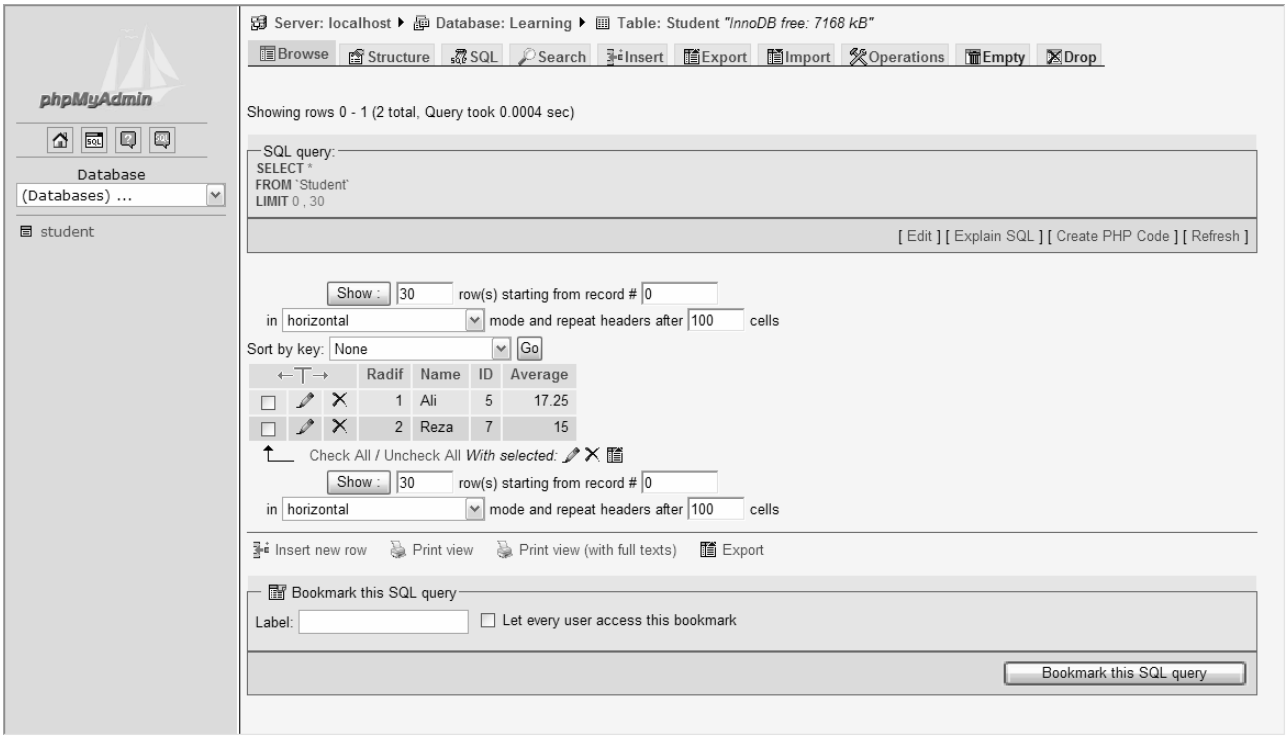
باز هم می‌توانیم دستور ساخت جدول را در قسمت SQL query مشاهده نماییم. برای تبدیل فیلد Radif به کلید اصلی (Primary Key)، باید برروی شکل در قسمت Action مربوطبه این فیلد، کلیک‌کنیم و در پاسخ کادر تأیید ظاهرشده، گزینه Yes را انتخاب‌کنیم:



بدین‌ترتیب، فیلد مذکور به‌صورت کلید اصلی درآمده و phpMyAdmin نیز ضمن اعلام دستور MySQL مربوطه، این موضوع را نشان می‌دهد:



برای آشنایی بیشتر با دستورات MySQL، می‌توانید از گزینه‌های Insert، Empty، Drop و... برای انجام اعمال مختلف و مشاهده دستور MySQL متناظر با آنها استفاده‌کنید و دستور مربوطه را در قسمت SQL query مشاهده‌نمایید. بعد از درج حداقل یک رکورد در بانک اطلاعاتی (توسط گزینه Insert)، گزینه Browse نیز فعال خواهد شد و با کلیک برروی آن، رکوردهای ثبت‌شده را مشاهده خواهید کرد:



بنابراین، توسط گزینه Browse و استفاده از گزینه‌های مختلف آن، با ساختار دستور SELECT نیز آشنا خواهید شد. درصورت کلیک‌کردن برروی دکمه می‌توانید یک رکورد را ویرایش‌کنید و بدین‌ترتیب، با ساختار دستور UPDATE آشنا شوید. نهایتاً با کلیک‌کردن برروی دکمه  هر رکورد، می‌توانید آنرا حذف‌نمایید و با ساختار دستور DELETE نیز آشنا شوید. در آینده می‌توانید از این دستورات برای انجام کارهای موردنظر خود برروی داده‌های موجود در بانک اطلاعاتی، در کُدهای PHP استفاده‌نمایید.

اتصال PHP به MySQL

PHP ازطریق توابعی که انحصاراً برای کار با بانک اطلاعاتی طراحی شده‌اند، به DBMSها (مثل MySQL) متصل‌شده و اعمال موردنظر را انجام می‌دهد. برای هر DBMS، مجموعه دستورات مختلفی وجود دارد که ازطریق آنها، بتوانید با آن DBMS (درصورت پشتیبانی PHP)، ارتباط برقرار کنید. برای مثال، به‌منظور کار با MySQL، باید از دستوراتی که با پیشوند mysql_ شروع می‌شوند (مثل mysql_connect، mysql_query و...) استفاده‌کنید؛ یا برای کار با Sybase از آن نیز از دستوراتی که با sybase_ شروع می‌شوند، استفاده‌نمایید.

فعال‌کردن دستورات یک DBMS

PHP برای اتصال به هرکدام از DBMSها از DLL مخصوص همان DBMS استفاده می‌کند که ازقبل نوشته‌شده و همراه‌با PHP عرضه می‌شود و در پوشه ext از مسیر نصب PHP قرار دارد. بنابراین، تنها کاری که باید انجام‌دهید، فعال‌کردن DLL مربوطه است. برای مثال، به‌منظور فعال‌کردن پشتیبانی از MySQL، باید پس‌از کلیک‌کردن برروی آیکُن WampServer، به منوی PHP Extensions رفته و گزینه‌های php_mysql و php_mysqli را فعال‌کنید. بدین‌ترتیب، تنظیمات مذکور به‌طور خودکار انجام‌شده و سرور Apache مجدداً راه‌اندازی می‌شود تا تنظیمات جدید را اعمال‌نماید. درصورت تمایل به انجام دستی این‌کار باید فایل php.ini را بازکرده و دستورات زیر را درون آن بیابید:

```
;extension=php_mysql.dll
;extension=php_mysqli.dll
```

و سِمی‌کالِن (;) ابتدای آنها را حذف‌نمایید. سپس فایل را ذخیره‌کرده و سرور Apache را مجدداً راه‌اندازی‌کنید (اگر روش این‌کار را نمی‌دانید، کافی‌است یک‌بار سیستم را مجدداً راه‌اندازی کنید).

درمورد سایر DBMSها هم با کمی جستجو، DLL لازم را در فهرست Extensionها یافته و آنرا فعال‌نمایید.

اتصال به بانک اطلاعاتی

اولین مرحله برای کار با بانک اطلاعاتی، اتصال به آن است. این عمل، شبیه بازکردن فایل است؛ با این تفاوت که برای اتصال به بانک اطلاعاتی، ابتدا باید به سرور MySQL متصل شویم. برای این‌کار، باید سه پارامتر را بدانیم:

۱- نام سرور

۲- نام کاربری

۳- رمز عبور

به‌طور پیش‌فرض، در کامپیوترهای شخصی (و نه در شبکه اینترنت)، نام سرور localhost است. همچنین MySQL به‌طور خودکار کاربری با نام root و رمز خالی (بدون رمز عبور) ایجاد می‌کند. بنابراین، درصورتی‌که تنظیمات پیش‌فرض MySQL را تغییر نداده‌باشید، این سه پارامتر به‌ترتیب به‌صورت زیر هستند:

```
1- "localhost"
2- "root"
3- ""
```

برای اتصال به بانک اطلاعاتی MySQL، از دستور mysql_connect استفاده می‌شود. این دستور، سه پارامتر فوق را دریافت‌کرده و یک کنترل‌کننده سرور باز می‌گرداند که مشابه کنترل‌کننده فایل عمل می‌کند و باید درون یک متغیر ذخیره‌شود. بنابراین، ساختار استفاده از این دستور، مشابه زیر خواهد بود:

```
$server=mysql_connect("localhost","root","");
```

بعد از اجرای این دستور، برای مشخص‌کردن سرور، باید از متغیر $server (یا هر متغیر دلخواه دیگر که در کد، به‌جای $server از آن استفاده کرده‌اید)، استفاده‌نماییم. از آنجا که این سه پارامتر در هر فایل PHP که به بانک اطلاعاتی نیاز دارد، به‌کار می‌روند؛ و از آنجا که ممکن‌است برروی هر کامپیوتر، این سه پارامتر تغییر نمایند، برای راحتی بیشتر، پیشنهاد می‌کنیم این پارامترها را درون یک فایل متنی (مثلاً به‌نام server.inc) ذخیره‌کنید:

```
<?php
      $servername="localhost";
      $username="root";
      $password="";
?>
```

سپس توسط دستور زیر، آنها را در هر فایلی که نیازمند اتصال به بانک اطلاعاتی است، ضمیمه‌نمایید:

```
include_once("server.inc");
```

بدین‌ترتیب، می‌توانید دستورات mysql_connect را به‌صورت زیر بنویسید:

```
$server=mysql_connect($servername,$username,$password);
```

مزیت این روش در آن‌است که درصورت تغییر هرکدام از پارامترها، کافی‌است محتویات فایل server.inc را تغییردهید و به‌طور خودکار، تمامی کدهایی که این فایل را ضمیمه‌کرده و ازطریق متغیرهای آن، به بانک اطلاعاتی متصل شده‌اند، اصلاح خواهند شد. اگر قصد دارید بعد از تکمیل طراحی سایت خود، آنرا برروی یک سرور در اینترنت قرار دهید (که در اکثر موارد، همین‌طور است!)، باید از این روش استفاده‌کنید؛ زیرا در اینترنت، نام سرور localhost نیست و باید برای اتصال به بانک اطلاعاتی خود، یک نام کاربری و رمز عبور تعریف‌کنید و دیگر از کاربر root بدون رمز عبور خبری نیست! بنابراین، به‌جای اصلاح تک‌تک کدهای PHP که با بانک اطلاعاتی سروکار دارند (و ممکن‌است به صدها مورد برسند)، کافی‌است محتویات فایل مذکور را اصلاح کنید.

انتخاب بانک اطلاعاتی

مرحله بعد، انتخاب بانک اطلاعاتی ازمیان بانک‌های اطلاعاتی موجود در سروری است که به آن متصل شده‌ایم. برای انجام این‌کار، از دستور mysql_select_db استفاده می‌کنیم. این دستور، نام بانک اطلاعاتی را به‌عنوان پارامتر دریافت می‌کند و خروجی ندارد. بنابراین، ساختاری مشابه دستور زیر خواهد داشت:

```
mysql_select_db("learning",$server);
```

بدین‌ترتیب، بانک اطلاعاتی با نام learning ازبین بانک‌های اطلاعاتی موجود در سرور $server انتخاب می‌شود (می‌توانید آنرا با نام بانک اطلاعاتی خودتان جایگزین‌کنید).

انجام اعمال موردنظر برروی بانک اطلاعاتی

برای انجام هر عمل برروی بانک اطلاعاتی، از زبان MySQL استفاده می‌شود. این زبان، شباهت بسیار زیادی به SQL استاندارد دارد و تنها چند مورد امکانات جانبی به آن افزوده شده‌است تا بتوان از آن در سایت‌های وب، به‌صورت بهینه‌تری استفاده‌نمود. برای اجرای یک دستور SQL برروی بانک اطلاعاتی، از دستور mysql_query استفاده می‌شود. این دستور دارای دو پارامتر است که پارامتر اول، دستور SQL موردنظر و پارامتر دوم، سرور را مشخص می‌کند. خروجی این دستور در یک متغیر ذخیره می‌شود که براساس دستور SQL موردنظر، کاربردهای مختلفی دارد. مثال:

```
$result=mysql_query("select * from student order by radif",$server);
```

درمورد خروجی دستور mysql_query، اگر دستور MySQL اجرا شده، یک دستور select باشد، خروجی آن به‌صورت یک جدول است که در متغیر $result ذخیره می‌شود و بعداً می‌توان رکوردها و فیلدهای آنرا توسط دستورات مخصوص به این‌کار، مورد دسترسی قرار داد. درمورد دستورات update و delete و سایر دستورات ویرایشی (که محتویات یک جدول را تغییر می‌دهند) نیز ازطریق $result می‌توان اعمال مختلفی همچون به‌دست‌آوردن تعداد رکوردهایی که تحت تأثیر دستور مربوطه، تغییر کرده‌اند و... را انجام‌داد.

دستورات PHP پرکاربرد برای کار با MySQL

دربین دستورات PHP، موارد زیر برای کار با بانک اطلاعاتی MySQL، کاربرد بیشتری دارند (برای درک بهتر، هر دستور را با ذکر یک مثال معرفی می‌کنیم):

```
$server=mysql_connect($servername,$username,$password);
```

این دستور برای اتصال به DBMS که نام سرور آن در $servername، نام کاربری در $username و رمز عبور در $password ذخیره شده‌است، به‌کار می‌رود و کنترل‌کننده سرور را درون متغیر $server قرار می‌دهد.

```
mysql_select_db("learning",$server);
```

این دستور برای انتخاب بانک اطلاعاتی learning ازبین بانک‌های اطلاعاتی موجود در سرور مشخص‌شده توسط $server، به‌کار می‌رود. از این مرحله به‌بعد، تمامی دستورات برروی بانک اطلاعاتی که توسط این‌دستور فعال‌شده‌است، انجام خواهد شد.

```
$result=mysql_query("select * from student order by radif",$server);
```

این دستور، یک پرس‌وجوی MyQSL (در این مثال، یک دستور SELECT) را اجراکرده و نتیجه را در $result قرار می‌دهد.

```
$count=mysql_num_rows($result);
```

درصورت استفاده از یک دستور SELECT، تعداد سطرهای جدول ذخیره‌شده در $result، در متغیر $count قرار می‌گیرد.

```
$count=mysql_num_fields($result);
```

درصورت استفاده از یک دستور SELECT، تعداد فیلدهای جدول ذخیره‌شده در $result، در متغیر $count قرار می‌گیرد.

```
$count=mysql_affected_rows();
```

درصورت انجام یک عمل ویرایشی (دستورات UPDATE، DELETE و...)، تعداد رکوردهایی که براثر آن‌عمل تغییر کرده‌اند را در متغیر $count ذخیره می‌کند.

```
$row=mysql_fetch_row($result);
```

درصورت استفاده از دستور SELECT، هربار فراخوانی دستور فوق، یک سطر از جدول موجود در $result را خوانده و درون $row ذخیره می‌کند. $row یک آرایه است و برای دسترسی به فیلدهای آن باید از اندیس عددی استفاده‌شود (برای مثال، $row[0] فیلد اول آنرا مشخص می‌کند). درصورت رسیدن به آخرین سطر و عدم‌توانایی در استخراج سطر جدید، مقدار false را درون $row قرار می‌دهد.

```
$row=mysql_fetch_assoc($result);
```

مشابه دستور قبل است، با این تفاوت که اندیس‌ها عددی نیستند و برای دسترسی به هر فیلد، باید از نام آن فیلد استفاده‌شود (برای مثال، $row["Radif"] فیلد ردیف را مشخص می‌کند). دقت‌کنید که اندیس نسبت‌به بزرگی و کوچکی حروف نام فیلدها حساس است.

```
$row=mysql_fetch_array($result,$type);
```

براساس مقدار $type، مشابه mysql_fetch_row یا mysql_fetch_assoc یا هر دو، عمل می‌کند:

۱- اگر $type برابر با MYSQL_NUM باشد، مشابه mysql_fetch_row عمل خواهد کرد (اندیس عددی).

۲- اگر $type برابر با MYSQL_ASSOC باشد، مشابه mysql_fetch_assoc عمل خواهد کرد (اندیس رشته‌ای).

۳- اگر $type برابر با MYSQL_BOTH باشد، از هر دو نوع اندیس عددی یا رشته‌ای می‌توان استفاده‌نمود.

به‌دلیل انعطاف‌پذیری دستور mysql_fetch_array، پیشنهاد می‌کنیم از این دستور استفاده‌نمایید.

```
$row=mysql_fetch_object($result);
```

یک سطر را به‌صورت یک شئ درون $row قرار می‌دهد و برای دسترسی به مقادیر هر فیلد از آن، باید از ساختاری مشابه $row->radif استفاده‌شود (نام فیلد دلخواه را با radif جایگزین‌کنید).

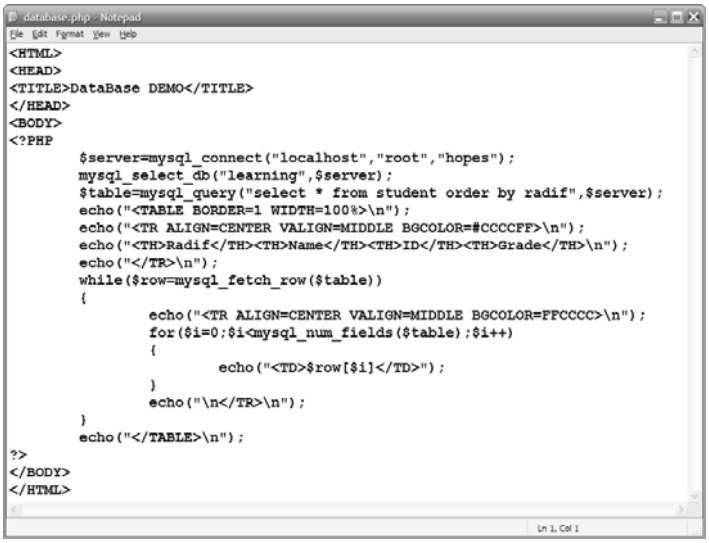
```
$value=mysql_result($result,$row_num,$field_num);
```

این دستور، از جدول $result، فیلد $field_num از سطر $row_num را می‌خواند و درون $value قرار می‌دهد. برای مثال، دستور mysql_result($result,0,0) اولین فیلد از اولین رکورد را مشخص می‌کند.

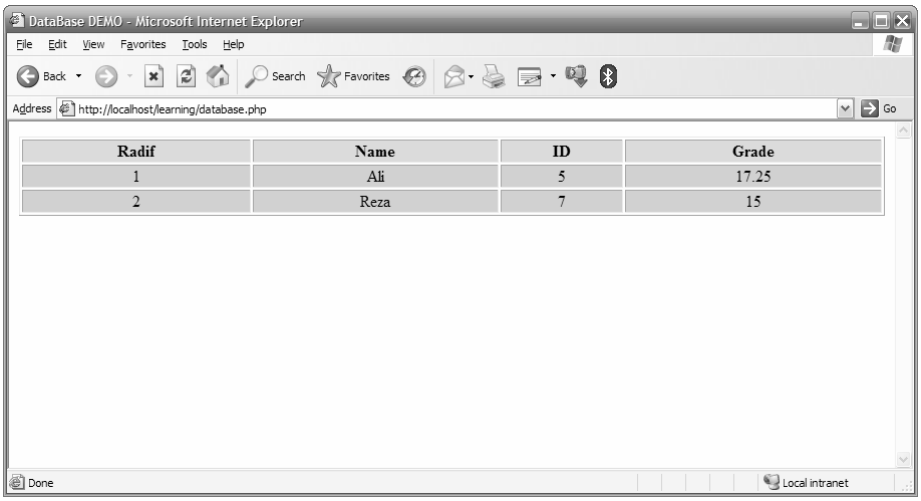
نمایش اطلاعات جدول در صفحه به‌کمک HTML

برای درک بهتر روش کار، به مثال زیر دقت‌کنید که محتویات جدول Student را درون یک جدول HTML نمایش می‌دهد:

```
<HTML>
<HEAD>
<TITLE>DataBase DEMO</TITLE>
</HEAD>
<BODY>
<?PHP
	$server=mysql_connect("localhost","root","hopes");
	mysql_select_db("learning",$server);
	$table=mysql_query("select * from student order by radif",$server);
	echo("<TABLE BORDER=1 WIDTH=100%>\n");
	echo("<TR ALIGN=CENTER VALIGN=MIDDLE BGCOLOR=#CCCCFF>\n");
	echo("<TH>Radif</TH><TH>Name</TH><TH>ID</TH><TH>Grade</TH>\n");
	echo("</TR>\n");
	while($row=mysql_fetch_row($table))
	{
		echo("<TR ALIGN=CENTER VALIGN=MIDDLE BGCOLOR=FFCCCC>\n");
		for($i=0;$i<mysql_num_fields($table);$i++)
		{
			echo("<TD>$row[$i]</TD>");
		}
		echo("\n</TR>\n");
	}
	echo("</TABLE>\n");
?>
</BODY>
</HTML>
```



نتیجه اجرای اسکریپت فوق به‌صورت زیر خواهد بود:



به همین‌ترتیب، می‌توانید از سایر دستورات MySQL نیز استفاده‌کنید.

انتقال اطلاعات بین صفحات وب

در موارد بسیاری ممکن‌است نیاز به انتقال اطلاعات خاصی، از یک صفحه وب به صفحه دیگر داشته‌باشیم. یکی‌از روش‌های این‌کار، استفاده از فرم‌ها است که قبلاً با آن آشنا شدیم؛ اما استفاده از فرم‌ها، در برخی موارد، مشکل است و در موارد خاصی نیز بنا به دلایل امنیتی، نمی‌توان از فرم‌ها استفاده‌نمود. برای مثال، فرض‌کنید کاربران سایت شما باید برای انجام کارهای مختلف در سایت، باید ابتدا نام کاربری و رمز عبور خود را اعلام کنند تا بتوانند وارد سیستم شوند. طبیعتاً نمی‌توانیم در هر صفحه، فرم واردکردن نام کاربری و رمز عبور را قرار دهیم؛ زیرا موجب نارضایتی کاربران و عدم سهولت در برقراری ارتباط با سایت خواهد شد. بنابراین بهتر است در زمان ورود کاربران، یک متغیر خاص را با مقدار مشخصی تعریف‌کنیم و آنرا بین صفحات منتقل نماییم (امنیت این روش، نسبت به روش انتقال نام کاربری و رمز عبور کاربران بین صفحات، بیشتر است).

برای انتقال این متغیر با استفاده از فرم‌ها، دو روش وجود دارد:

۱- استفاده از متد POST و انتقال مقدار مربوطه با استفاده از تگ INPUT و تنظیم خاصیت TYPE از نوع HIDDEN

۲- استفاده از متد GET و انتقال مقدار مربوطه ازطریق URL و ارسال مقادیر درون آدرس صفحه

هر دو روش، مشکلات خاص خود را دارا می‌باشند. روش اول، نیازمند ایجاد یک فرم واقعی و کلیک کاربر برروی دکمه SUBMIT است (که منطقی نمی‌باشد). روش دوم نیز امنیت کمی دارد (زیرا مقدار متغیر مربوطه، در آدرس مشاهده می‌شود و بازدیدکنندگان سایت، به‌راحتی می‌توانند با افزودن مقدار متغیر به آدرس، سایت را فریب‌داده و از امکانات ویژه کاربران، استفاده کنند). البته روش اول نیز امنیت بالایی ندارد؛ زیرا با کلیک راست برروی صفحه و انتخاب گزینه View Source، به‌راحتی تگ INPUT مربوطه که خاصیت TYPE آن از نوع HIDDEN است، درون کد صفحه قابل‌مشاهده بوده و مقدار آن نیز قابل شناسایی است. بنابراین، در چنین مواردی، نیازمند روشی هستیم که بدون استفاده از فرم‌ها و بدون اطلاعات کاربر، مقادیر خاصی را بین صفحات منتقل‌کند. برای این‌کار، دو روش وجود دارد که در این جلسه، با آنها آشنا می‌شویم.

استفاده از کوکی (Cookie)

در PHP می‌توانیم اطلاعات را درون Cookieها ذخیره‌کنیم. کوکی، یک فایل متنی ساده است که با ساختار variable=value ذخیره می‌شود و در آن، variable نام متغیر و value مقدار آن است. تمامی صفحات یک‌سایت تا زمانی که عمر یک کوکی به‌اتمام نرسیده‌است (برحسب ثانیه) یا پنجره مرورگر بسته نشده‌است، می‌توانند مقادیر ذخیره‌شده در آنرا بخوانند. مرورگرها فایل‌های کوکی را برروی کامپیوتر کاربر (کلاینت) ذخیره می‌کنند. بنابراین در فضای سرور و پهنای باند، صرفه‌جویی می‌شود. در نگاه اول، به‌نظر می‌رسد که کوکی‌ها مشکل انتقال اطلاعات را از صفحه‌ای به صفحه دیگر، برطرف می‌کنند (کافی‌است کوکی را ذخیره‌کنید و سپس، در هر صفحه دلخواه، مقدار آنرا خوانده یا تغییر دهید. درحقیقت، کوکی می‌تواند به‌نحوی تنظیم‌شود که در زمان خروج کاربر از سایت شما، باقی بماند و بعد از مدت طولانی (مثلاً یک‌ماه) که کاربر مجدداً به سایت شما مراجعه‌کرد، بار دیگر در سایت شما، مورد استفاده قرار گیرد.

با توجه به مطالب فوق، آیا می‌توان گفت که مشکل کاملاً برطرف شده‌است؟ باید بگوییم که نه کاملاً! کوکی‌ها تحت کنترل شما نیستند. درحقیقت کوکی‌ها در اختیار کاربر هستند. کاربر می‌تواند در هر لحظه دلخواه، کوکی را پاک کند. حتی کاربران می‌توانند مرورگرهای خود را به‌نحوی تنظیم‌کنند که از سایت‌های مختلف، کوکی دریافت نکند. بسیاری از کاربران، کوکی‌ها را رد کرده یا آنها را مرتباً پاک می‌کنند. کاربران زیادی وجود دارند که با ایده ذخیره اطلاعات توسط یک غریبه برروی کامپیوترشان، احساس آرامش و راحتی نمی‌کنند (به‌خصوص فایل‌هایی که بعد از بستن یک سایت، باقی بمانند). همین مسئله باعث کاهش چشمگیر کارآیی کوکی‌ها می‌شود. برای مثال، فرض کنید که سایت شما وابسته به کوکی‌ها باشد و یک کاربر، پشتیبانی از کوکی را در مرورگر خود غیرفعال کند. طبیعتاً سایت شما دیگر برروی کامپیوتر فرد مذکور، قابل استفاده نخواهد بود. برای غلبه بر این مشکلات، روش دیگری ابداع شده‌است که در ادامه همین جلسه، با آن آشنا خواهیم شد.

کوکی‌ها در اصل برای ذخیره مقادیر اندک اطلاعات برای مدت زمان کوتاه طراحی شده‌اند. کوکی‌ها در زمان خروج یک کاربر از سایت شما، ازبین می‌روند؛ مگر این‌که شما مشخصاً کوکی را به‌نحوی تنظیم‌کنید که مدت‌زمان بیشتری باقی بماند. علیرغم کارآیی کوکی‌ها در موارد خاص، نکات زیر را قبل‌از استفاده از آنها درنظر بگیرید:

- کاربران ممکن‌است استفاده از کوکی را در مرورگر خود غیرفعال کنند. بنابراین، سایت خود را به‌نحوی طراحی‌کنید که بدون کوکی‌ها هم بتواند به فعالیت خود ادامه‌دهد.
- PHP دارای ویژگی‌هایی است که بهتر از کوکی‌ها عمل می‌کنند. از نسخه چهارم PHP به بعد، پشتیبانی از Sessionها (بخوانید سِشِن) به PHP افزوده‌شد که امکان ذخیره اطلاعات را تا زمان ماندن یک کاربر در سایت شما (مشابه کوکی‌ها) فراهم می‌کردند. برخلاف کوکی‌ها، Sessionها برای ذخیره اطلاعات خود، از کامپیوتر سرور استفاده می‌کنند. بنابراین، امکان غیرفعال‌کردن آنها وجود ندارد؛ اما در عوض، به‌محض خروج کاربر از سایت شما، بلافاصله Sessionهای آن سایت ازبین می‌روند.
- می‌توانید اطلاعات را در بانک اطلاعاتی ذخیره‌کنید. اگر به بانک اطلاعاتی دسترسی دارید و می‌توانید داده‌ها را در آن ذخیره‌کرده و از آن بخوانید، استفاده از بانک اطلاعاتی در اکثر موارد، گزینه بهتری نسبت‌به کوکی‌ها است؛ زیرا کاربران نمی‌توانند بدون دانستن نام سرور، نام کاربری و رمز عبور، تصادفاً به بانک اطلاعاتی سایت شما متصل‌شوند.

با توجه به اطلاعات فوق، به سراغ روش استفاده از کوکی‌ها می‌رویم.

روش تعریف یک کوکی

برای تعریف یک کوکی، از دستور setcookie استفاده می‌شود. این دستور، دو پارامتر ورودی دارد که اولی، نام متغیری است که باید درون فایل کوکی ذخیره‌شود و دومی، مقدار آنرا مشخص می‌کند. مثال:

```
setcookie("usertype","admin");
```

به‌کمک این دستور، یک متغیر به‌نام usertype درون فایل کوکی ذخیره می‌شود و مقدار آن برابر با admin درنظر گرفته خواهد شد. همان‌طور که ملاحظه می‌کنید، متغیرهای کوکی‌ها از علامت $ استفاده نمی‌کنند. اگر از دستور setcookie مشابه مثال فوق استفاده‌کنید، متغیر مربوطه به‌محض خروج‌از سایت، ازبین خواهدرفت.

این امر در موارد بسیاری، مناسب است؛ اما ممکن‌است در برخی موارد، نیاز به ذخیره یک کوکی برای مدت بیشتر داشته‌باشیم. در چنین وضعیتی، باید از پارامتر سوم این دستور نیز استفاده‌کنیم که اختیاری است و مدت‌زمان باقی‌ماندن کوکی را برحسب ثانیه مشخص می‌کند. این مدت‌زمان، باید با زمان جاری جمع‌شود تا طول عمر واقعی را مشخص‌کند. بنابراین، از تابع time برای محاسبه زمان جاری استفاده‌کرده و آنرا با زمان دلخواه جمع می‌کنیم:

```
setcookie("u_t","admin",time()+3600);      //Cookie will expire after 1 hour
setcookie("u_t","admin",time()+3*86400);   //Cookie will expire after 3 days
```

در دستور اول، کوکی بعد از یک‌ساعت و در دستور دوم، کوکی بعد از سه روز، از بین خواهد رفت.

البته می‌توان از تابع mktime نیز برای تولید زمان دقیق پایان عمر کوکی نیز استفاده‌نمود:

```
setcookie("var","val",mktime(0,0,0,1,1,2010);
//Cookie will expire on 0:0:0 (12AM) 1/1/2010
setcookie("var","val",mktime(13,0,0,,,);
//Cookie will expire on first 13:0:0 (1PM) after execution of this statement
```

در دستور اول، کوکی در ساعت ۱۲ صبح اول ژانویه ۲۰۱۰ و در دستور دوم، در اولین ساعت ۱ عصر (بعد از اجرای دستور فوق)، از بین خواهد رفت. پارامترهای تابع mktime به‌ترتیب، مشخص‌کننده ساعت، دقیقه، ثانیه، ماه، روز و سال هستند.

برای ازبین‌بردن یک کوکی، کافی‌است یک رشته خالی درون آن قراردهید (یا هیچ‌چیزی درون آن قرار ندهید). هر دو دستور زیر، باعث ازبین‌رفتن کوکی usertype خواهند شد:

```
setcookie("usertype");
setcookie("usertype","");
```

خواندن مقدار یک کوکی

برای خواندن مقدار یک کوکی، باید از آرایه $_COOKIE استفاده کنیم. این آرایه، حاوی تمامی کوکی‌های موجود است و اندیس هر کوکی، نام کوکی می‌باشد. برای مثال، به‌منظور نمایش مقدار کوکی usertype، از دستور زیر استفاده می‌کنیم:

```
echo($_COOKIE["usertype"]."<BR/>\n");
```

می‌توانیم از تابع import_request_variables برای تعریف متغیرهای موردنیاز براساس مقادیر این آرایه نیز استفاده‌کنیم (مشابه روش به‌کار رفته درمورد مقادیر ارسال‌شده توسط متدهای GET و POST). تنها تفاوت در این‌است که به‌جای حروف p (برای POST) و g (برای GET)، از c (مخفف COOKIE) استفاده می‌کنیم:

```
import_request_variables("c","c_");
```

این دستور، موجب می‌شود تا تمامی متغیرهای کوکی با پیشوند c_ تعریف‌شوند. بنابراین، برای نمایش مقدار کوکی usertype می‌توانیم از دستور زیر نیز استفاده‌کنیم (بعد از اجرای دستور فوق):

```
echo($c_usertype."<BR/>\n");
```

یک نکته مهم درمورد دستور setcookie

دستور setcookie برای تعریف یک کوکی، نیازمند تغییر هِدِر (Header) صفحه HTML است. این‌مسئله کاملاً طبیعی است؛ زیرا صفحات بعد باید بتوانند اطلاعات را دریافت کنند و اطلاعات کوکی نیز همانند داده‌های GET و POST، باید قبل از تولید یک صفحه، ارسال‌شوند. نکته مهم در آن است که اگر داده‌ای برای مرورگر ارسال‌شود، موجب بروز خطایی با ساختار زیر خواهد شد:

```
Warning: Cannot modify header information - headers already sent by
(output started at URL:L#1) in URL on line L#2
```

که به‌جای URL، مسیر صفحه وب شما و به‌جای L#1، شماره اولین خطی که داده‌ها را برای مرورگر ارسال کرده‌است و به‌جای L#2، شماره خطی که دستور setcookie در آن قرارداشته و تلاش کرده است تا هِدِر را تغییردهد، ذکر می‌شود. بروز این خطا طبیعی است؛ زیرا هِدِر، ساختار و مشخصات یک صفحه وب را مشخص می‌کند و باید قبل‌از ارسال صفحه به مرورگر، تنظیم‌شود. تغییر هِدِر بعد از ارسال داده‌ها به مرورگر، دقیقاً مشابه تلاش برای تغییر نام یک فایل است که همزمان، در یک برنامه ویرایشگر باز شده‌است! برای پرهیز از بروز چنین خطاهایی، دو راه وجود دارد:

۱- استفاده از دستور setcookie به‌عنوان اولین دستور در فایل PHP

۲- پرهیز از ارسال اطلاعات تا زمان کامل‌شدن صفحه و تعیین‌شدن همه cookieها

واضح‌است که روش اول، چندان کارآمد نیست؛ چون ممکن‌است بخواهیم دستور setcookie را براساس یک شرط اجرا کنیم. حتی ممکن‌است داده‌ای که برای مرورگر ارسال شده‌است، قابل مشاهده نباشد! برای مثال، دستورات زیر را ملاحظه‌کنید:

```
01 <HTML>
02 <HEAD>
03 <TITLE>My Web Page</TITLE>
04 </HEAD>
05
06 <BODY>
07 <?PHP
08    setcookie("greeting","hello");
09 ?>
10 </BODY>
11 </HTML>
```

به خط 05 دقت‌کنید: این خط، کاملاً خالی است. مشکل در اینجا است که HTML، کارکترهای Enter اضافی را به‌عنوان Space تعبیر می‌کند و آنها را برای مرورگر ارسال می‌کند. در انتهای یک خط (مثلاً خط 03)، یک کارکتر Space اضافی، به‌طور تصادفی تایپ شده‌باشد. چنین خطاهایی به‌ندرت و به‌سختی قابل تشخیص هستند. بنابراین، ترجیح می‌دهیم که از روش دوم استفاده‌کنیم.

در این روش، از PHP می‌خواهیم که اطلاعات را تا زمان کامل‌شدن صفحه و تعیین تمامی مقادیر و تنظیم هِدِر، نگه‌داری کند و در زمان موردنظر، همه‌را با هم برای مرورگر ارسال نماید. برای این‌کار از دستور ob_start در ابتدای اسکریپت، استفاده می‌کنیم. این دستور، بافِر خروجی (Output Buffer) را فعال می‌کند. با فعال‌شدن بافر خروجی، هیچ داده‌ای برای مرورگر ارسال نمی‌شود و همه داده‌ها در بافر خروجی قرار می‌گیرد. سپس، هر زمان که صفحه کامل شد، با دستور ob_end_flush، محتویات بافر خروجی را برای صفحه وب ارسال می‌کنیم. در نتیجه این تغییرات، اسکریپت فوق به‌صورت زیر تغییر می‌کند:

```
01 <?PHP
02    ob_start();
03 ?>
04 <HTML>
05 <HEAD>
06 <TITLE>My Web Page</TITLE>
07 </HEAD>
08
09 <BODY>
10 <?PHP
11    setcookie("greeting","hello");
12 ?>
13 </BODY>
14 </HTML>
15 <?PHP
16    ob_end_flush();
17 ?>
```

استفاده از سِشِن (Session)

کلمه سِشِن به‌معنای جلسه است و به مدت‌زمانی گفته می‌شود که یک کاربر، در سایت شما سپری می‌کند. کاربران ممکن‌است صفحات زیادی از سایت شما را مشاهده‌کنند و سپس از آن خارج‌شوند. با استفاده از سِشِن‌ها می‌توانید اطلاعات موردنظر خود را از لحظه ورود کاربر به سایت شما (یا هر لحظه دلخواه دیگر که کاربر درون سایت شما قرار دارد و از آن بیرون نرفته‌است)، به‌نحوی ذخیره‌کنید که تا قبل از خروج کاربر از سایت شما، در اختیار تمامی صفحات سایتتان باشد. بنابراین، سِشِن از جهات زیادی، شبیه کوکی است؛ اما یک تفاوت اساسی بین آنها وجود دارد: سِشِن برخلاف کوکی، برروی کامپیوتر سرور ذخیره می‌شود. بنابراین کاربران نمی‌توانند آنرا غیرفعال یا حذف نمایند. به‌علاوه، امکان خواندن آن به‌طور تصادفی نیز وجود ندارد. درعوض، برخلاف کوکی، نمی‌توانید مدت‌زمان مشخصی را برای باقی‌ماندن آن مشخص‌کنید و به‌طور خودکار، با خارج‌شدن کاربر از سایت شما، سِشِن‌ها ازبین می‌روند. از نسخه ٤ به بعد، PHP یک روش مناسب برای انجام این کار، ارائه نموده‌است.

درک روش کار سِشِن‌های PHP

PHP به شما اجازه می‌دهد که یک سِشِن تعریف‌کرده و متغیرهای دلخواه خود را در آن ذخیره‌کنید. بعد از ساخت یک سِشِن، متغیرهای آن جلسه برای استفاده هر صفحه وب از سایت شما، قابل استفاده خواهند بود. برای ایجاد قابلیت دسترسی به اطلاعات یک سِشِن، PHP مراحل زیر را انجام می‌دهد:

۱- PHP یک شماره شناسه جلسه غیرتکراری ایجادکرده و به سِشِن نسبت می‌دهد. این شماره، بسیار طولانی و نامفهوم است و برای هر کاربر، منحصربه‌فرد می‌باشد. بنابراین نه‌تنها قابل حدس‌زدن نیست، بلکه موجب می‌شود هر کاربر، جلسه مربوط به خود را داشته‌باشد (مشابه زمانی که توسط کوکی، فایل‌های هر کاربر برروی کامپیوتر خودش ذخیره می‌شود). این شماره شناسه در یک متغیر سیستمی PHP به‌نام PHPSESSID ذخیره می‌شود.

۲- PHP متغیرهایی را که قصد دارید در جلسه خود ذخیره‌کنید، در فایلی برروی سرور ذخیره می‌کند. نام فایل، همان شماره شناسه است و در مسیری که در فایل تنظیمات PHP (php.ini) و در متغیر session.save_path مشخص شده‌است، ذخیره می‌شود. بدیهی‌است که مسیر مذکور باید از قبل برروی سرور ایجاد شده باشد.

۳- PHP شماره شناسه جلسه را برای هر صفحه سایت، ارسال می‌کند. اگر کاربر قابلیت استفاده از کوکی‌ها را فعال‌کرده‌باشد، PHP شماره شناسه جلسه را ازطریق کوکی‌ها دراختیار همه صفحات می‌گذارد؛ اما اگر کوکی‌ها غیرفعال باشند، رفتار PHP بستگی به وضعیت متغیر trans-sid در فایل تنظیمات PHP (php.ini) دارد که در ادامه همین جلسه، با روش کار آن آشنا خواهیم شد.

٤- PHP متغیرها را از فایل سِشِن خوانده و در اختیار هر صفحه در همان جلسه، قرار می‌دهد. هرگاه کاربر یک صفحه جدید بازکند که متعلق به همان جلسه (سایت مرتبط با سِشِن) است، PHP مقادیر را از فایل مربوطبه همان کاربر (توسط شماره شناسه سِشِن) می‌خواند. این شناسه، توسط هر صفحه، برای صفحه بعد ارسال می‌شود. بنابراین، صفحات مختلف سایت، ردپای کاربر را گم نخواهند کرد. متغیرهای سِشِن در آرایه $_SESSION ذخیره می‌شوند.

بازکردن و بستن جلسات

شما باید یک جلسه را در ابتدای هر صفحه، باز کنید. بازکردن جلسه توسط دستور session_start با ساختار زیر انجام می‌شود:

```
session_start();
```

این دستور، ابتدا وجود یک شماره شناسه سِشِن را بررسی می‌کند و اگر آنرا بیابد، متغیرهای همان سِشِن را از فایل مربوطه، می‌خواند؛ اما اگر شماره شناسه سِشِن را نیابد، یک جلسه جدید ایجادکرده و شماره شناسه مربوطبه آنرا تولید می‌کند. از آنجا که جلسات از کوکی‌ها استفاده می‌کنند، اگر کاربر کوکی‌ها را در تنظیمات مرورگر خود فعال کرده‌باشد، session_start از محدودیت‌های کوکی‌ها تبعیت خواهد کرد. بنابراین بهتر است در زمان استفاده از سِشِن‌ها نیز بافر خروجی را فعال‌کنیم و در پایان صفحه، آنرا به خروجی بفرستیم. از آنجا که برای استفاده از سِشِن‌ها، باید در ابتدای هر صفحه، از دستور session_start استفاده‌کنیم، درصورت تمایل می‌توانیم PHP را به‌نحوی تنظیم‌کنیم که به‌طور خودکار این‌کار را برای هر صفحه انجام‌دهد (هرچند این روش پیشنهاد نمی‌شود). کافی‌است در تنظیمات PHP (php.ini)، متغیر session.auto_start را با عدد یک تنظیم‌کنیم و یک‌بار سرویس‌دهنده PHP را مجدداً راه‌اندازی نماییم (اگر خودتان اجازه مدیریت PHP را دارید، می‌توانید این فایل را تغییردهید؛ درغیر این‌صورت باید از مدیر PHP یا بخش پشتیبانی وب‌سایت خود، بخواهید تا آنرا برای شما ویرایش‌کند). در این‌صورت، نیاز به افزودن دستور session_start به ابتدای هر صفحه، نخواهید داشت.

همان‌طور که قبلاً گفتیم، ممکن‌است سایت خود را به‌نحوی تنظیم کرده‌باشید که فقط به کاربران عضو، خدمات‌رسانی کند. طبیعتاً در چنین مواردی، علاوه‌بر ورود کاربران، باید قابلیت خروج آنها را نیز درنظر بگیرید. دقت‌کنید که به‌محض خارج‌شدن یک کاربر از سایت (حتی اگر به سایت دیگری مراجعه نکند و صرفاً از صفحه مخصوص کاربری خود خارج شده‌باشد)، باید سِشِن مربوطبه آن کاربر را ازبین ببریم. برای بستن یک جلسه (و از بین بردن سِشِن)، از دستور زیر استفاده می‌کنیم:

```
session_destroy();
```

این دستور، همه اطلاعات موجود در فایل سِشِن را از بین می‌برد و PHP دیگر شماره شناسه سِشِن را برای صفحات بعد ارسال نمی‌کند. دقت‌کنید که متغیرهای صفحه جاری، ازبین نمی‌روند. برای مثال، اگر توسط دستور زیر:

```
$value=$_SESSION["usertype"];
```

مقدار متغیر سِشِن به‌نام usertype را درون متغیر $value ذخیره کرده‌باشیم و سپس با استفاده از دستور session_destroy، جلسه را ازبین ببریم، هنوز هم متغیر $value معتبر است و برای ازبین‌بردن آن، باید از دستور unset مطابق زیر استفاده‌کنیم:

```
unset($value);
```

بدین‌ترتیب، متغیر $value نیز از بین خواهد رفت. می‌توانید توسط یک دستور unset، چندین متغیر را نیز ازبین ببرید. برای این‌کار، کافی‌است آنها را با ویرگول (,) از هم جدا کنید. مثال:

```
unset($var1,$var2);
```

استفاده از متغیرهای سِشِن در PHP

برای ذخیره یک متغیر در یک جلسه، به‌نحوی که برای صفحات بعد، قابل دسترسی باشد، باید آنرا به‌صورت زیر در آرایه $_SESSION ذخیره‌کنید:

```
$_SESSION["variable_name"]=value;
```

که به‌جای variable_name، نام متغیر و به‌جای value، مقدار آن قرار می‌گیرد (بدیهی‌است که در قسمت نام، از گیومه تک نیز می‌توانیم استفاده‌کنیم). مثال:

```
$_SESSION["usertype"]="admin";
$_SESSION['count']=5;
```

همچنین دستوری به‌نام session_register نیز وجود دارد که با ساختاری مشابه مثال‌های زیر، عمل تعریف یک متغیر سِشِن را انجام می‌دهد:

```
session_register("usertype","admin");
session_register('count',5);
```

درنتیجه، این متغیرها ازطریق مقادیر $_SESSION["usertype"] و $_SESSION['count'] در صفحات دیگر همین‌سایت، قابل دسترسی خواهند بود. البته در اینجا نیز می‌توان از تابع import_request_variables با ساختاری مشابه مثال زیر استفاده‌نمود:

```
import_request_variables("s","s_");
```

که s در پارامتر اول، مخفف session است و بعد از اجرای این دستور، ازطریق متغیرهای $s_usertype و $s_count می‌توان مقادیر sessionهای مربوطه را خواند. دقت‌کنید که درصورت تغییر این متغیرها، متغیر سِشِن متناظر با آنها تغییر نخواهد کرد و برای تغییر آن، باید از همان آرایه $_SESSION استفاده‌کنید.

برای ازبین‌بردن یک متغیر سِشِن، می‌توانید از دستور unset مطابق مثال زیر استفاده‌کنید:

```
unset($_SESSION['count']);
```

البته برای ازبین‌بردن متغیرهای سِشِن، دستوری به‌نام session_unregister وجود دارد که برای استفاده از آن، باید مطابق مثال زیر عمل‌کنید:

```
session_unregister('count');
```

همچنین دستور session_unset با ساختار زیر، موجب حذف تمامی متغیرهای سِشِن می‌شود:

```
session_unset();
```

دقت‌کنید که این دستور، با دستور session_destroy متفاوت است و فقط متغیرهای session را ازبین می‌برد و جلسه را به‌اتمام نمی‌رساند.

نکاتی درمورد روش‌های انتقال شماره شناسه جلسه

بسیاری از کاربران اینترنت، کوکی‌ها را در مرورگرهای خود غیرفعال می‌کنند. PHP مرورگر کاربر را بررسی می‌کند و براساس فعال یا غیرفعال‌بودن کوکی‌ها، به‌صورت زیر عمل می‌کند:

الف) اگر کوکی‌ها فعال باشند:

۱- متغیر $PHPSESSID را تعریف‌کرده و آنرا با مقدار شماره شناسه سِشِن، مقداردهی می‌کند.

۲- از کوکی‌ها برای انتقال $PHPSESSID از یک صفحه به صفحه بعد استفاده می‌کند.

ب) اگر کوکی‌ها فعال نباشند:

۱- یک ثابت به‌نام SID تعریف می‌کند. مقدار این ثابت، به‌صورت زیر است:

```
PHPSESSID=################################
```

که در سمت راست تساوی، یک رشته ۳۲ کارکتری شامل حروف و اعداد وجود دارد و شناسه سِشِن را مشخص می‌کند.

۲- درصورت فعال‌بودن trans-sid، مقدار ثابت فوق را از یک صفحه به صفحه دیگر منتقل می‌کند. اگر کاربر توسط یک لینک، یک تابع header یا یک فرم GET به صفحه بعد منتقل‌شود، SID در آدرس ظاهر می‌شود؛ اما اگر توسط یک فرم POST به صفحه بعد منتقل‌شود، SID ازطریق یک تگ INPUT نوع HIDDEN ارسال خواهد شد. درصورت غیرفعال‌بودن trans-sid، مقدار ثابت فوق منتقل نمی‌شود و شما باید خودتان به‌صورت دستی (درصورت تمایل)، آنرا به‌روشی که در ادامه خواهیم‌گفت، منتقل‌کنید.

trans-sid به‌طور پیش‌فرض غیرفعال است؛ اما درصورت تمایل، می‌توانید آنرا فعال‌کنید. برای این‌کار باید در فایل php.ini، عبارت session.use_trans_id را یافته و آنرا با عدد یک، مقداردهی کنید. پس‌از تنظیم این ویژگی و ذخیره‌کردن فایل php.ini، باید سرور PHP را مجدداً راه‌اندازی کنید.

فعال‌بودن trans-sid مزایا و معایب خاص خود را دارد. درصورت فعال‌بودن trans-sid، می‌توانید از سِشِن‌ها درست همانند زمانی که کوکی‌ها فعال‌هستند، استفاده‌کنید و PHP به‌طور خودکار عمل انتقال SID را از یک صفحه به صفحه دیگر، مدیریت می‌کند و نیازی به تغییر کد اسکریپت خود نخواهید داشت. در عوض، ایراد این روش در آن‌است در اغلب موارد، شماره شناسه سِشِن درون آدرس صفحه ذکر می‌شود که بنا به دلایل امنیتی، ممکن‌است چندان مناسب به‌نظر نرسد. همچنین وقتی که SID درون آدرس ذکر می‌شود، قابل علامت‌گذاری‌کردن (Bookmark) توسط اعمالی همچون افزودن به فهرست علاقمندی‌ها (Favorites) و... خواهد بود. بنابراین، اگر کاربر توسط آدرس علامت‌گذاری‌شده، به سایت شما بازگردد (درحالی‌که SID قبل درون آن موجود است)، SID جدید و SID قبلی با یکدیگر تداخل خواهند داشت و این مسئله، در اکثر موارد منجر به بروز اشکالات جدی خواهد شد.

ارسال شماره شناسه سِشِن به‌صورت دستی

همان‌طور که قبلاً ذکر شد، درصورت غیرفعال‌بودن trans-sid و همچنین غیرفعال‌بودن کوکی‌ها در مرورگر کاربر، PHP شماره شناسه سِشِن را به صفحه بعد ارسال نمی‌کند. درعوض، شما باید خودتان این‌کار را انجام‌دهید. خوشبختانه در چنین مواردی، PHP به‌طور خودکار یک ثابت به‌نام SID تعریف می‌کند که دارای ساختار زیر است:

```
PHPSESSID=################################
```

و در سمت راست تساوی، شماره شناسه سِشِن ذکر می‌شود. مثال:

```
PHPSESSID=877c22163d8df9deb342c7333cfe38a7
```

اگر بخواهید کاربر را توسط یک لینک، یک تابع header (مثل header("location: URL");) یا ازطریق یک فرم GET، به صفحه بعد هدایت‌کنید، باید آدرس را به‌صورت زیر اعلام‌کنید:

```
nextpage.php?<?php echo(SID); ?>
```

که در آن، nextpage.php نام فایل صفحه بعد است. برای مثال، به لینک زیر دقت‌کنید:

```
echo("<a href=\"nextpage.php?".SID."\">Next Page</a>");
```

و این آدرس، به‌طور خودکار به‌صورت زیر تبدیل خواهد شد:

```
<a href="nextpage.php?PHPSESSID=877c22163d8df9deb342c7333cfe38a7">Next Page</a>
```

اما اگر از متد POST در یک فرم استفاده‌کنید، برای ارسال شماره شناسه سِشِن باید مشابه مثال زیر عمل‌کنید:

```
<?php
	$PHPSESSID=session_id();
	echo("<form action=\"nextpage.php\" method=post>\n");
	echo("<input type=\"hidden\" name=\"PHPSESSID\" value=\"$PHPSESSID\"/>\n");
	echo("<input type=\"submit\" value=\"Next Page\"/>\n");
	echo("</form>\n");
?>
```

تنها نکته مهم در کد فوق، تابع session_id است که به‌سادگی، شماره شناسه سِشِن جاری را مشخص می‌کند. بدین‌ترتیب در صفحه جدید، PHP به‌طور خودکار PHPSESSID را خواهد یافت و در صفحه بعد، نیاز به هیچ‌گونه برنامه‌نویسی اضافه نخواهید داشت.

ایجاد جلسات فقط برای کاربران

جلسات PHP برای وب‌سایت‌هایی که محدود هستند و نیاز به ورود کاربران توسط نام کاربری و رمز عبور دارند تا خدمات خود را فقط به کاربرانشان ارائه‌دهند، مناسب می‌باشند. این‌گونه سایت‌ها بدون‌شک صفحات زیادی دارند و قطعاً تمایل ندارید که کاربرانتان در هر صفحه، نام کاربری و رمز عبور خود را وارد کنند! جلسات PHP در چنین وضعیتی، به‌کمک شما می‌آیند(کافی‌است مراحل زیر را انجام‌دهید):

۱- یک صفحه ورود به سایت نشان‌دهید.

۲- اگر کاربر با موفقیت وارد شد، یک متغیر سِشِن ایجاد و ذخیره‌کنید.

۳- هرگاه کاربر به یک صفحه جدید می‌رود، متغیر سِشِن را بررسی کنید تا مطمئن‌شوید که کاربر وارد سایت شده‌است.

٤- اگر کاربر وارد سایت شده‌باشد (متغیر سِشِن دارای مقدار صحیح باشد)، صفحه را نشان‌دهید.

۵- در غیر این‌صورت، کاربر را به صفحه ورود به سایت هدایت‌کنید.

برای مثال، فرض‌کنید به‌محض ورود موفقیت‌آمیز کاربر، یک متغیر سِشِن به‌نام login را با مقدار go ایجاد و ذخیره کرده‌اید. حال کافی‌است کد زیر را در ابتدای هر صفحه قراردهید:

```
<?php
	ob_start();
	session_start();
	if(@$_SESSION["login"]!="go")
	{
		header("Location: loginpage.php");
		exit();
	}
?>
```

بدین‌ترتیب، درصورتی‌که مقدار متغیر سِشِن به‌نام login مخالف go باشد، کاربر به صفحه loginpage.php هدایت خواهد شد.

نکته: علامت @ در ابتدای $_SESSION موجب می‌شود که درصورت عدم وجود متغیر سِشِن به‌نام login، پیغام‌خطا تولید نشود و برنامه، به‌کار خود ادامه‌دهد.

توضیح و تکمیل CMS جلسه قبل

در جلسه قبل، یک CMS نسبتاً کامل برای درج و مدیریت مقالات مختلف پیرامون آموزش ساخت روبات و نظرات کاربران درباره آن مقالات نوشتیم. در این جلسه، علاوه‌بر توضیح کامل کدهای CMS مذکور، اقدام به تکمیل آن و افزودن برخی کنترل‌های امنیتی اولیه و ساده می‌نماییم (کنترل‌های امنیتی پیشرفته‌تر موضوع جلسات آینده خواهد بود).

هدف و نمای کلی سایت

در طراحی و برنامه‌نویسی یک وب‌سایت، همیشه باید ابتدا هدف سایت و نمای کلی که کاربر با آن مواجه خواهد شد، مشخص‌باشد. هدف این سایت، ارائه برخی مقالات آموزشی درباره ساخت روبات است و نمای کلی آن، بدین‌ترتیب است که در بالای همه صفحات، نشان (Logo) سایت به‌نمایش درآمده و در سمت راست نیز منوی اصلی که شامل چهار گزینه است، ظاهر می‌شود. با انتخاب هر گزینه نیز بخش مربوطه در سمت چپ ظاهر خواهد شد. البته گزینه آلبوم تصاویر از این قاعده مستثنی است و صفحه مربوطه‌آن‌در یک پنجره جداگانه ظاهر خواهد شد. حال با توجه به این توضیحات، به‌سراغ کد صفحات سایت، با همان ترتیبی که کاربران سایت با آنها مواجه می‌شوند، می‌رویم.

نکته بسیار مهم: برای پرهیز از مشکلات احتمالی در کار با Sessionها، توابع header و... تمامی فایل‌ها را توسط یک ویرایشگر مناسب مثل NotePad++یا Rapid PHP با کدگذاری UTF-8 Without BOM ذخیره کنید (درصورت استفاده از DreamWeaver برای نوشتن کدها، در زمان ذخیره‌کردن فایل توسط دستور Save As...، گزینه BOM (Byte Order Mark) را در پنجره ذخیره‌سازی فایل، غیرفعال کنید).

فایل تنظیمات سایت config.php

این فایل، هیچ‌گاه مستقیماً مشاهده نخواهد شد! شاید تعجب‌کنید که در این صورت، چرا آنرا در ابتدای فهرست بقیه فایل‌ها ذکر کرده‌ایم. برای درک اهمیت این‌فایل، باید به این نکته دقت‌کنید که در تمامی کدهای ما، برخی اطلاعات و کارها بطور مشابه و یا با اندکی تغییر تکرار می‌شوند. برای مثال، عنوان سایت در تمامی صفحات سایت درج می‌شود یا هر زمان نیاز به استخراج اطلاعات از پایگاه داده‌ها داشته‌باشیم، باید نام سرور، نام کاربری و رمز عبور پایگاه داده‌ها را وارد نماییم. با وجود آنکه درج همه این اطلاعات برای یک سایت در صفحات مختلف، به‌تنهایی کار سختی است، قسمت سخت‌تر کار زمانی آشکار می‌شود که بخواهیم از همین کد برای یک سایت دیگر نیز استفاده‌کنیم. تصور کنید که ویرایش تمامی اسکریپت‌ها و اصلاح کدها چقدر می‌تواند کار طاقت‌فرسایی باشد. اهمیت فایل تنظیمات سایت در اینجا مشخص می‌شود: می‌توانیم اطلاعات ضروری و تکراری سایت را در فایل config.php (یا هر نام دلخواه دیگری که خودتان دوست دارید برای این فایل درنظر بگیرید) قراردهیم و آنرا در سایر اسکریپت‌هایی که به این اطلاعات نیاز دارند، ضمیمه‌نماییم. بدین‌ترتیب، درصورت نیاز به تغییر این اطلاعات، فقط یک‌بار و درون این فایل، تغییرات را اعمال می‌کنیم و از آنجا که تمامی اسکریپت‌های‌دیگر از اطلاعات موجود در این فایل استفاده می‌کنند، به‌طور خودکار اصلاح خواهند شد. کد درون این فایل، با کمی تغییر نسبت‌به جلسه قبل، به‌شرح زیر است:

```
001. <?php
002.     //Copyright محمد مصطفی شهرکی @ http://www.ncis.ir
003.     define('TITLE', 'وب سایت آموزش ساخت روبات');
004.     define('URL', 'http://localhost/cms');
005.     define('HOST', 'localhost');
006.     define('USER', 'root');
007.     define('PASS', 'ncis');
008.     define('DB', 'cms');
009.     define('UN', '3fa70cadf058bd0bf704dabd102195ef');
010.     define('PW', '24e7d1bb55dffdc6a9bdb39c543d5c74');
011. ?>
```

همان‌طور که ملاحظه می‌کنید، به‌جای تعریف مقادیر توسط متغیرها، از دستور define برای تعریف ثابت‌های متناظر با متغیرهای جلسه قبل استفاده شده‌است. علل این تغییر، عبارتند از:

- ✓ ثابت‌ها برخلاف متغیرها، فضای حافظه را اشغال نمی‌کنند.
- ✓ ثابت‌ها برخلاف متغیرها، دارای محدوده اعتبار (Scope) نیستند و از آنها می‌توان در تمامی محدوده‌ها (داخل توابع و...) به‌راحتی استفاده‌نمود.
- ✓ برخلاف متغیرها، در سایر اسکریپت‌ها نمی‌توان مقدار ثابت‌ها را تغییر داد و این مسئله، امنیت نسبتاً بیشتری برای سایت فراهم می‌کند.

بدیهی‌است که با توجه به تغییرات این فایل، باید تمامی اسکریپت‌های دیگر براساس این تغییرات بازنویسی‌شوند که به‌ترتیب، درمورد هر فایل، تغییرات لازم را ذکر خواهیم‌کرد.

فایل ارتباط با پایگاه داده‌ها db.php

این فایل نیز مشابه فایل config.php، مستقیماً توسط کاربر مورد استفاده قرار نمی‌گیرد اما نقشی حیاتی را در سایت ما ایفا می‌کند. وظیفه این فایل، برقراری ارتباط با پایگاه داده‌ها و انجام دستورات موردنیاز ما برروی آن و دریافت اطلاعات لازم از آن است. کد فایل به‌صورت زیر است:

```
001. <?php
002.     //Copyright محمد مصطفی شهرکی @ http://www.ncis.ir
003.     require_once 'config.php';
004.
005.     class Articles {
006.         private $link = null;
007.
008.         public function Articles() {
009.             $this->Connect();
010.         }
011.
012.         public function CommentsCount($id) {
013.             $comments = new Comments();
014.             $result = $comments->SelectByArticle($id);
015.             return mysql_num_rows($result);
016.         }
017.
```

```
018.        private function Connect() {
019.            if($this->link == null || !mysql_ping($this->link)) {
020.                $this->link = mysql_connect(HOST, USER, PASS);
021.            }
022.            mysql_select_db(DB);
023.            mysql_query('SET NAMES \'utf8\'');
024.        }
025.
026.        public function Delete($id) {
027.            $this->Connect();
028.            $result = mysql_query("UPDATE `articles` SET `status`='deleted' WHERE (`id`='{$id}')");
029.            return mysql_affected_rows();
030.        }
031.
032.        public function GetNewID() {
033.            $this->Connect();
034.            $id = 0;
035.            $result = mysql_query('SELECT * FROM `articles` ORDER BY `id` DESC LIMIT 1');
036.            if($result !== false && mysql_num_rows($result) > 0) {
037.                $id = mysql_result($result, 0, 0);
038.            }
039.            $id++;
040.            return $id;
041.        }
042.
043.        public function IncrementVisits($id) {
044.            $this->Connect();
045.            $result = mysql_query("UPDATE `articles` SET `visits`=`visits`+1 WHERE (`id`='{$id}')
                                LIMIT 1");
046.            return mysql_affected_rows();
047.        }
048.
049.        public function Insert($title, $abstract, $body, $filename) {
050.            $this->Connect();
051.            $id = $this->GetNewID();
052.            $result = mysql_query("INSERT INTO `articles` VALUES ('{$id}', '{$title}', '{$abstract}',
                                '{$body}', '{$filename}', '0', 'normal')");
053.            return mysql_affected_rows();
054.        }
055.
056.        public function Recover($id) {
057.            $this->Connect();
058.            $result = mysql_query("UPDATE `articles` SET `status`='normal' WHERE (`id`='{$id}')");
059.            return mysql_affected_rows();
060.        }
061.
062.        public function SelectAll() {
063.            $this->Connect();
064.            $result = mysql_query('SELECT * FROM `articles` ORDER BY `id`');
065.            return $result;
066.        }
067.
068.        public function SelectNormal() {
069.            $this->Connect();
070.            $result = mysql_query('SELECT * FROM `articles` WHERE (`status`=\'normal\') ORDER BY `id`');
071.            return $result;
072.        }
073.
074.        public function SelectRow($id) {
075.            $this->Connect();
076.            $result = mysql_query("SELECT * FROM `articles` WHERE (`id`='$id') ORDER BY `id` LIMIT 1");
077.            return $result;
078.        }
079.
080.        public function Update($id, $title, $abstract, $body, $filename, $visits) {
081.            $this->Connect();
082.            $result = mysql_query("UPDATE `articles` SET `title`='{$title}', `abstract`='{$abstract}',
                                `body`='{$body}', `filename`='{$filename}', `visits`='{$visits}'
                                WHERE (`id`='{$id}')");
083.            return mysql_affected_rows();
084.        }
085.    }
086.
087.    class Comments {
088.        private $link = null;
089.
090.        public function Comments() {
091.            $this->Connect();
092.        }
```

```
093.
094.        private function Connect() {
095.            if($this->link == null || !mysql_ping($this->link)) {
096.                $this->link = mysql_connect(HOST, USER, PASS);
097.            }
098.            mysql_select_db(DB);
099.            mysql_query('SET NAMES \'utf8\'');
100.        }
101.
102.        public function GetNewID() {
103.            $this->Connect();
104.            $id = 0;
105.            $result = mysql_query('SELECT * FROM `comments` ORDER BY `id` DESC LIMIT 1');
106.            if($result !== false && mysql_num_rows($result) > 0) {
107.                $id = mysql_result($result, 0, 0);
108.            }
109.            $id++;
110.            return $id;
111.        }
112.
113.        public function Insert($aid, $name, $body, $status) {
114.            $this->Connect();
115.            $id = $this->GetNewID();
116.            $result = mysql_query("INSERT INTO `comments` VALUES ('{$id}', '{$aid}', '{$name}', '{$body}',
                                    '0', '{$status}')");
117.            return mysql_affected_rows();
118.        }
119.
120.        public function SelectAll() {
121.            $this->Connect();
122.            $result = mysql_query('SELECT * FROM `comments` ORDER BY `id`');
123.            return $result;
124.        }
125.
126.        public function SelectByArticle($aid) {
127.            $this->Connect();
128.            $result = mysql_query("SELECT * FROM `comments` WHERE (`aid`='{$aid}' AND `status`<>'deleted')
                                    ORDER BY `id`");
129.            return $result;
130.        }
131.
132.        public function SelectRow($id) {
133.            $this->Connect();
134.            $result = mysql_query("SELECT * FROM `comments` WHERE (`id`='{$id}') ORDER BY `id` LIMIT 1");
135.            return $result;
136.        }
137.
138.        public function Set($id, $switch) {
139.            $this->Connect();
140.            switch(strtolower($switch)) {
141.                case 'confirmed':
142.                    $result = mysql_query("UPDATE `comments` SET `confirmed`='1' WHERE (`id`='{$id}')
                                            LIMIT 1");
143.                    break;
144.                case 'waiting':
145.                    $result = mysql_query("UPDATE `comments` SET `confirmed`='0' WHERE (`id`='{$id}')
                                            LIMIT 1");
146.                    break;
147.                case 'private':
148.                    $result = mysql_query("UPDATE `comments` SET `status`='private' WHERE (`id`='{$id}')
                                            LIMIT 1");
149.                    break;
150.                case 'public':
151.                    $result = mysql_query("UPDATE `comments` SET `status`='public' WHERE (`id`='{$id}')
                                            LIMIT 1");
152.                    break;
153.            }
154.            return mysql_affected_rows();
155.        }
156.
157.        public function Update($id, $aid, $body, $status) {
158.            $this->Connect();
159.            $result = mysql_query("UPDATE `comments` SET `aid`='{$aid}', `body`='{$body}',
                                    `status`='{$status}' WHERE (`id`='{$id}')");
160.            return mysql_affected_rows();
161.        }
162.    }
163. ?>
```

تغییرات این فایل نسبت به جلسه قبل:

- تبدیل تگ PHP از حروف بزرگ به کوچک درجهت رعایت استانداردهای جدید
- حذف پرانتزها در خط 003 (دستور require_once یک تابع نیست و لذا نیازی به پرانتز ندارد)
- تغییر ترتیب قرارگیری توابع داخلی کلاس و مرتب‌سازی آنها براساس حروف الفبا به‌منظور سهولت در مراجعات بعدی
- حذف عناصر داخلی کلاس به نام‌های host$، user$، pass$ و db$ و استفاده از ثابت‌های HOST، USER، PASS و DB به‌جای آنها
- اضافه‌شدن عنصر داخلی کلاس به‌نام link$ که وظیفه کنترل‌کردن اتصال به پایگاه داده‌ها را برعهده‌دارد
- تغییر سازنده کلاس‌ها به‌نحوی که صرفاً تابع Connect را فراخوانی‌نماید
- تغییر تابع Connect به‌نحوی که ابتدا بررسی‌کند که فقط درصورتی‌که متغیر کنترل‌کننده اتصال (link$) وجود ندارد یا اتصال آن قطع شده‌است، اتصال ایجاد کند
- قراردادن متغیرهای درون رشته‌ها (با گیومه جفت) در داخل آکولاد باز و بسته به‌منظور رعایت استانداردها و پرهیز از مشکلات ناخواسته احتمالی

توضیح کد:

از آنجا که در این فایل، از اصول شئ‌گرایی در برنامه‌نویسی استفاده شده‌است، اجازه‌دهید کمی درباره برنامه‌نویسی شئ‌گرا توضیح‌دهیم. در برنامه‌نویسی سنتی که به آن، رویه‌گرا (Procedural) نیز می‌گویند، هر برنامه از دو بخش اصلی تشکیل می‌شود: داده‌ها و کد (که با داده‌ها سروکار دارد). در این روش، یک بخش از کد می‌تواند به داده‌های خاصی دسترسی داشته‌باشد یا از دسترسی به آنها محروم باشد؛ اما درصورت داشتن دسترسی، دیگر نمی‌توان سطحی برای دسترسی مشخص‌نمود (برای مثال، نمی‌توان داده‌ای را بصورت فقط‌خواندنی در اختیار یک کد قرار داد). بعلاوه در برنامه‌های بزرگ، درک داده‌ها و کدهای جدا از هم بسیار مشکل خواهد شد. درنتیجه، به‌لطف برنامه‌نویسی شئ‌گرا، داده‌ها و کدهای مربوط به یک عنصر خاص را درون یک قالب به‌نام کلاس (Class) قرار می‌دهیم. بدین‌ترتیب، این داده‌ها و کدها در داخل کلاس به‌خوبی با یکدیگر تعامل داشته و دسترسی کاملی به یکدیگر دارند، اما در خارج از کلاس، می‌توان توسط کلمات کلیدی private، public و... سطح دسترسی به این عناصر را محدود کرد. باید دقت‌کنید که یک کلاس به‌تنهایی هیچ کاری انجام نمی‌دهد زیرا فقط یک قالب است و برای استفاده از آن، باید ازروی این قالب، متغیرهایی تعریف‌کنیم که به آنها شئ می‌گویند. اشیاء ایجادشده از کلاس، درحقیقت نمونه‌های آن کلاس هستند که تمامی خواص و عناصری را که درون کلاس تعریف کرده‌ایم، دارا می‌باشند؛ اما باید به این نکته بسیار مهم توجه‌کنید که فقط عناصر public ازطریق اشیاء قابل دسترسی و فراخوانی هستند. برای مثال، اگر یک کلاس حاوی یک عنصر public و یک عنصر private باشد، ازطریق شئ ایجادشده ازروی کلاس مربوطه فقط می‌توانیم به عنصر public آن دسترسی داشته‌باشیم. البته در داخل کد عنصر public، می‌توانیم عنصر private را مورد دستیابی قراردهیم.

نکته مهم: مفهوم کلاس را در شئ‌گرایی با کلمه مشابه (کلاس) در CSS اشتباه نگیرید. در انتهای این جلسه، مفهوم کلاس‌های CSS را توضیح خواهیم داد.

برای درک بهتر، به کد زیر دقت‌کنید:

```
class Student {
    private $avg = rand(1, 20);
    public function Average() {
        return $this->avg;
    }
}
$stu = new Student();
echo $stu->Average();
$stu->avg = 15; // Error
```

همان‌طور که در کد فوق مشخص‌است، عنصر avg$ یک عنصر خصوصی (private) است. در درون بدنه کلاس (بین آکولاد باز ابتدای کلاس و آکولاد بسته انتهای آن) می‌توانیم به این عنصر دسترسی داشته و درصورت تمایل آنرا تغییر دهیم (به دستور داخل تابع Average دقت‌کنید). حال در بیرون از کلاس، یک شئ به‌نام stu$ ازروی آن ایجاد می‌کنیم. همان‌طور که ملاحظه می‌کنید، می توانیم با فراخوانی تابع Average ازطریق شئ، معدل دانشجو را نمایش‌دهیم اما نمی‌توانیم ازطریق همین‌شئ، عنصر avg$آنرا ویرایش‌کنیم (زیرا private است و مکانیزم public خاصی برای ویرایش آن نوشته‌ایم). درصورت تمایل، می‌توانیم یک تابع public به کلاس Student اضافه‌کنیم که وظیفه تغییر مقدار عنصر avg$ را برعهده داشته‌باشد. البته این ویژگی تنها مزیت شئ‌گرایی نیست! امتیاز اصلی شئ‌گرایی در آن است که یک‌بار ساختار را تعریف می‌کنیم و سپس می‌توانیم به‌تعداد دلخواه ازروی آن شئ ایجاد کنیم و هر شئ، رفتار و مقادیر خاص خودش را خواهد داشت (مثلاً هر دانشجو می‌تواند معدل متفاوتی داشته‌باشد).

با توجه به توضیحات فوق، به‌سراغ کلاس Articles می‌رویم. این کلاس، وظیفه تعامل با جدول Articles را در پایگاه داده‌ها برعهده دارد و اعمالی نظیر درج مقاله جدید، ویرایش، حذف، افزایش آمار بازدید و... را انجام می‌دهد (البته نه خود کلاس، بلکه اشیاء ایجادشده ازروی آن).

ابتدا در خط 006 یک عنصر خصوصی به‌نام link$ با مقدار اولیه null ایجاد می‌شود که وظیفه آن، نگهداری اتصال ایجادشده به MySQL است. سپس در خطوط 008 تا 010، تابع سازنده کلاس Articles تعریف می‌شود که این تابع، به‌نوبه خود تابع خصوصی Connect را از درون کلاس، فراخوانی می‌کند. تابع سازنده، یک تابع public و هم‌نام کلاس است که به‌طور خودکار در زمان ایجاد یک شئ ازروی کلاس، فراخوانی خواهد شد. در خطوط 012 تا 016، تابع CommentsCount تعریف می‌شود. وظیفه این تابع، دریافت شماره یک مقاله و شمارش تعداد نظرات آن مقاله است. تابع مذکور، این وظیفه را ازطریق ایجاد یک شئ از کلاس Comments و فراخوانی تابع SelectByArticle و ارسال شماره مقاله دریافت‌شده برای آن و درنهایت شمارش تعداد رکوردهای بازگردانده‌شده توسط آن‌تابع، انجام می‌دهد (کلاس Comments و تابع SelectByArticle بعد از کلاس Articles توضیح داده خواهند شد). در خطوط 018 تا 024، تابع خصوصی Connect تعریف می‌شود. این تابع، ابتدا عنصر خصوصی link$ را بررسی می‌کند و اگر این‌عنصر null بود (هنوز اتصال برقرار نشده‌است) یا خروجی تابع mysql_ping برروی آن برابر با false بود (اتصال قطع شده‌است)، یک اتصال جدید به MySQL برقرارکرده و آن‌اتصال را در عنصر link$ قرار می‌دهد. سپس پایگاه داده‌ها که در فایل config.php تعیین شده‌است، انتخاب می‌شود و دستور زیر روی آن اجرا می‌گردد:

```
SET NAMES 'utf8'
```

این دستور، به MySQL اعلام می‌کند که اطلاعات تبادل‌شده با پایگاه داده‌ها از کدگذاری Unicode (UTF-8) استفاده می‌کنند. این مسئله برای ذخیره و نمایش صحیح اطلاعات فارسی اهمیت بسیار زیادی دارد. در خطوط 026 تا 030، تابع Delete تعریف می‌شود که شماره یک مقاله را دریافت‌کرده و آنرا حذف می‌کند. البته ازآنجا که در طراحی ما، حذف بصورت منطقی درنظر گرفته شده‌است، عمل حذف صرفاً با تغییر رکورد مقاله موردنظر و مقداردهی فیلد status آن با مقدار deleted انجام می‌پذیرد تا بعداً بتوانیم درصورت تمایل، مقاله مذکور را مجدداً بازیابی‌کنیم. خطوط 032 تا 040 مربوط‌به تابع GetNewID هستند که وظیفه آن، تولید یک id برای درج رکورد جدید است. روش کار نیز بسیار ساده‌است: ابتدا متغیر $id با مقدار صفر تعریف می‌شود. سپس، آخرین‌رکورد جدول برحسب ID استخراج می‌شود:

```
SELECT * FROM `articles` ORDER BY `id` DESC LIMIT 1
```

یعنی ابتدا رکوردها برحسب فیلد id مرتب می‌شوند (ORDER BY `id`) اما بصورت نزولی (DESC) و سپس، اولین‌رکورد در ترتیب نزولی (که درواقع آخرین رکورد جدول است) جدا می‌گردد (LIMIT 1). حال اگر دستور مربوطه خطا نداشته‌باشد (خروجی دستور برابر با false نباشد) و ضمناً تعداد رکوردهای بازگردانده‌شده بیشتر ازصفر باشد (جدول خالی نباشد)، مقدار فیلد id رکورد استخراج‌شده در متغیر $id قرار می‌گیرد:

```
$id = mysql_result($result, 0, 0);
```

تابع mysql_result سه پارامتر دریافت می‌کند و یک مقدار باز می‌گرداند. پارامتر اول، خروجی دستور mysql_query و درواقع، حاوی رکوردهای استخراج‌شده است. پارامتر دوم، شماره سطر موردنظر است که در اینجا، چون فقط یک‌سطر داریم، شماره آن صفر خواهد بود (شماره‌گذاری سطرها از صفر آغاز می‌شود). پارامتر سوم نیز شماره ستونی است که می‌خواهیم مقدار آنرا از سطر مربوطه استخراج‌کنیم که از آنجا که ستون (فیلد) id اولین ستون است، این پارامتر نیز مقدار صفر خواهد داشت (شماره‌گذاری ستون‌ها نیز از صفر شروع می‌شود). بدین‌ترتیب، اگر جدول خالی باشد، $id برابر با صفر و درغیر اینصورت، برابر با آخرین id موجود است. درنتیجه، کافی‌است به این متغیر یک‌واحد اضافه‌کنیم (خط 039) تا id رکورد جدید را به‌دست آوریم. در خطوط 043 تا 047 تابع IncrementVisits تعریف می‌شود که شماره مقاله را دریافت‌کرده و تعداد بازدیدهای آنرا یک‌واحد افزایش می‌دهد. این تابع، این‌کار را با اضافه‌کردن مقدار ۱ به مقدار قبلی فیلد visits رکوردی که id آن برابر با $id دریافتی بعنوان پارامتر است، انجام می‌دهد. خروجی این تابع، تعداد رکوردهای تغییریافته براساس دستور مربوطه است. درنتیجه، اگر رکوردی تغییر نکند (مقاله‌ای با id برابر با $id وجود نداشته‌باشد)، خروجی این تابع صفر خواهد بود. تابع بعد، Insert نام‌دارد که در خطوط 049 تا 054 تعریف شده‌است و با دریافت عنوان ($title)، چکیده ($abstract)، بدنه ($body) و نام فایل ($filename) بعنوان پارامتر، عمل درج یک مقاله جدید را انجام می‌دهد. برای این‌کار نیز ابتدا با فراخوانی تابع GetNewID شماره id مقاله جدید را به‌دست آورده و مقاله جدید را با تعداد بازدید صفر و وضعیت عادی (حذف‌نشده) (مقدار normal برای فیلد status) درج می‌کند. در خطوط 056 تا 060 تابع Recover معرفی شده‌است که عکس تابع Delete عمل می‌کند و مقاله حذف‌شده را با دریافت شماره آن، بازیابی می‌نماید. برای این‌کار نیز به‌سادگی فیلد status رکورد مربوط به مقاله را مجدداً با مقدار normal مقداردهی می‌نماید. خطوط 062 تا 066 حاوی تابع SelectAll هستند که تمامی مقالات را از جدول استخراج می‌کند. در خطوط 068 تا 072 تابع SelectNormal تعریف شده‌است که مقالات حذف‌نشده را (مقالاتی که فیلد status آنها دارای مقدار normal است)، استخراج می‌نماید. در خطوط 074 تا 078 تابع SelectRow قرار دارد که شماره مقاله را دریافت‌کرده و صرفاً همان مقاله را از جدول استخراج می‌کند. در انتهای کلاس Articles نیز تابع Updateدرخطوط 080 تا 084 تعریف شده‌است که شماره یک مقاله، عنوان، چکیده، بدنه، نام فایل و تعداد بازدید آنرا دریافت‌کرده اطلاعات مقاله مربوطه را ویرایش می‌کند.

در ادامه، کلاس Comments تعریف شده‌است که حاوی توابعی برای کار با جدول comments (نظرات) می‌باشد. از آنجا که عنصر داخلی $link، سازنده کلاس، تابع داخلی Connect، تابع GetNewID، تابع Insert، تابع SelectAll، تابع SelectRow و تابع Update مشابه توابع همنام خود در کلاس Articles هستند و فقط برخی پارامترهای آنها (مثل نام جدول و...) متفاوت است، از توضیح آنها خودداری می‌کنیم. توابع جدید در این کلاس، شامل موارد زیر است:

در خطوط 126 تا 130 تابع SelectByArticle تعریف شده‌است که شماره مقاله را بعنوان پارامتر ورودی دریافت‌کرده و نظرات مرتبط با آن مقاله را (که فیلد aidآنها با شماره مقاله مربوطه برابر است)، استخراج می‌کند. در خطوط 138 تا 155 نیز تابع Set قرار دارد که دارای دو پارامتر ورودی است: پارامتر اول شماره نظر و پارامتر دوم، مشخص‌کننده نوع عملی است که قصد داریم برروی آن نظر انجام‌دهیم که یکی از حالت‌های زیر است:

- confirmed : با تنظیم فیلد confirmed نظر مربوطه برروی عدد یک، آنرا به وضعیت «تأییدشده» تغییر می‌دهد
- waiting : با تنظیم فیلد confirmed نظر مربوطه برروی عدد صفر، آنرا به وضعیت «در انتظار تأیید» تغییر می‌دهد
- private : با تنظیم فیلد status نظر مربوطه با مقدار private، آنرا به وضعیت «خصوصی» تغییر می‌دهد
- public : با تنظیم فیلد status نظر مربوطه با مقدار public، آنرا به وضعیت «عمومی» تغییر می‌دهد

در نهایت، بعد از انجام یکی از اعمال فوق برروی نظری که شماره آن توسط $id برای این تابع ارسال شده‌است، تعداد رکوردهای تحت تأثیر این تابع بازگردانده خواهد شد.

صفحه اصلی سایت index.php

این فایل، اولین صفحه‌ای است که کاربران سایت با واردکردن نشانی آن (URL) درون مرورگر خود، آنرا مشاهده می‌کنند. درحقیقت، هرگاه آدرس یک سایت یا یک پوشه درون سایت را در مرورگر بنویسیم، درصورتی‌که تنظیمات پیش‌فرض سرویس‌دهنده وب (Apache، IIS و...) را تغییر نداده‌باشیم، فایل index.php موجود در آن سایت یا پوشه مشخص‌شده اجرا خواهد شد. بنابراین، به‌طور ساده می‌توانیم بگوییم که index.php نقطه ورود کاربران به سایت محسوب می‌شود. در طراحی ما، قرار است صفحه سایت به دو قسمت اصلی (قسمت بالا برای نشان سایت و قسمت پایین برای بدنه صفحه) تقسیم‌شود و قسمت پایین نیز به‌نوبه خود، به دو قسمت (سمت راست برای منوی اصلی و سمت چپ برای محتوای صفحه متناظر با هر گزینه منو) تقسیم‌گردد. برای این‌کار، در این آموزش از قاب‌بندی استفاده شده‌است که البته راه‌های بهتری نیز برای این‌کار وجوددارد که در جلسات آینده با آنها آشنا خواهیم شد. با توجه به توضیحات فوق‌الذکر، کد فایل index.php به‌صورت زیر خواهد بود:

```
001. <?php
002.     //Copyright محمد مصطفی شهرکی @ http://www.ncis.ir
003.     require_once 'config.php';
004. ?>
005. <!doctype html>
```

```
006. <html dir="rtl">
007. <head>
008. <meta charset="utf-8"/>
009. <title><?php echo TITLE; ?></title>
010. <link href="style.css" rel="stylesheet" type="text/css"/>
011. </head>
012. <frameset rows="100px,*">
013.     <frame frameborder="0" name="topFrame" noresize="noresize" scrolling="no" src="top.php"/>
014.     <frameset cols="*,200px">
015.         <frame frameborder="0" name="mainFrame" noresize="noresize" scrolling="yes" src="main.php"/>
016.         <frame frameborder="0" name="linksFrame" noresize="noresize" scrolling="no" src="links.php"/>
017.     </frameset>
018. </frameset>
019. <noframes>
020. <body>
021. خطا: مرورگر شما از قاب بندي پشتيباني نمي كند.
022. </body>
023. </noframes>
024. </html>
```

تغییرات ایجاد شده در این فایل نسبت به جلسه قبل:

- تبدیل اسامی تگ‌ها از حروف بزرگ به حروف کوچک درجهت رعایت استانداردهای جدید
- تغییر ترتیب خاصیت تمامی تگ‌ها و قرارگیری خاصیت‌ها به‌ترتیب حروف الفبا جهت سهولت در دسترسی‌های بعدی
- حذف پرانتزهای دستور require_once در خط 003
- اضافه‌شدن تگ doctype (خط 005) برای اعلام زبان مورد استفاده در صفحه (در اینجا از <!doctype html> که معرف HTML5 هست، استفاده شده‌است)
- تغییر خط 008 و کوتاه‌شدن آن به‌منظور رعایت ساختار HTML5
- تغییر خط 009 و استفاده از ثابت TITLE به‌جای متغیر $title (به‌دلیل تغییرات ایجادشده در فایل config.php)
- اضافه‌شدن عبارت px به اعداد ۱۰۰ و ۲۰۰ در خطوط 012 و 014 درجهت رعایت استانداردهای تعیین اندازه عناصر برحسب پیکسل در زبان HTML

توضیح کد:

همان‌طور که ملاحظه می‌کنید، در خط 010 فایل style.css بعنوان قالب صفحه مشخص شده‌است. در طراحی صفحات وب، بطور کلی از CSS برای تغییر ظاهر پیش‌فرض عناصر استفاده می‌شود. همان‌طور که حتماً می‌دانید، نرم‌افزارهای مرورگر وب (مثل Internet Explorer، FireFox، Chrome، Opera، Safari و...) توسط شرکت‌های مختلف و درنتیجه برنامه‌نویسان متفاوتی تولید می‌شوند و این برنامه‌نویسان براساس تصمیم‌گیری‌های داخلی آن شرکت‌ها، ظاهر پیش‌فرض دلخواه خود را برای عناصر مختلف درنظر می‌گیرند (برای مثال، تگ <hr/> که برای تولید یک خط جداکننده افقی به‌کار می‌رود، در مرورگرهای IE و FF با اندکی تغییر در ظاهر آن به‌نمایش می‌آید). حال برای پرهیز از اینگونه تفاوت‌های ظاهری و تنظیم صفحه وب به‌نحوی که در تمامی مرورگرها بصورت یکسان ظاهر شود و همچنین تعیین یکسری تنظیمات پیشرفته ظاهری، از CSS استفاده می‌کنیم. استفاده از CSS برای تنظیم ظاهر صفحات امروزه آنقدر مهم و ضروری شده‌است که کنسرسیوم جهانی وب (W3C) قانون زیر را بعنوان یک استاندارد پذیرفته‌است:

از HTML صرفاً برای تعیین ساختار صفحات (در کجا یک پاراگراف ایجاد شود، در کدام قسمت یک جدول تولید گردد و...) استفاده‌کنید و از CSS برای تنظیم نحوه نمایش ظاهری آنها (پاراگراف‌ها به چه‌صورت ظاهر شوند، حاشیه بین خانه‌های جداول چقدر باشد، اندازه و نوع قلم صفحه چه باشد و...) بهره‌گیرید.

بعلاوه استفاده از CSS برای تنظیم ظاهر صفحه یک مزیت عمده دیگر نیز دارد و آن اینکه یک‌بار اعلام می‌کنیم که یک تگ خاص چگونه ظاهر شود و از آن به‌بعد در تمامی قسمت‌ها، تگ مربوطه به همان‌شکلی که می‌خواهیم ظاهر خواهد شد. از آنجا که CSS موضوع اصلی این آموزش نیست، کد فایل style.css را در انتهای این جلسه توضیح خواهیم داد.

همان‌طور که در مقاله «آموزش طراحی صفحات وب با HTML» ذکر شده‌است (این مقاله به‌نام Web Design With HTML یا به‌اختصار WDWHTML در وب‌سایت علمی – تخصصی nCIS به‌نشانی http://www.ncis.ir برای دریافت بصورت رایگان موجود است)، برای تقسیم صفحه به چند بخش، از تگ frameset استفاده می‌شود و هر بخش توسط تگ frame به صفحه وب مربوطه متصل می‌شود. درنتیجه، در خط 012 توسط کد زیر:

```
<frameset rows="100px,*">
```

صفحه به دو بخش افقی تقسیم می‌شود که بخش بالایی دارای ارتفاع ۱۰۰ پیکسل بوده و بخش پایین، مابقی صفحه را در بر خواهد گرفت.

سپس در خط 013 توسط کد زیر:

```
<frame frameborder="0" name="topFrame" noresize="noresize" scrolling="no" src="top.php"/>
```

قاب بالایی به صفحه top.php متصل می‌شود و درنتیجه خروجی صفحه مذکور در قسمت بالای صفحه index.php ظاهر خواهد شد. ضمناً نام قاب بالایی، topFrame تعیین‌شده و امکان تغییر اندازه و پیمایش آن وجود نخواهد داشت. دقت‌کنید که در استاندارد HMTL، خاصیت frameborder را فقط می‌توان با مقادیر 0 یا 1 مقداردهی نمود که به‌ترتیب بیانگر وجود یا عدم‌وجود خط حاشیه برای قاب می‌باشد و برای تنظیم ضخامت حاشیه (درصورت تمایل)، باید از CSS استفاده‌شود. از آنجا که ما می‌خواهیم حاشیه‌ای وجود نداشته‌باشد، از عدد 0 برای این خاصیت استفاده کرده‌ایم.

در ادامه، در خط 014 توسط کد زیر:

```
<frameset cols="*,200px">
```

قاب پایین صفحه به‌نوبه خود به دو قاب عمودی دیگر تقسیم می‌شود که در اینجا، قاب سمت راست اندازه ثابت‌دارد (۲۰۰ پیکسل) و قاب سمت چپ، بقیه صفحه را می‌پوشاند. در ادامه، خطوط 015 و 016 توسط کدهای زیر:

```
<frame frameborder="0" name="mainFrame" noresize="noresize" scrolling="yes" src="main.php"/>
<frame frameborder="0" name="linksFrame" noresize="noresize" scrolling="no" src="links.php"/>
```

موجب می‌شوند تا قاب سمت چپ به‌نام mainFrame به صفحه main.php و قاب سمت راست به‌نام linksFrame به صفحه links.php متصل گردد. همچنین قاب سمت چپ دارای قابلیت پیمایش بوده و درنتیجه، پیمایشگرهای عمودی و افقی به ان افزوده خواهند شد. از خاصیت name تگ frame می‌توان در لینک‌های مختلف استفاده‌نمود، بدین‌ترتیب که خاصیت target تگ a را با نام قاب مربوطه مقداردهی می‌کنیم تا درصورت کلیک برروی لینک مربوطه، صفحه مربوطه‌به خاصیت href آن‌لینک در قاب مشخص‌شده باز شود.

نکته مهم در زمان استفاده از قاب‌بندی آن‌است که صفحه شامل قاب‌بندی (در اینجا فایل index.php) درحقیقت شامل بدنه نیست و بدنه آنرا صفحات دیگر تشکیل می‌دهند. لذا قاعدتاً این فایل نباید شامل تگ body باشد اما ممکن‌است به هر دلیلی (مثل قدیمی‌بودن مرورگر کاربر و...) امکان استفاده از قاب‌بندی میسر نباشد. در چنین وضعیتی، از تگ noframes استفاده می‌کنیم و بدنه (تگ body) دلخواه را برای صفحه درون این تگ قرار می‌دهیم (خطوط 019 تا 023).

صفحه نشان سایت top.php

این فایل صرفاً وظیفه نمایش نشان (Logo) سایت را برعهده دارد. درنتیجه، عملیات خاصی درون آن انجام نمی‌گیرد و کد آن، به‌سادگی زیر است:

```
001. <?php
002.     //Copyright محمد مصطفی شهرکی @ http://www.ncis.ir
003.     require_once 'config.php';
004. ?>
005. <!doctype html>
006. <html dir="rtl">
007. <head>
008. <meta charset="utf-8"/>
009. <title><?php echo TITLE; ?></title>
010. <link href="style.css" rel="stylesheet" type="text/css"/>
011. </head>
012. <body class="nomargin">
013. <a href="<?php echo URL; ?>" target="_top"><img src="title.png" style="height:100px; width:100%"/></a>
014. </body>
015. </html>
```

تغییرات این فایل نسبت به جلسه قبل:

- تبدیل اسامی تگ‌ها به حروف کوچک
- مرتب‌شدن خاصیت‌های تگ‌ها به‌ترتیب الفبا
- حذف پرانتزهای دستور require_once در خط 003
- اضافه‌شدن تگ doctype در خط 005
- کوتاه‌شدن تگ meta در خط 008
- جایگزینی متغیرهای $title و $url با ثابت‌های TITLE و URL در خطوط 009 و 013
- حذف جدول (تگ‌های table و tr و td) و درج مستقیم تصویر در صفحه در خط 013

توضیح کد:

در این فایل، ابتدا درخط 010 فایل style.css بعنوان قالب صفحه انتخاب‌شده و سپس در خط 012 کلاس تگ body با مقدار nomargin تنظیم می‌گردد. توسط کلاس‌ها در CSS می‌توان نحوه نمایش عناصر دلخواه را تنظیم‌نمود. برای مثال، اگر کد زیر را درون فایل style.css بنویسیم:

```
.nomargin {
    margin: 0px;
}
```

هر عنصری که خاصیت class آنرا با nomargin مقداردهی کنیم (در اینجا تگ body) از خاصیت مذکور (margin: 0px;) تبعیت خواهد کرد که در اینجا موجب حذف حاشیه‌های خالی اطراف آن می‌شود. درمورد کلاس‌های CSS در انتهای این آموزش و زمانی که به توضیح فایل style.css می‌پردازیم، توضیحات بیشتری خواهیم‌داد.

در ادامه، یک لینک توسط تگ a ساخته می‌شود که با کلیک‌کردن روی آن، صفحه اصلی سایت در قسمت top_ پنجره (کل صفحه) باز خواهد شد. دقت‌کنید که اگر در فایل top.php خاصیت target لینک مذکور را برروی top_ تنظیم نکنیم، با کلیک‌کردن روی لینک، کل صفحه در قسمت بالای سایت باز می‌شود و موجب می‌گردد که قاب بالا مجدداً قاب‌بندی شود و به سه قسمت بالا و چپ و راست تقسیم‌گردد که قطعاً موردنظر ما نخواهد بود. درنتیجه، از آنجا که می‌خواهیم صفحه اصلی سایت، به‌نحوی باز شود که تمام صفحه را بپوشاند، باید حتماً خاصیت target تگ a را با top_ مقداردهی کنیم. محتوای لینک نیز تصویر title.png است که به‌کمک خاصیت style توسط CSS ارتفاع و پهنای آن به‌اندازه ارتفاع و پهنای کل قاب تنظیم خواهد شد.

صفحه لینک‌ها links.php

این صفحه که در سمت راست سایت ظاهر خواهد شد، وظیفه نمایش منوی اصلی سایت را برعهده دارد و کد آن نیز بسیار ساده است. در این فایل، هیچ کار خاصی به‌جز نمایش چند گزینه به کاربر انجام نمی‌شود:

```php
<?php
    //Copyright محمد مصطفی شهرکی @ http://www.ncis.ir
    require_once 'config.php';
?>
<!doctype html>
<html dir="rtl">
<head>
<meta charset="utf-8"/>
<title><?PHP echo TITLE; ?></title>
<link href="style.css" rel="stylesheet" type="text/css"/>
<base target="mainFrame"/>
</head>
<body>
<ul class="link">
<li><a href="<?PHP echo URL; ?>/main.php">صفحه اصلی</a></li>
<li><a href="<?PHP echo URL; ?>/gallery.php?height=100&width=100&items=4" target="_blank">
آلبوم تصاویر</a></li>
<li><a href="<?PHP echo URL; ?>/contact.php">ارتباط با ما</a></li>
<li><a href="<?PHP echo URL; ?>/about.php">درباره ما</a></li>
</ul>
</body>
</html>
```

تغییرات کد نسبت به جلسه قبل:

- تبدیل اسامی تگ‌ها به حروف کوچک
- مرتب‌شدن خاصیت‌های تگ‌ها به‌ترتیب الفبا
- حذف پرانتزهای دستور require_once در خط 003
- اضافه‌شدن تگ doctype در خط 005
- کوتاه‌شدن تگ meta در خط 008
- جایگزینی متغیرهای $title و $url با ثابت‌های TITLE و URL در خط 009 و خطوط 015 تا 018
- حذف تگ‌های p که به‌جهت ایجاد فاصله مناسب بین گزینه‌های منو به‌کار رفته‌بودند و تنظیم فاصله خطوط توسط CSS برای کلاس link در فایل style.css

توضیح کد:

ابتدا در خط 011 توسط تگ base خاصیت target پیش‌فرض تمامی لینک‌ها با مقدار mainFrame تنظیم می‌شود. بدین‌ترتیب، اگر برای یک لینک در این صفحه، صراحتاً خاصیت target را ذکر نکنیم، از آنجا که در صفحه index.php خاصیت name قاب سمت چپ را mainFrame تعیین کرده بودیم، صفحه مقصد آن لینک در قاب سمت چپ صفحه بارگذاری خواهد شد. در ادامه، چهار لینک به صفحه اصلی، آلبوم تصاویر، صفحه ارتباط با ما و همچنین صفحه تماس با ما به کاربر نشان‌داده خواهد شد که به‌جز آلبوم تصاویر که در یک پنجره جدید باز خواهد شد، سه لینک دیگر در قاب سمت چپ بارگذاری می‌شوند (زیرا خاصیت target را ذکر نکرده‌اند و درنتیجه از خاصیت target تگ base پیروی خواهند نمود). ضمناً اگر به‌دقت لینک آلبوم تصاویر را بررسی‌کنید، متوجه خواهید شد که این لینک، سه مقدار height، width و items را به‌ترتیب با مقادیر ۱۰۰، ۱۰۰ و ٤ ازطریق آدرس (متد Get) برای صفحه gallery.php ارسال می‌کند تا بعداً در آن‌صفحه از این مقادیر بهره‌گیری لازم را داشته‌باشیم.

صفحه اصلی سمت چپ main.php

این فایل، وظیفه اصلی نمایش چکیده مقالات، تعداد بازدید، تعداد نظرات و لینک دریافت فایل مربوط به هر مقاله را برعهده دارد. بعلاوه اگر یک مقاله، حاوی ادامه مطلب باشد، این مطلب را به اطلاع کاربر می‌رساند تا کاربر بداند با کلیک روی عنوان مقاله، اطلاعات بیشتری کسب خواهد نمود. کد این فایل بصورت زیر است:

```php
<?php
    //Copyright محمد مصطفی شهرکی @ http://www.ncis.ir
    require_once 'config.php';
?>
<!doctype html>
<html dir="rtl">
<head>
<meta charset="utf-8"/>
<title><?php echo TITLE; ?></title>
<link href="style.css" rel="stylesheet" type="text/css"/>
</head>
<body style="background-image: url(robot.jpg); background-repeat: no-repeat;">
<?php
    require_once 'db.php';
    $articles = new Articles();
    $result = $articles->SelectNormal();
    if($result !== false && mysql_num_rows($result) > 0) {
        echo '<table border="0px" cellpadding="2px" cellspacing="0px" width="100%">'."\n";
        while($row = mysql_fetch_assoc($result)) {
            echo '<tr align="right" valign="top">';
            echo '<th class="title">';
            echo '<a href="article.php?id='.$row['id'].'">'.$row['title'].'</a>';
            echo ' <span class="visits">( بازدید '.$row['visits'].' بار )';
            echo ' ( نظر '.$articles->CommentsCount($row['id']).' عدد )</span>';
```

```
025.                if($row['body'] != '') {
026.                    echo ' <span class="continued">(حاوي ادامه مطلب)</span>';
027.                }
028.                echo '</th></tr>'."\n";
029.                echo '<tr align="right" valign="top"><td>'.nl2br($row['abstract']).'</td></tr>'."\n";
030.                if($row['filename'] != '') {
031.                    echo '<tr align="right" valign="top">';
032.                    echo '<td><a href="'.URL.'/files/'.$row['filename'].'" target="_blank">
                            دريافت فايل</a></td>';
033.                    echo '</tr>'."\n";
034.                }
035.                echo '<tr align="right" valign="top"><td><hr/></td></tr>'."\n";
036.            }
037.            echo '</table>'."\n";
038.        }
039. ?>
040. </body>
041. </html>
```

تغییرات کد نسبت به جلسه قبل:

- تبدیل اسامی تگ‌ها به حروف کوچک
- مرتب‌شدن خاصیت‌های تگ‌ها به‌ترتیب الفبا
- حذف پرانتزهای دستور require_once در خط 003
- اضافه‌شدن تگ doctype در خط 005
- کوتاه‌شدن تگ meta در خط 008
- جایگزینی متغیر $title با ثابت TITLE در خط 009
- تغییر نحوه تعیین تصویر پس‌زمینه تگ body در خط 012 به ساختار CSS توسط تگ style جهت رعایت استانداردهای جدید
- جایگزینی متغیر $url با ثابت URL در خط 032

توضیح کد:

ابتدا در خط 014 فایل db.php که وظیفه برقراری ارتباط با پایگاه داده‌ها را برعهده دارد، ضمیمه می‌شود. بدین‌ترتیب، می‌توانیم از کدها و کلاس‌های نوشته‌شده درون آن‌فایل استفاده‌کنیم. در خط 015 یک شئ به‌نام $articles از کلاس Articles ایجاد می‌شود. به نحوه ایجاد یک شئ دقت‌کنید: کلمه کلیدی new موجب می‌شود که سازنده کلاس فراخوانی‌شود. سپس، در خط 016 مقالاتی که دارای وضعیت عادی (حذف‌نشده) هستند، توسط تابع SelectNormal شئ $articles استخراج‌شده و در متغیر $result قرار می‌گیرند. در خط 017 متغیر $result بررسی می‌شود و اگر مخالف false بوده (دستور MySQL اجرا شده توسط تابع SelectNormal دارای خطای گرامری نباشد) و همچنین تعداد رکوردهای موجود در متغیر مذکور بیشتر از صفر باشد (مقاله‌ای برای نمایش داشته‌باشیم)، در خط 018 یک جدول با ضخامت حاشیه صفر پیکسل (بدون‌حاشیه) و فاصله ۲ پیکسل بین لبه خانه‌ها تا محتوای آنها و فاصله صفر پیکسل بین خانه‌های مجاور و نهایتاً پهنای ۱۰۰درصد ایجاد می‌کند. در خط 019 یک حلقه تکرار تشکیل می‌شود که تا زمانی که رکوردی برای استخراج وجود داشته‌باشد، کارهای زیر را انجام می‌دهد:

در خط 020 یک سطر جدید با تراز افقی راست و تراز عمودی بالا به جدول اضافه می‌کند. در خط 021 یک خانه با کلاس title به سطر فوق می‌افزاید (در انتهای این جلسه، توسط CSS در فایل style.css نحوه نمایش کلاس مذکور را تعیین خواهیم کرد). در خط 022 یک لینک به صفحه article.php ایجاد می‌شود که شماره مقاله (فیلد id) را ازطریق آدرس (روش Get) برای صفحه مذکور می‌فرستد و عنوان مقاله، متن لینک را تشکیل خواهد داد. در ادامه، در خط 023 یک تگ span با کلاس visits اضافه خواهد شد که تعداد بازدید مقاله مربوطه را نمایش می‌دهد. در خط 024 تعداد نظرات مقاله موردنظر با فراخوانی تابع CommentCount شئ $articles و ارسال شماره id مقاله بعنوان پارامتر برای آن، استخراج‌شده و به‌نمایش در آمده و نهایتاً تگ span بسته می‌شود. در خطوط 025 تا 027 درصورتی‌که مقاله مذکور دارای بدنه باشد (فیلد body خالی نباشد)، یک span با کلاس continued ایجاد، عبارت «حاوی ادامه مطلب» در آن ذکر و تگ span بسته می‌شود. در خط 028 تگ‌های th و tr بسته می‌شوند. سپس در سطر 029 یک سطر دیگر با تراز افقی راست و تراز عمودی بالا اضافه می‌شود و چکیده مقاله بعد از ارسال بعنوان پارامتر برای تابع nl2br (از توابع داخلی PHP که وظیفه تبدیل کارکتر New Line به تگ br برای نمایش صحیح در HTML را برعهده دارد)، در سطر مذکور درج می‌گردد و این سطر نیز بسته می‌شود. در خطوط 030 تا 034 اگر مقاله موردنظر دارای فایل برای دریافت باشد، یک سطر دیگر که حاوی لینک دریافت فایل مربوطه است، ایجاد خواهد شد و عبارت «دریافت فایل» متن لینک را تشکیل خواهد داد. خط 035 نیز یک سطر دیگر به انتهای مقاله اضافه می‌کند که حاوی یک خط جداکننده افقی (تگ hr) است تا مقالات از یکدیگر تفکیک شوند. در نهایت، بعد از آنکه همه مقالات درج شدند، در سطر 037 جدول ایجادشده (تگ table) بسته می‌شود.

صفحه مشاهده مقاله بطور کامل article.php

درصورتی‌که کاربر برروی نام مقاله در صفحه main.php کلیک‌کند، به صفحه article.php هدایت خواهد شد که در این صفحه، می‌تواند مقاله مربوطه را که شماره id آن ازطریق آدرس (متد Get) ارسال شده‌است، به‌طور کامل (چکیده و بدنه) مشاهده کند. ابتدا کد این فایل را مشاهده کنید:

```
001. <?php
002.     //Copyright محمد مصطفي شهركي @ http://www.ncis.ir
003.     require_once 'config.php';
004. ?>
005. <!doctype html>
```

```
<html dir="rtl">
<head>
<meta charset="utf-8"/>
<title><?php echo TITLE; ?></title>
<link href="style.css" rel="stylesheet" type="text/css"/>
</head>
<body style="background-image: url(robot.jpg); background-repeat: no-repeat;">
<?php
    if(!isset($_GET['id']) || $_GET['id'] == '') {
        header('location: main.php');
        exit();
    }
    require_once 'db.php';
    $articles = new Articles();
    $articles->IncrementVisits($_GET['id']);
    $result = $articles->SelectRow($_GET['id']);
    $row = mysql_fetch_assoc($result);
    if($row['status'] == 'normal') {
        echo '<table border="0px" cellpadding="2px" cellspacing="0px" width="100%">'."\n";
        echo '<tr align="right" valign="top"><th class="title">';
        echo $row['title'].' <span class="visits">( بازدید '.$row['visits']
            .' بار )';
        echo ' ( نظر '.$articles->CommentsCount($row['id']).' عدد )</span>';
        echo '</th></tr>'."\n";
        echo '<tr align="right" valign="top"><td>'.nl2br($row['abstract']).'</td></tr>'."\n";
        echo '<tr align="right" valign="top"><td>'.nl2br($row['body']).'</td></tr>'."\n";
        if($row['filename'] != '') {
            echo '<tr align="right" valign="top"><td><a href="'.URL.'/files/'.$row['filename'].'"
                target="_blank">دریافت فایل</a></td></tr>'."\n";
        }
        echo '</table>'."\n";
        echo '<hr/>'."\n";
        echo '<span class="red">نظر دهید:</span><br/>'."\n";
        echo '<form action="comment.php" method="post">'."\n";
        echo '<input name="aid" type="hidden" value="'.$row['id'].'"/>'."\n";
        echo '<table border="0px" cellpadding="2px" cellspacing="0px" width="100%">'."\n";
        echo '<tr align="right" valign="middle">';
        echo '<th width="150px"><label for="name">نام و نام خانوادگي</label></th>';
        echo '<td><input class="transparent" id="name" maxlength="255" name="name" style="width: 100%;"
            type="text"/></td>';
        echo '</tr>'."\n";
        echo '<tr align="right" valign="middle">';
        echo '<th><label for="body">متن نظر</label></th>';
        echo '<td><textarea class="transparent" id="body" name="body" rows="3" style="width: 100%;">
            </textarea></td>';
        echo '</tr>'."\n";
        echo '<tr align="right" valign="middle">';
        echo '<th>وضعیت</th>';
        echo '<td>';
        echo '<input checked="checked" class="transparent" id="stpublic" name="status" type="radio"
            value="public"/> <label for="stpublic">عمومي</label>';
        echo '   ';
        echo '<input class="transparent" id="stprivate" name="status" type="radio" value="private"/>
             <label for="stprivate">خصوصي</label>';
        echo '</td>';
        echo '</tr>'."\n";
        echo '<tr align="center" valign="middle">';
        echo '<td colspan="2"><input style="width: 100%;" type="submit" value="ارسال"/></td>';
        echo '</tr>'."\n";
        echo '</table>'."\n";
        echo '</form>'."\n";
        echo '<hr/>'."\n";
        echo '<span class="red">نظرات:</span><br/>'."\n";
        $comments = new Comments();
        $result = $comments->SelectByArticle($_GET['id']);
        while($row = mysql_fetch_assoc($result)) {
            echo '<b>'.$row['name'].'</b><br/>'."\n";
            if($row['confirmed'] == 0) {
                echo 'در انتظار تأیید توسط مدیر';
            }
            else {
                echo $row['status'] == 'public' ? nl2br($row['body']) : 'خصوصي';
            }
            echo "\n".'<hr/>'."\n";
        }
    }
    else {
        echo 'این مقاله وجود ندارد یا توسط مدیر حذف شده است.<br/>'."\n";
    }
?>
براي بازگشت به صفحه اصلي <b><a href="main.php">اینجا</a></b> کلیک کنید.
</body>
</html>
```

تغییرات کد نسبت به جلسه قبل:

- تبدیل اسامی تگ‌ها به حروف کوچک
- مرتب‌شدن خاصیت‌های تگ‌ها به‌ترتیب الفبا
- حذف پرانتزهای دستور require_once در خطوط 003 و 018
- اضافه‌شدن تگ doctype در خط 005
- کوتاه‌شدن تگ meta در خط 008
- جایگزینی متغیر $title با ثابت TITLE در خط 009
- تغییر نحوه تعیین تصویر پس‌زمینه تگ body در خط 012 به ساختار CSS توسط تگ style جهت رعایت استانداردهای جدید
- اضافه‌شدن شرط (!isset($_GET['id'])) بصورت ترکیب «یا» منطقی با شرط خط 014 برای جلوگیری از نمایش پیغام خطا درصورت عدم ارسال id ازطریق آدرس
- اضافه‌شدن دستور exit در سطر 016 جهت جلوگیری از اجرای مابقی کد درصورت عدم‌اجرای تابع header به هر دلیل
- جایگزینی متغیر $url با ثابت URL در خط 032
- درج تمامی عناصر با کمک دستور echo در PHP به‌جای استفاده مستقیم از HTML جهت افزایش خوانایی کد
- اضافه‌شدن عبارت px به انتهای اعداد معرف اندازه عناصر برحسب پیکسل جهت رعایت استانداردهای HTML
- جایگزینی تگ font با span به‌منظور رعایت استانداردهای جدید HTML
- حذف دستور switch و جایگزینی آن با یک شرط if ساده جهت افزایش سرعت و کوتاه‌شدن کد در خط 071
- اضافه‌شدن قسمت else به شرط if سطر 023 در خطوط 076 تا 078 به‌منظور اعلام عدم‌وجود یا حذف مقاله درصورت اعلام شماره id اشتباه ازطریق آدرس

توضیح کد:

ابتدا در خط 014 ارسال‌شدن پارامتر id ازطریق آدرس (متد Get) بررسی می‌شود و اگر این پارامتر ارسال نشده باشد یا خالی باشد، کاربر به صفحه main.php هدایت می‌شود. سپس، در خط 018 فایل db.php ضمیمه می‌شود و در خط 019 شئ $articles ازروی کلاس Articles ایجاد می‌گردد. در خط 020 تعداد بازدید مقاله جاری با فراخوانی تابع IncrementVisits شئ $articles و ارسال شماره مقاله بعنوان پارامتر برای آن، یک‌واحد افزایش می‌یابد. بعد از این‌کار، در سطر 021 مقاله جاری با کمک تابع SelectRow شئ $articles استخراج می‌شود و تنها رکورد موجود در سطر 022 در متغیر $row قرار می‌گیرد. حال در سطح 023 اگر فیلد status این رکورد برابر با normal باشد، باید مقاله و فرم نظرات کاربران ظاهر شود و درغیر اینصورت (قسمت else)، پیغام عدم‌وجود مقاله یا حذف احتمالی آن توسط مدیر به کاربر نشان‌داده می‌شود. سپس مقاله مربوطه با روشی مشابه فایل main.php به کاربر نمایش داده می‌شود، با این تفاوت که بدنه مقاله نیز در خط 030 به کاربر نشان‌داده خواهد شد. سپس در خطوط 036 تا 060 یک فرم برای دریافت نظر کاربر به وی نشان‌داده می‌شود که مقصد فرم، فایل comment.php و روش ارسال اطلاعات، متد Post است. در خط 038 توسط یک تگ input مخفی (خاصیت type برابر با hidden)، شماره مقاله برای صفحه مذکور ارسال می‌شود. در سطر 042 یک کادر متن برای دریافت «نام و نام خانوادگی» و در سطر 046 یک محدوده متن (Text Area) برای واردکردن متن چندسطری در اختیار کاربر قرار می‌گیرد. ضمناً کاربر می‌تواند وضعیت نظر خود را از بین گزینه‌های عمومی و خصوصی، انتخاب‌کند که این‌کار به‌کمک دو کنترل radio در خطوط 051 و 053 امکان‌پذیر شده‌است. به روش تعریف دو کنترل radio دقت‌کنید: برای آنکه از بین این دو گزینه، فقط یک‌مورد در هر لحظه قابل‌انتخاب باشد، به هر دو کنترل یک نام ("name="status") اختصاص داده‌ایم و درعوض، خاصیت value آنها متفاوت است. بدین‌ترتیب، در صفحه مقصد (در اینجا comment.php) متغیر $_POST['status'] دارای مقداری برابر با value کنترل انتخاب‌شده توسط کاربر خواهد بود. برای مثال، اگر کاربر نوع خصوصی را انتخاب‌کند، مقدار متغیر $_POST['status'] در صفحه مقصد برابر با private (خاصیت value کنترل مربوطه) خواهد بود. با این روش می‌توانیم کنترل‌های radio را به‌راحتی گروه‌بندی کنیم. برای مثال، اگر دو کنترل radio با نام group1 و چهار کنترل radio با نام group2 داشته‌باشیم و به هرکدام، خاصیت value متفاوتی اختصاص‌دهیم، کاربر از بین کنترل‌های group1 در هر لحظه فقط می‌تواند یکی را انتخاب‌کند و همچنین ازبین کنترل‌های group2 نیز در هر لحظه فقط یک‌مورد قابل‌انتخاب خواهد بود؛ اما امکان همزمان یک کنترل از group1 و یک کنترل از group2 برای وی فراهم است، زیرا کنترل‌های انتخاب‌شده، هم‌نام نیستند و لذا به یکدیگر ارتباطی نخواهند داشت. ضمناً خاصیت "checked="checked" در سطر 051 به مرورگر اعلام می‌کند که در زمان نمایش فرم، کنترل «عمومی» به‌طور پیش‌فرض انتخاب شده باشد (اما کاربر درصورت تمایل می‌تواند با انتخاب گزینه «خصوصی» این گزینه را از حالت انتخاب خارج‌کند). در خط 057 نیز یک دکمه جهت ارسال اطلاعات واردشده در اختیار کاربر قرار می‌گیرد (کنترل input با خاصیت type برابر با submit). بقیه سطرهای بین خطوط 036 تا 060 جهت ایجاد یک جدول بدون حاشیه به‌کار می‌روند تا موجب‌شود فرم حالت منظم‌تری داشته‌باشد. در ادامه، یک خط جداکننده افقی (تگ hr) ترسیم می‌شود و سپس، نظرات مربوط به مقاله انتخاب‌شده، ازطریق ایجاد یک شئ به‌نام $comments از کلاس Comments و فراخوانی تابع SelectByArticle و ارسال شماره id مقاله برای آن، استخراج می‌شوند و به‌کمک یک حلقه while، به‌نمایش در می‌آیند. در درون حلقه نیز اگر نظر مربوطه تأیید نشده باشد (فیلد confirmed آن برابر با صفر باشد)، پیغام مناسب به کاربر نشان‌داده می‌شود و در غیر اینصورت، اگر وضعیت پیغام مزبور، عمومی باشد (فیلد status برابر با public باشد)، متن پیغام و در غیر اینصورت عبارت «خصوصی» به کاربر نشان‌داده خواهد شد. در انتهای صفحه نیز توسط خط 080 یک لینک برای بازگشت به صفحه اصلی در اختیار کاربر قرار می‌گیرد.

فایل درج نظر جدید comment.php

این فایل، مقصد فرم موجود در صفحه article.php است و اطلاعات واردشده توسط کاربر بعد از کلیک‌کردن روی دکمه «ارسال»، با روش POST برای این صفحه فرستاده می‌شوند و کاربر نیز به این صفحه هدایت خواهد شد. بنابراین، کلیه پردازش‌های لازم برای ثبت نظر مرتبط با یک مقاله را در این فایل انجام خواهیم داد. کد فایل:

```
001. <?php
002.     //Copyright محمد مصطفي شهركي @ http://www.ncis.ir
003.     require_once 'config.php';
004. ?>
005. <!doctype html>
006. <html dir="rtl">
007. <head>
008. <meta charset="utf-8"/>
009. <title><?php echo TITLE; ?></title>
010. <link rel="stylesheet" type="text/css" href="style.css"/>
011. </head>
012. <body style="background-image: url(robot.jpg); background-repeat: no-repeat;">
013. <?php
014.     $flag = true;
015.     $vars = array('aid', 'name', 'body', 'status');
016.     foreach($vars as $var) {
017.         if(!isset($_POST[$var]) || $_POST[$var] == '') {
018.             $flag = false;
019.         }
020.     }
021.     if($flag) {
022.         require_once 'db.php';
023.         $comments = new Comments();
024.         $result = $comments->Insert($_POST['aid'], $_POST['name'], $_POST['body'], $_POST['status']);
025.         echo ($result > 0 ? 'نظر شما ثبت شده و در انتظار تأیید توسط مدیر است.' :
                'به دلیل بروز خطا، نظر شما ثبت نشد. لطفاً مجدداً تلاش کنید.').'<br/>'."\n";
026.     }
027.     else {
028.         echo 'اطلاعات ناقص است. لطفاً مجدداً فرم را تکمیل کنید.<br/>'."\n";
029.     }
030. ?>
031. کلیک کنید. برای بازگشت به صفحه اصلی <b><a href="main.php">اینجا</a></b>
032. </body>
033. </html>
```

تغییرات کد نسبت به جلسه قبل:

- تبدیل اسامی تگ‌ها به حروف کوچک
- مرتب‌شدن خاصیت‌های تگ‌ها به‌ترتیب الفبا
- حذف پرانتزهای دستور require_once در خطوط 003 و 018
- اضافه‌شدن تگ doctype در خط 005
- کوتاه‌شدن تگ meta در خط 008
- جایگزینی متغیر $title با ثابت TITLE در خط 009
- تغییر نحوه تعیین تصویر پس‌زمینه تگ body در خط 012 به ساختار CSS توسط تگ style جهت رعایت استانداردهای جدید
- اضافه‌شدن دستورات 014 تا 020 و شرط دستور 021 جهت کنترل ارسال‌شدن تمامی مقادیر و خالی‌نبودن هیچ‌کدام از موارد مربوطه جهت ثبت در پایگاه داده‌ها
- حذف دستور import_request_variables جهت رعایت مستندات زبان PHP و استفاده از آرایه $_POST برای دسترسی به مقادیر ارسال‌شده برای صفحه
- استفاده از عناصر آرایه $_POST در خط 024 به‌جای متغیرهایی که با استفاده از دستور import_request_variables تعریف می‌شدند (به‌دلیل حذف دستور)
- نمایش پیغام خطای مناسب برای کاربر به‌جای خروجی تابع mysql_error در خط 025 به‌دلیل امنیت بیشتر و عدم اطلاع احتمالی کاربر از ساختار پایگاه داده‌ها

توضیح کد:

خوشبختانه کد فوق بسیار راحت و روان است و نیاز به توضیح چندانی ندارد. با این حال، ذکر نکات زیر خالی از لطف نیست:

در خط 014 یک متغیر به‌نام $flag تعریف شده‌است که مقدار اولیه آن، true است و وظیفه کنترل‌کردن صحت ارسال تمامی اطلاعات لازم برای درج در پایگاه داده‌ها را برعهده دارد. سپس، در خط 015 اسامی عناصری که به روش POST باید برای این صفحه ارسال شده باشند، در یک آرایه به‌نام $vars قرار می‌گیرد. در ادامه در خطوط 016 تا 020 توسط یک حلقه foreach، همه این عناصر بررسی می‌شوند و هرکدام که ارسال نشده‌باشد یا ارسال‌شده ولی خالی باشد، متغیر $flag را به مقدار false تغییر می‌دهد. بدین ترتیب، در خط 021 با بررسی متغیر $flag می‌توانیم مطمئن‌شویم که همه مقادیر برای صفحه به‌روش Post ارسال شده‌اند و درنتیجه با خیال راحت می‌توانیم از آنها استفاده‌کنیم؛ درغیر اینصورت، در خطوط 027 تا 029 (قسمت else) پیغام کامل‌نبودن اطلاعات به کاربر نشان‌داده خواهد شد. اما اگر اطلاعات کامل باشد، ابتدا در خط 022 فایل db.php ضمیمه می‌شود تا بتوانیم از کلاس‌های تعریف‌شده در آن استفاده‌کنیم. سپس در خط 023 یک شئ به‌نام $comments ازروی کلاس Comments ایجاد می‌گردد. در خط 024 با فراخوانی تابع Insert از شئ مذکور و ارسال شماره مقاله مربوط به نظر، نام فرد نظردهنده، متن نظر و وضعیت آن (عمومی/خصوصی)، اقدام به ثبت آن در پایگاه داده‌ها می‌نماییم. حال در خط 025، اگر این اقدام موفقیت‌آمیز باشد، پیغام ثبت موفقیت‌آمیز نظر و انتظار آن جهت تأیید توسط مدیر و درغیر اینصورت، پیغام عدم ثبت موفقیت‌آمیز نظر را به کاربر نشان می‌دهیم. در انتها نیز (خط 031) یک لینک برای بازگشت به صفحه اصلی سایت در اختیار کاربر قرار می‌دهیم.

صفحه گالری تصاویر gallery.php

این صفحه، بطور خودکار تصاویر موجود در پوشه images را که با پسوند jpg. ذخیره شده‌اند، استخراج‌کرده و یک نسخه پیش‌نمایش (thumbnail) از آنها در پوشه thumbs ایجاد می‌کند که ابعاد تصاویر پیش‌نمایش باید توسط دو پارامتر width و height ازطریق آدرس (روش Get) برای صفحه ارسال شده باشد. سپس تصاویر پیش‌نمایش ساخته‌شده را به‌صورت لینک‌هایی در قالب یک جدول به کاربر نشان می‌دهد که کاربر با کلیک برروی هر تصویر پیش‌نمایش (لینک)، می‌تواند تصویر اصلی را در پنجره جدید، در اندازه‌های واقعی مشاهده‌کند. تعداد تصاویر پیش‌نمایش در هر سطر از جدول نیز باید توسط پارامتر items ازطریق آدرس (روش Get) برای صفحه ارسال شده باشد. ابتدا به کد فایل توجه کنید:

```
001. <?php
002.     //Copyright محمد مصطفی شهرکی @ http://www.ncis.ir
003.     require_once 'config.php';
004. ?>
005. <!doctype html>
006. <html dir="rtl">
007. <head>
008. <meta charset="utf-8"/>
009. <title><?php echo TITLE; ?></title>
010. <link href="style.css" rel="stylesheet" type="text/css"/>
011. <base target="_blank"/>
012. </head>
013. <body style="background-image: url(robot.jpg); background-repeat: no-repeat;">
014. <?php
015.     $flag = true;
016.     $vars = array('height', 'width', 'items');
017.     foreach($vars as $var) {
018.         if(!isset($_GET[$var]) || $_GET[$var] == '' || !is_numeric($_GET[$var])) {
019.             $flag = false;
020.         }
021.     }
022.     if(!$flag) {
023.         header('location: main.php');
024.         exit();
025.     }
026.     $thumbs = scandir('thumbs');
027.     foreach($thumbs as $thumb) {
028.         $pi = pathinfo('thumbs/'. $thumb);
029.         if(isset($pi['extension']) && strtolower($pi['extension']) == 'jpg') {
030.             unlink('thumbs/'.$thumb);
031.         }
032.     }
033.     $images = scandir('images');
034.     foreach($images as $image) {
035.         $pi = pathinfo('images/'.$image);
036.         if(isset($pi['extension']) && strtolower($pi['extension']) == 'jpg') {
037.             $src = imagecreatefromjpeg('images/'.$image);
038.             $dst = imagecreatetruecolor($_GET['width'], $_GET['height']);
039.             imagecopyresized($dst, $src, 0, 0, 0, 0, $_GET['width'], $_GET['height'],
                                 imagesx($src), imagesy($src));
040.             imagejpeg($dst, 'thumbs/'.$image, 100);
041.             imagedestroy($src);
042.             imagedestroy($dst);
043.         }
044.     }
045.     echo '<table border="0px" width="100%">'."\n";
046.     $thumbs = scandir('thumbs');
047.     $count = 0;
048.     foreach($thumbs as $thumb) {
049.         $pi = pathinfo('thumbs/'.$thumb);
050.         if(isset($pi['extension']) && strtolower($pi['extension']) == 'jpg') {
051.             if($count == 0) {
052.                 echo '<tr align="center" valign="middle">'."\n";
053.             }
054.             echo '<td>';
055.             echo '<a href="'.URL.'/images/'.$thumb.'">';
056.             echo '<img src="'.URL.'/thumbs/'.$thumb.'"/>';
057.             echo '</a>';
058.             echo '</td>'."\n";
059.             $count++;
060.             if($count == $_GET['items']) {
061.                 echo '</tr>'."\n";
062.                 $count = 0;
063.             }
064.         }
065.     }
```

```
066.     if($count != 0) {
067.         while($count < $_GET['items']) {
068.             echo '<td> </td>'."\n";
069.             $count++;
070.         }
071.         echo '</tr>'."\n";
072.     }
073.     echo '</table>'."\n";
074. ?>
075. </body>
076. </html>
```

تغییرات کد نسبت به جلسه قبل:

- حذف دستورات ob_start و ob_end_flush از ابتدا و انتهای کد به‌دلیل عدم ضرورت
- تبدیل اسامی تگ‌ها به حروف کوچک
- مرتب‌شدن خاصیت‌های تگ‌ها به‌ترتیب الفبا
- حذف پرانتزهای دستور require_once در خط 003
- اضافه‌شدن تگ doctype در خط 005
- کوتاه‌شدن تگ meta در خط 008
- جایگزینی متغیر $title با ثابت TITLE در خط 009
- تغییر نحوه تعیین تصویر پس‌زمینه تگ body در خط 013 به ساختار CSS توسط تگ style جهت رعایت استانداردهای جدید
- تغییر روش هدایت کاربر به صفحه main.php درصورت عدم ارائه صحیح پارامترهای width، height و items ازطریق آدرس در خطوط 015 تا 025
- استفاده از گیومه تک در پسوند فایل‌ها در خطوط 029 و 036 و 050 به‌جای گیومه جفت جهت افزایش سرعت پردازش کد
- حذف دستور import_request_variables و استفاده مستقیم از عناصر آرایه $_GET به‌منظور رعایت استانداردهای جدید PHP
- تغییر روش خواندن فایل‌های یک پوشه و استفاده از تابع scandir به‌جای بازکردن پوشه و استخراج تک‌تک فایل‌ها با حلقه foreach به‌منظور افزایش سرعت
- حذف شناسه __DIR__ در دسترسی به فایل‌ها و پوشه‌ها به‌دلیل عدم ضرورت
- اضافه‌شدن عبارت px به خاصیت border جدول در خط 045 جهت رعایت استانداردهای HTML در تعیین اندازه عناصر برحسب پیکسل
- جایگزینی متغیر $url با ثابت URL در خطوط 055 و 056
- اضافه‌شدن شرط خطوط 066 تا 072 جهت تکمیل و بستن سطر آخر درصورت عدم تکمیل

توضیح کد:

قبل از هرگونه توضیح درباره نحوه کار این کد، توجه شما را به نکته بسیار مهم زیر جلب می‌کنیم:

برای تولید تصاویر پیش‌نمایش در این کد از کتابخانه GD در PHP استفاده شده است که در اکثریت قریب به اتفاق سرورهای وب بطور پیش‌فرض فعال است؛ با این حال، درصورت بروز خطا در اجرای این کد، در کامپیوتر شخصی خود مسیر زیر را طی کنید:

درصورت استفاده از Wamp Server به ترتیب از چپ به راست روی گزینه‌های زیر کلیک کنید:

```
Wamp Server Icon -> PHP -> PHP Extensions -> php_gd2
```

تا در کنار آن، یک علامت تیک ✓ ظاهر شود (اگر از قبل علامت خورده است، به آن دست نزنید زیرا کد به‌درستی کار خواهد کرد).

درصورت استفاده از سرورهای دیگر، به مسیر نصب PHP رفته و فایل php.ini را پیدا کنید و خط زیر را درون آن بیابید:

```
;extension=php_gd2.dll
```

و سمی‌کالن (;) ابتدای سطر را حذف کنید. حال فایل را ذخیره کرده و Apache را مجدداً راه‌اندازی کنید (اگر نحوه این‌کار را نمی‌دانید، کافی‌است کامپیوتر خود را مجدداً راه‌اندازی کنید).

درصورتی‌که این کتابخانه در سرور وب واقعی (Host) شما فعال نیست، برای فعال‌کردن آن با پشتیبانی سرور تماس بگیرید و از آنها درخواست کنید (به هیچ‌عنوان شخصاً اقدام به دستکاری تنظیمات سرور وب واقعی نکنید زیرا ممکن‌است به‌طور کلی وب‌سایت شما غیرفعال شود و بخش پشتیبانی نیز دیگر به شما خدمات نخواهد داد.

حال با توجه به مطالب فوق، به‌سراغ توضیح کد می‌رویم.

در خط 011 توسط تگ base، خاصیت target لینک‌ها برروی _blank (بازشدن در پنجره جدید) تنظیم می‌شود. بدین‌ترتیب اگر لینکی در صفحه داشته‌باشیم و خاصیت target آنرا صراحتاً ذکر نکنیم، لینک مزبور در پنجره جدید باز خواهد شد. در خط 015 متغیر $flag با مقدار اولیه true تعریف می‌شود. سپس در خط 016 مقادیر width، height و items که باید به‌روش GET برای صفحه ارسال شوند، درون یک آرایه قرار می‌گیرند و به‌کمک یک حلقه foreach در خطوط 017 تا 021 این عناصر از آرایه $_GET بررسی می‌شوند و اگر هرکدام از عناصر فوق‌الذکر ارسال نشده‌باشند یا ارسال‌شده و خالی باشند و یا اینکه دارای مقداری غیرعددی باشند، مقدار متغیر $flag به false تغییر خواهد کرد. بدین‌ترتیب، توسط شرط تعریف‌شده در خطوط 022 تا 025 با بررسی متغیر $flag، می‌توانیم کاربر را درصورت false بودن آن، به صفحه main.php هدایت‌کنیم. دستور exit موجود در خط 024 نیز برای اطمینان بیشتر ذکر شده‌است تا اگر به هر دلیل دستور header موجود در سطر 023 اجرا نشد و

نتوانستیم کاربر را به صفحه مذکور منتقل‌کنیم، بقیه کد اجرا نشود. در خط 026، فهرست فایل‌های موجود در پوشه thumbs توسط دستور scandir خوانده‌شده و بصورت یک آرایه در متغیر $thumbs قرار می‌گیرد. سپس توسط حلقه foreach موجود در خط 027، تمامی عناصر این آرایه پردازش می‌شوند. در خط 028 به‌کمک تابع pathinfo، اطلاعات اولیه درباره عنصر حلقه (نام فایلی که از آرایه $thumbs استخراج شده و نام آن در بدنه حلقه $thumb است) شامل نام فایل، پسوند، مسیر و... بدست می‌آید و بصورت یک آرایه در متغیر $pi ذخیره می‌شود. از آنجا که آرایه $thumbs حاوی پوشه‌ها (مثل . و .. و پوشه‌های فرعی داخل پوشه thumbs) می‌باشد، ابتدا بررسی می‌کنیم که آیا پسوندی برای آن فایل وجود دارد یا نه و اگر پسوند وجود داشت و برابر با jpg بود، آنرا توسط دستور unlink خط 030 حذف می‌کنیم. علت این کار، به‌روز بودن تصاویر پیش‌نمایش است، به‌نحوی که هربار کاربر اقدام به بازدید از صفحه آلبوم تصاویر نماید، فایل‌های پیش‌نمایش قبلی حذف می‌شوند تا مجدداً تصاویر پیش‌نمایش ازروی تصاویر اصلی تولید گردند.

حال در خط 033 فهرست فایل‌های موجود در پوشه images (که تصاویر اصلی در آن قرار دارند) بصورت یک آرایه در متغیر $images قرار می‌گیرد. در حلقه foreach خطوط 034 تا 044 مشابه حلقه قبل، ابتدا پسوند فایل بررسی می‌شود و اگر jpg بود، در خط 037 توسط دستور imagecreatefromjpeg، تصویر مربوطه (اصلی) خوانده‌شده و در متغیر $src ذخیره می‌شود. در خط 038 توسط دستور imagecreatetruecolor یک تصویر True Color (۳۲ بیت) با ابعاد مشخص‌شده توسط پارامترهای width و height ایجاد می‌شود و در متغیر $dst قرار می‌گیرد. در خط 039 توسط دستور imagecopyresized، تصویر موجود در متغیر $src تغییراندازه داده‌شده و در متغیر $dst قرار می‌گیرد. پارامترهای این دستور به‌ترتیب زیر است:

- متغیر مربوط به تصویر مبدأ
- متغیر مربوط به تصویر مقصد
- مؤلفه X نقطه شروع خواندن از تصویر مبدأ
- مؤلفه Y نقطه شروع خواندن از تصویر مبدأ
- مؤلفه X نقطه شروع نوشتن در تصویر مقصد
- مؤلفه Y نقطه شروع نوشتن در تصویر مقصد
- پهنای تصویر مقصد
- ارتفاع تصویر مقصد
- پهنای تصویر مبدأ (توسط تابع imagesx می‌توان پهنای متغیر $src را بدست‌آورد)
- ارتفاع تصویر مبدأ (توسط تابع imagesy می‌توان ارتفاع متغیر $src را بدست‌آورد)

در خط 040 توسط دستور imagejpeg، محتوای متغیر $dst درون فایلی هم‌نام با فایل تصویر اصلی، در پوشه thumbs با کیفیت ۱۰۰٪ ذخیره می‌شود. در خطوط 041 و 042 نیز متغیرهای $src و $dst ازبین می‌روند. دقت‌کنید که برای ازبین‌بردن متغیرهای حاوی اطلاعات تصویر از تابع unset استفاده نمی‌کنیم؛ زیرا دستور مذکورفقط متغیر را ازبین می‌برد ولی دستور imagedestroy، حافظه مربوط‌به اطلاعات تصویر را نیز در RAM آزاد می‌نماید. درنتیجه دستورات فوق و در پایان حلقه foreach اخیر، به‌ازای هر تصویر jpg موجود در پوشه images، یک تصویر پیش‌نمایش با همان نام در پوشه thumbs و با ابعاد مشخص‌شده ایجاد خواهد شد. حال نوبت به نمایش تصاویر می‌رسد. ابتدا در خط 045 یک جدول با ضخامت حاشیه صفر پیکسل (بدون حاشیه) و با پهنای ۱۰۰٪ ایجاد می‌شود. سپس در خط 046 فهرست فایل‌های پیش‌نمایش موجود در پوشه thumbs در متغیر $thumbs قرار می‌گیرد. در خط 047 متغیری به‌نام $count با مقدار اولیه صفر تعریف می‌شود که وظیفه آن، تنظیم تعداد عناصر موجود در هر سطر متناسب‌با پارامتر items است که برای صفحه به‌روش Get ارسال می‌شود. توسط حلقه foreach موجود در خطوط 048 تا 065 اعمال زیر رخ می‌دهد:

ابتدا پسوند فایل در خطوط 049 و 050 بررسی می‌شود و اگر jpg بود، در خط 051 درصورتی‌که متغیر $count برابر با صفر باشد (در ابتدای سطر جدید باشیم)، یک تگ tr برای ایجاد سطر جدید در جدول توسط خط 052 تولید می‌شود. در خط 054 یک تگ td برای نگهداری تصویر ایجاد می‌شود. در خط 055 یک لینک که به تصویر اصلی (موجود در پوشه images) اشاره می‌کند، تولید می‌شود (از آنجا که فایل پیش‌نمایش هم‌نام با فایل تصویر تولید شده است، می‌توانیم از همان نام پیش‌نمایش برای اشاره به تصویر استفاده‌کنیم). در خط 056 تصویر پیش‌نمایش درج می‌شود و در خط 057 و 058 به‌ترتیب تگ‌های a و td بسته می‌شوند. سپس در خط 059 متغیر $count یک‌واحد اضافه می‌شود. حال اگر بعد از افزایش این متغیر، اگر مقدار آن با پارامتر items برابر شد (یعنی در سطر جاری از جدول، به‌تعداد مشخص‌شده، تصویر درج شده‌است و به انتهای سطر رسیده‌ایم)، تگ tr بسته می‌شود و متغیر $count مجدداً صفر می‌شود تا در تکرار بعدی حلقه foreach مجدداً یک tr دیگر تولید شود. حال در خط 066 و در انتهای حلقه foreach اگر مقدار $count مخالف صفر باشد، به‌معنای وجود یک سطر تکمیل‌نشده در جدول است. لذا، به‌کمک حلقه while موجود در سطرهای 067 تا 070، به‌اندازه اختلاف $count و پارامتر items، خانه بدون محتوا در سطر آخر درج نموده و سپس، سطر آخر جدول را توسط خط 071 می‌بندیم و نهایتاً در خط 073 نیز تگ table بسته می‌شود.

صفحه ارتباط با ما contact.php

این صفحه، صرفاً یک فرم در اختیار کاربر قرار می‌دهد تا بتواند نام و نام خانوادگی، پست الکترونیکی، موضوع و متن پیام خود را وارد کند تا بدین‌وسیله، با سایت ارتباط برقرار نماید. اطلاعات دریافت‌شده توسط کاربر نیز به‌روش Post برای فایل result.php ارسال خواهد شد. کد این صفحه بسیار ساده و به‌صورت زیر است:

```
001. <?php
002.     //Copyright محمد مصطفی شهرکی @ http://www.ncis.ir
003.     require_once 'config.php';
004. ?>
```

```
005. <!doctype html>
006. <html dir="rtl">
007. <head>
008. <meta charset="utf-8"/>
009. <title><?php echo TITLE; ?></title>
010. <link href="style.css" rel="stylesheet" type="text/css"/>
011. </head>
012. <body style="background-image: url(robot.jpg); background-repeat: no-repeat;">
013. <form action="result.php" method="post">
014. <table border="0px" width="500px"/>
015. <tr align="right" valign="middle">
016. <th width="150px"><label for="name">نام و نام خانوادگي</label></th>
017. <td><input class="transparent" id="name" maxlength="50" name="name" style="width: 100%;" type="text"/></td>
018. </tr>
019. <tr align="right" valign="middle">
020. <th><label for="email">پست الکترونيکي</label></th>
021. <td><input class="transparent" id="email" maxlength="50" name="email" style="width: 100%;"
     type="text"/></td>
022. </tr>
023. <tr align="right" valign="middle">
024. <th><label for="subject">موضوع</label></th>
025. <td><input class="transparent" id="subject" maxlength="50" name="subject" style="width: 100%;"
     type="text"/></td>
026. </tr>
027. <tr align="right" valign="middle">
028. <th><label for="body">متن پيام</label></th>
029. <td><textarea class="transparent" id="body" name="body" rows="5" style="width: 100%"></textarea></td>
030. </tr>
031. <tr align="center" valign="middle">
032. <td colspan="2"><input style="width: 100%;" type="submit" value="ارسال"/></td>
033. </tr>
034. </table>
035. </form>
036. </body>
037. </html>
```

تغییرات کد نسبت به جلسه قبل:

- حذف دستورات ob_start و ob_end_flush از ابتدا و انتهای کد به‌دلیل عدم ضرورت
- تبدیل اسامی تگ‌ها به حروف کوچک
- مرتب‌شدن خاصیت‌های تگ‌ها به‌ترتیب الفبا
- حذف پرانتزهای دستور require_once در خط 003
- اضافه‌شدن تگ doctype در خط 005
- کوتاه‌شدن تگ meta در خط 008
- جایگزینی متغیر $title با ثابت TITLE در خط 009
- تغییر نحوه تعیین تصویر پس‌زمینه تگ body در خط 012 به ساختار CSS توسط تگ style جهت رعایت استانداردهای جدید
- اضافه‌شدن عبارت px به اعداد مربوط‌به اندازه عناصر جهت رعایت استانداردهای HTML در تعیین اندازه عناصر برحسب پیکسل
- حذف خطوط مربوط‌به دریافت وب‌سایت کاربر به‌دلیل عدم استفاده در صفحه مقصد فرم
- حذف خاصیت width تگ‌های th به‌جز سطر اول به‌دلیل عدم ضرورت (در HTML کافی‌است پهنای یکی از خانه‌های یک ستون تعیین‌شود تا ستون به آن اندازه درآید)

توضیح کد:

این فایل، حاوی هیچ‌گونه کد PHP نیست و کد HTML آن نیز بسیار ساده است. درنتیجه نیاز به توضیح خاصی ندارد.

فایل مقصد صفحه ارتباط با ما result.php

این فایل، مقصد فرم موجود در صفحه ارتباط با ما است. وظیفه این صفحه، ارسال اطلاعات واردشده توسط کاربر به ایمیل مدیریت سایت است. کد فایل:

```
001. <?php
002.     //Copyright محمد مصطفي شهركي @ http://www.ncis.ir
003.     require_once 'config.php';
004. ?>
005. <!doctype html>
006. <html dir="rtl">
007. <head>
008. <meta charset="utf-8"/>
009. <title><?php echo TITLE; ?></title>
010. <link href="style.css" rel="stylesheet" type="text/css"/>
011. </head>
012. <body style="background-image: url(robot.jpg); background-repeat: no-repeat;">
013. <?php
014.     $flag = true;
```

```
015.     $vars = array('name', 'email', 'subject', 'body');
016.     foreach($vars as $var) {
017.         if(!isset($_POST[$var]) || $_POST[$var] == '') {
018.             $flag = false;
019.         }
020.     }
021.     if($flag && isset($_SERVER['HTTP_REFERER']) && $_SERVER['HTTP_REFERER'] == URL.'/contact.php') {
022.         $to = 'mmshfe@gmail.com';
023.         $headers = '';
024.         $headers .= 'From: '.$_POST['email']."\r\n";
025.         $headers .= 'Reply-To: '.$_POST['email']."\r\n";
026.         $headers .= 'X-Mailer: PHP/'.phpversion();
027.         $sent = @mail($to, $_POST['subject'], $_POST['body'], $headers);
028.         echo $sent ? 'پیام با موفقیت ارسال شد.' : 'پیام ارسال نشد. لطفاً دوباره تلاش کنید.';
029.     }
030. ?>
031. <br/>
032. <a href="<?php echo URL; ?>/index.php" target="_top">بازگشت به صفحه اصلی</a><br/>
033. </body>
034. </html>
```

تغییرات کد نسبت به جلسه قبل:

- حذف دستورات ob_start و ob_end_flush از ابتدا و انتهای کد به‌دلیل عدم ضرورت
- تبدیل اسامی تگ‌ها به حروف کوچک
- مرتب‌شدن خاصیت‌های تگ‌ها به‌ترتیب الفبا
- حذف پرانتزهای دستور require_once در خط 003
- اضافه‌شدن تگ doctype در خط 005
- کوتاه‌شدن تگ meta در خط 008
- جایگزینی متغیر $title با ثابت TITLE در خط 009
- تغییر نحوه تعیین تصویر پس‌زمینه تگ body در خط 012 به ساختار CSS توسط تگ style جهت رعایت استانداردهای جدید
- اضافه‌شدن خطوط 014 تا 020 جهت کنترل ارسال‌شدن همه موارد موردنیاز (شامل نام و نام خانوادگی، پست الکترونیکی، موضوع و متن پیام) برای صفحه به روش Post
- تکمیل شرط خط 021 به‌منظور جلوگیری از دسترسی مستقیم به این صفحه
- جایگزینی متغیر $url با ثابت URL در خطوط 021 و 032
- اضافه‌شدن متغیر $sent در خط 027 و عملگر @ به ابتدای تابع mail جهت جلوگیری از نمایش پیغام‌های خطای احتمالی به کاربر در جهت حفظ اطلاعات Server
- استفاده از متغیر $sent در خط 028 برای نمایش پیغام مناسب به کاربر (ارسال موفقیت‌آمیز پیام یا عدم ارسال آن و لزوم تلاش مجدد کاربر)

توضیح کد:

بعد از بررسی ارسال تمامی پارامترهای لازم برای صفحه به روش Post در خطوط 014 تا 020، در خط 021 ابتدا متغیر $flag بررسی می‌شود و اگر برابر با true بود وجود متغیر $_SERVER['HTTP_REFERER'] بررسی می‌شود. لازم به توضیح است که آرایه $_SERVER حاوی اطلاعات مهمی درخصوص داده‌های مبادله‌شده بین سرور و کلاینت می‌باشد. یکی از این اطلاعات، صفحه ارجاع‌دهنده به صفحه جاری است که به آن، Referer می‌گوییم. برای مثال، اگر در صفحه A لینکی به صفحه B داشته باشیم و با کلیک روی آن لینک، به صفحه مربوطه هدایت‌شویم، Referer حاوی آدرس صفحه A خواهد بود؛ اما اگر مستقیماً در مرورگر آدرس صفحه B را تایپ‌کنیم، دیگر Referer وجود نخواهد داشت. در سطر 021، ابتدا بعد از بررسی $flag، وجود Referer را بررسی می‌کنیم (کاربر نباید مستقیماً وارد این صفحه شده باشد) و بعد از اینکه مطمئن‌شدیم Referer وجود دارد، آنرا با آدرس صفحه ارتباط با ما مقایسه می‌کنیم (کاربر باید ازطریق فرم موجود در سایت ما وارد این صفحه شده باشد). این کنترل، هرچند ۱۰۰٪ قابل‌اعتماد نیست اما جلوی بسیاری از نفوذهای خرابکارانه را می‌گیرد. برای مثال، اگر فرمی مشابه در سایتی دیگر وجود داشته باشد که آدرس صفحه مقصد آن، همین صفحه از سایت ما باشد، این صفحه اطلاعات را از آن نمی‌پذیرد و ایمیل‌های درخواست‌شده توسط آن صفحه را نمی‌فرستد. درحقیقت، بطور خلاصه می‌توان گفت با کمک متغیر $_SERVER['HTTP_REFERER'] می‌توانیم بفهمیم کاربران ازطریق چه صفحه‌ای وارد صفحه جاری شده‌اند. از این اطلاعات می‌توان استفاده‌های زیادی نمود (مثلاً قبول‌کردن اطلاعات ورودی تنها از یک یا چند منبع مشخص، بدست‌آوردن فهرست سایت‌هایی که به ما لینک داده‌اند و...). حال اگر تمامی شروط مشخص‌شده برقرار باشند، درخط 022 آدرس ایمیل مقصد مشخص می‌شود که می‌توانید آنرا با آدرس ایمیل خودتان جایگزین‌کنید. سپس Headerهای صفحه که بیانگر اطلاعاتی درخصوص ایمیل درحال‌ارسال هستند تنظیم می‌شوند: در خط 023 متغیر $headers با مقدار اولیه یک رشته خالی ایجاد می‌شود. در خط 024 آدرس ایمیل فرستنده نامه که از کاربر دریافت کرده‌ایم، به Header اضافه می‌شود. در خط 025 آدرس مقصد جواب نامه (ایمیلی که جواب نامه به آن ارسال می‌شود) نیز با آدرس ایمیل دریافت‌شده از کاربر مقداردهی می‌شود. در خط 026 نیز نسخه مورد استفاده PHP بعنوان موتور ارسال‌کننده ایمیل تعیین می‌شود. این مورد معمولاً جهت ارسال صحیح ایمیل ضروری است. نکته قابل‌توجه، آن است که همه این اطلاعات به Header اضافه می‌شوند و با \r\n از هم جدا می‌گردند که معادل کلید Enter صفحه‌کلید است. در خط 027 متغیر $sent با خروجی تابع mail (از توابع داخلی PHP برای ارسال ایمیل) مقداردهی می‌شود. پارامترهای تابع mail به‌ترتیب شامل گیرنده نامه، موضوع، بدنه و اطلاعات Header است و خروجی آن، یکی از مقادیر true یا false است که به‌ترتیب بیانگر ارسال موفقیت‌آمیز ایمیل یا عدم‌ارسال آن می‌باشد. البته بسته به نوع تنظیمات سرور، ممکن‌است این تابع درصورت عدم‌ارسال موفقیت‌آمیز ایمیل،

علت بروز خطا را در خروجی نمایش‌دهد که در اینجا با کمک کارکتر @ قبل از این دستور، جلوی نمایش پیغام‌های خطای آنرا گرفته‌ایم (کارکتر @ در PHP قبل از هر دستور ذکر شود، خطاهای آنرا از دید کاربر پنهان می‌سازد). البته بهتر است به استفاده از این کارکتر عادت نکنید و سعی‌کنید حتی‌الأمکان کدی بنویسید که خطا نداشته‌باشد (بجای آنکه خطاهای آنرا پنهان کنید). علت استفاده از کارکتر @ در اینجا، امنیت نسبتاً بیشتر و اطلاع‌نیافتن کاربر از برخی تنظیمات مهم سرور مثل آدرس سرور ایمیل و شماره پورت و... است که بسته به نوع خطا، احتمال‌دارد در پیغام خطای مربوطه ظاهر شوند. در نهایت در خط 028 متغیر $sent بررسی می‌شود و پیغام مناسب بسته به true یا false بودن آن برای کاربر به‌نمایش در می‌آید. یکی‌از تفاوت‌های کد این فایل با سایر صفحات، نحوه آدرس‌دهی لینک بازگشت به صفحه اصلی است. همان‌طور که ملاحظه می‌کنید در خط 032 لینک بازگشت به صفحه اصلی، به‌جای اشاره به فایل main.php، به فایل اصلی سایت یعنی index.php اشاره می‌کند و خاصیت target آن نیز برروی _top تنظیم شده‌است. علت این مسئله، احتمال واردکردن آدرس این صفحه بطور مستقیم در مرورگر یا ارسال اطلاعات ازطریق فرم‌های سایت‌های دیگر است و از آنجا که چنین مواردی در طراحی این صفحه مدنظر بوده‌است، این صفحه را به‌نحوی تنظیم کرده‌ایم که اگر به هر شکل اجرا شود (بطور تمام‌صفحه یا درون یک قاب و...)، لینک مربوطه صفحه اصلی سایت را به‌نحوی ظاهر سازد که تمام صفحه را بپوشاند. بنابراین، نمی‌توانیم در اینجا از فایل main.php استفاده کنیم چون درصورت استفاده آن، لینک‌های سمت‌راست صفحه و همچنین نشان سایت در بالای صفحه نشان‌داده نخواهد شد. البته از این روش در سایر صفحات نیز می‌توان استفاده‌کرد اما از آنجا که احتمال استفاده از آنها بدین‌صورت کمتر است، به‌اندازه این صفحه درمورد آنها سخت‌گیری نشده‌است (شما می‌توانید درصورت تمایل، لینک «بازگشت به صفحه اصلی» را در سایر صفحات تغییر دهید تا بصورت این لینک درآیند).

صفحه درباره ما about.php

این فایل، کد بسیار ساده‌ای دارد که به‌صورت زیر است:

```
001. <?php
002.     //Copyright محمد مصطفي شهركي @ http://www.ncis.ir
003.     require_once 'config.php';
004. ?>
005. <!doctype html>
006. <html dir="rtl">
007. <head>
008. <meta charset="utf-8"/>
009. <title><?php echo TITLE; ?></title>
010. <link href="style.css" rel="stylesheet" type="text/css"/>
011. <base target="_blank"/>
012. </head>
013. <body style="background-color: url(robot.jpg); background-repeat: no-repeat;">
014. <table border="0px" width="50%">
015. <tr align="right" valign="middle">
016. <th width="250px">طراح و برنامه نويس</th><td>محمد مصطفي شهركي</td>
017. </tr>
018. <tr align="right" valign="middle">
019. <th>وب سايت رسمي مدير سايت</th><td><a href="http://www.ncis.ir">وب سايت علمي تخصصي nCIS</a></td>
020. </tr>
021. <tr align="right" valign="middle">
022. <th>پست الكترونيكي</th><td><a href="mailto:mmshfe@gmail.com">mmshfe@gmail.com</a></td>
023. </tr>
024. <tr align="right" valign="middle">
025. <th>تلفن تماس</th><td><span dir="ltr">+989156309626</span></td>
026. </tr>
027. </table>
028. <a href="<?php echo URL; ?>/index.php" target="_top">بازگشت به صفحه اصلي</a><br/>
029. </body>
030. </html>
```

تغییرات کد نسبت به جلسه قبل:

- حذف دستورات ob_start و ob_end_flush از ابتدا و انتهای کد به‌دلیل عدم ضرورت
- تبدیل اسامی تگ‌ها به حروف کوچک
- مرتب‌شدن خاصیت‌های تگ‌ها به‌ترتیب الفبا
- حذف پرانتزهای دستور require_once در خط 003
- اضافه‌شدن تگ doctype در خط 005
- کوتاه‌شدن تگ meta در خط 008
- جایگزینی متغیر $title با ثابت TITLE در خط 009
- اضافه‌شدن تگ base جهت تعیین محل پیش‌فرض بازشدن لینک‌ها در خط 011
- تغییر نحوه تعیین تصویر پس‌زمینه تگ body در خط 013 به ساختار CSS توسط تگ style جهت رعایت استانداردهای جدید
- اضافه‌شدن عبارت px به اعداد در خطوط 014 و 016 جهت رعایت استانداردهای HTML در تعیین اندازه عناصر برحسب پیکسل
- حذف خاصیت target لینک‌های موجود در خطوط 019 و 022 به‌دلیل استفاده از تگ base

توضیح کد:

این کد تماماً بصورت HTML است و شامل دستور خاص و پیچیده‌ای که نیازمند توضیح باشد نیست. درصورت تمایل، می‌توانید اطلاعات این صفحه را تغییر داده و اطلاعات خودتان را جایگزین آن کنید.

فایل قالب صفحات style.css

این فایل که در صفحات مختلف توسط تگ link بعنوان قالب صفحات مورد استفاده قرار گرفته‌است، با استفاده از ساختار CSS، نحوه نمایش عناصر مختلف را تنظیم می‌کند.

ابتدا کد فایل را ملاحظه کنید:

```
001. /* Copyright محمد مصطفي شهركي @ http://www.ncis.ir */
002. * {
003.     font-family: Tahoma;
004.     text-shadow: 0px 0px 1px #000000;
005. }
006. .continued {
007.     font-size: x-small;
008. }
009. .nomargin {
010.     margin: 0px;
011. }
012. .red {
013.     color: #ff0000;
014.     text-shadow: 0px 0px 1px #ff0000;
015. }
016. .title {
017.     background-color: #ffff00;
018. }
019. .transparent {
020.     background: transparent;
021. }
022. .visits {
023.     color: #afafaf;
024.     font-size: small;
025.     text-shadow: 0px 0px 1px #afafaf;
026. }
027. a {
028.     color: #0000ff;
029.     text-decoration: none;
030.     text-shadow: 0px 0px 1px #0000ff;
031. }
032. a:hover {
033.     color: #ff0000;
034.     text-shadow: 0px 0px 1px #ff0000;
035. }
036. body, th {
037.     font-size: medium;
038. }
039. img {
040.     border: none;
041. }
042. td, input, textarea {
043.     font-size: small;
044. }
045. ul.link {
046.     color: #0000ff;
047.     font-weight: bold;
048.     line-height: 2em;
049.     text-shadow: 0px 0px 1px #0000ff;
050. }
```

تغییرات کد نسبت به جلسه قبل:

- مرتب‌شدن انتخاب‌گرها (Selector) و خصوصیات داخلی آنها به‌ترتیب الفبا
- تغییر نحوه تعیین نوع نمایش تگ a در خطوط 027 تا 035 به‌منظور کوتاه‌تر شدن کد
- اضافه‌شدن خط 047 به انتخاب‌گر تگ ul با کلاس link جهت معادل‌سازی تگ b که از فایل links.php حذف شده است
- اضافه‌شدن خط 048 به انتخاب‌گر تگ ul با کلاس link جهت معادل‌سازی تگ p که از فایل links.php حذف شده است

توضیح کد:

قبل‌از آنکه کد این فایل را توضیح‌دهیم، بهتر است کمی با CSS آشنا شوید. به‌طور کلی در استانداردهای جدید طراحی وب‌سایت، صفحات وب به دو بخش عمده تقسیم می‌شوند: ساختار و نحوه نمایش. ساختار صفحات، مشخص‌کننده استفاده یا عدم‌استفاده از هر عنصر صفحه ازقبیل جدول، پاراگراف، تصویر و... است و نحوه نمایش، تعیین‌کننده روش ظاهرشدن هر عنصر در صفحه می‌باشد (برای مثال، پاراگراف اول به‌رنگ آبی و بقیه پاراگراف‌ها به رنگ قرمز، یا وجود تصویر پس‌زمینه برای بدنه صفحه و...). درچندسال‌قبل

که استانداردها به‌اندازه زمان فعلی مدون نبودند، زبان HTML مجبور بود بخشی از بار تعیین نحوه نمایش عناصر را نیز برعهده بگیرد. به همین دلیل، تگ‌هایی مثل font و... و همچنین خاصیت‌هایی برای عناصر ازقبیل align و... به آن اضافه‌شدند که صرفاً وظیفه کنترل نحوه نمایش را برعهده داشته و هیچ‌گونه ارتباطی به ساختار صفحه ندارند(ساختار صفحه توسط خود تگ‌ها مشخص می‌شود نه اینگونه خاصیت‌های آنها). به‌همین‌منظور، کنسرسیوم وب جهانی یا W3C تصمیم‌گرفت بخش نمایش را از ساختار جدا کند و وظایف آنرا تماماً به CSS محول‌نماید. CSS مختلف Cascading Style Sheet است که به‌معنای برگه قالب آبشاری می‌باشد. علت این نام‌گذاری، ساختار پله‌ای و آبشارمانند تعیین نحوه نمایش صفحات است که در CSS وجود دارد. برای مثال، خاصیت style بالاترین اولویت را دارد ولی فقط برای یک تگ خاص به‌کار می‌رود و برای سایر تگ‌ها درصورت نیاز، باید تکرار شود، بعد از آن تگ style قراردارد که می‌توان از آن برای تعیین نحوه نمایش عناصر یک‌صفحه در یک مکان و بدون نیاز به تکرار استفاده‌نمود و در پایین‌ترین مرتبه اولویت، فایل CSS خارجی است که با کمک تگ link به صفحه معرفی می‌شود و در عوض، امکان استفاده در صفحات مختلف را دارد. به‌دلیل آنکه CSS موضوع اصلی ما در این آموزش نیست، از بیان فنون و ظرایف آن در اینجا خودداری می‌کنیم و توضیحات فوق را نیز صرفاً جهت اطلاعات بیشتر علاقمندان مطرح‌کردیم تا درصورت تمایل، آموزش‌های CSS موجود در سایت http://www.ncis.ir را نیز پیگیری نمایند. بطور کلی نحوه استفاده از فایل‌های خارجی CSS بدین‌ترتیب است که کدهای CSS خود را درون یک فایل می‌نویسیم و آنرا با پسوند css. ذخیره می‌کنیم و سپس، در درون صفحاتی که تمایل داریم از این الگوی نمایش استفاده‌کنند، توسط تگ link (مشابه آنچه که در فایل‌های قبل ملاحظه کردید)، این فایل را بعنوان قالب نمایش صفحه معرفی می‌کنیم. در درون این فایل نیز برای تعیین نحوه نمایش عناصر باید ابتدا عنصر موردنظر را انتخاب‌کنیم که این‌کار با استفاده از مفهومی به‌نام انتخاب‌گر (Selector) انجام می‌شود. یک انتخاب‌گر CSS می‌تواند یکی‌از موارد زیر یا ترکیبی از آنها باشد:

۱) همه عناصر: برای این‌کار از شناسه * استفاده می‌کنیم. این شناسه خاص، به‌معنای کلیه عناصر موجود در صفحه است
۲) یک تگ خاص: برای این‌کار، نام آن تگ را می‌نویسیم (مثل body یا img)
۳) یک کلاس مشخص: ابتدا یک نقطه و بعد، نام کلاس موردنظر را می‌نویسیم (مثل normargin. یا myclass.)
٤) یک شناسه مشخص: ابتدا علامت # و بعد، نام شناسه موردنظر را ذکر می‌کنیم (مثل login# یا myid#)

در ادامه، بعد از انتخاب‌گر موردنظر، یک آکولاد باز ({) می‌نویسیم و بعد از آن، ساختار موردنظرمان را مشخص می‌کنیم و درنهایت، آکولاد را می‌بندیم. برای مثال:

```
* {
    font-family: Tahoma;
    font-size: small;
}
```

موجب می‌شود همه عناصر صفحه از قلم Tahoma با اندازه small استفاده‌کنند. البته امکان تعیین چند انتخاب‌گر نیز وجود دارد:

```
body, p {
    color: #ff0000;
}
```

بدین‌ترتیب، متن تگ‌های p و body در هر جای صفحه که به‌کار روند، با رنگ قرمز به‌نمایش در خواهد آمد.
همچنین می‌توانیم چند انتخاب‌گر را ترکیب‌کنیم:

```
p.inverse {
    background-color: #000000;
    color: #ffffff;
}
```

بدین‌ترتیب، تنها تگ‌های p که خاصیت class="inverse" داشته‌باشند، با زمینه مشکی و رنگ متن سفید ظاهر خواهند شد. یا در مثال زیر:

```
img#logo {
    left: 0px;
    position: absolute;
    top: 0px;
}
```

فقط تگ img که خاصیت id="logo" داشته‌باشد، در گوشه بالا و سمت چپ صفحه ظاهر می‌شود.
البته امکانات CSS صرفاً شامل موارد فوق نیست و می‌توان هرگونه ترکیبی از عناصر را به‌کار برد. برای مثال، کد زیر را درنظر بگیرید:

```
div.article img#icon {
    border: 10px;
}
```

کد فوق موجب می‌شود تا تگ img با خاصیت id="icon" فقط درصورتی‌که داخل یک تگ div با خاصیت class="article" قرار داشته‌باشد، با ضخامت حاشیه‌ای برابر با ۱۰ پیکسل ترسیم‌شود.
همان‌طور که ملاحظه می‌کنید، با کمک انتخاب‌گرهای CSS می‌توانیم هر عنصری را در صفحه انتخاب و ظاهر موردنظر را برای آن تعیین نماییم. هرچند باز هم این موارد شامل تمام امکانات شما درزمینه انتخاب‌گرها نمی‌شود و انواع ترکیب‌های انتخاب‌گر در CSS بسیار بیشتر از موارد ذکرشده است؛ اما از آنجا که موضوع اصلی ما در این آموزش، CSS نیست، به همین حد از توضیحات بسنده می‌کنیم. درمورد نحوه نمایش نیز در بین آکولادهای باز و بسته باید از خواص مشخص‌شده در CSS استفاده‌کنیم که باز هم به‌دلیل عدم محوریت CSS در این آموزش، صرفاً به توضیح خواصی که در فایل style.css به‌کار رفته‌اند اکتفا می‌کنیم و علاقمندان به یادگیری CSS را به مطالعه آموزش‌های مرتبط با این موضوع دعوت می‌نماییم. حال به‌سراغ توضیح کدهای فایل style.css می‌رویم:

- در خطوط 002 تا 005 قلم همه عناصر Tahoma تعیین‌شده و یک سایه مشکی با ضخامت ۱ پیکسل برای آنها تعیین می‌شود (سایه در همه مرورگرهاقابل مشاهده نیست)

- در خطوط 006 تا 008 برای عناصری که خاصیت class="continued" دارند، اندازه قلم x-small (یک‌واحد کوچک‌تر از small) تعیین می‌شود
- در خطوط 009 تا 011 برای عناصر با خاصیت class="nomargin" فاصله حاشیه خارجی (لبه عنصر تا لبه والد) صفر پیکسل تعیین می‌گردد
- در خطوط 012 تا 015 برای عناصر با خاصیت class="red" رنگ متن قرمز و سایه قرمزرنگ با ضخامت ۱ پیکسل مشخص می‌شود
- در خطوط 016 تا 018 برای عناصر با خاصیت class="title" رنگ پس‌زمینه زرد تعیین می‌شود
- خطوط 019 تا 021 اعلام می‌کند که عناصر با خاصیت class="transparent" با پس‌زمینه شفاف ترسیم‌شوند تا زمینه زیر آنها قابل‌رؤیت باشد
- خطوط 022 تا 026 عناصر با خاصیت class="visits" را با رنگ متن و سایه خاکستری روشن و اندازه قلم small و ضخامت سایه ۱ پیکسل ترسیم می‌کند
- خطوط 027 تا 031 مشخص می‌کند که لینک‌ها (تگ a) بدون خط زیر (Underline)، با رنگ متن آبی و سایه آبی به‌ضخامت ۱ پیکسل ترسیم‌شوند
- خطوط 032 تا 035 موجب نمایش لینک‌هایی که با ماوس روی آنها قرار داریم، با رنگ متن و سایه قرمز می‌گردند
- خطوط 036 تا 038 موجب تعیین اندازه قلم تگ‌های body و th با مقدار medium (متوسط، بزرگتر از small) می‌شوند
- توسط خطوط 039 تا 041 ضخامت حاشیه تصاویر صفر پیکسل تعیین می‌شود تا اگر از یک تصویر بعنوان لینک استفاده‌کردیم (مثل آلبوم تصاویر)، حاشیه نداشته‌باشد
- در خطوط 042 تا 044 اندازه قلم تگ‌های td و input و textarea با مقدار small تنظیم می‌شود
- در خطوط 045 تا 050 نیز نهایتاً تگ ul با خاصیت class="link"، به رنگ آبی، با قلم توپُر، ارتفاع خط دوبرابر اندازه معمولی و سایه آبی ترسیم می‌شود

بخش مدیریت

بخش مدیریت سایت شامل تمامی امکانات لازم برای مدیریت مقالات و نظرات موجود در سایت اعم‌از درج مقاله/نظر جدید یا ویرایش موارد موجود می‌باشد. بنابراین، بدلیل اهمیت این‌بخش، باید از آن به‌نحو مناسب محافظت‌شود تا کاربران نتوانند به‌راحتی وارد آن شوند. در این جلسه، فعلاً امنیت مقدماتی را برای این بخش فراهم می‌کنیم که شامل محافظت از دسترسی با کمک نام کاربری و رمز عبور است و در جلسات آینده موضوع امنیت سایت را ارتقاء خواهیم داد. در روش دسترسی با کمک نام کاربری و رمز عبور، کاربران با واردکردن مسیر ورود به بخش مدیریت، با یک فرم ورود (Login) مواجه می‌شوند که باید نام کاربری و رمز عبور را مطابق آنچه قبلاً تعریف شده است، وارد نمایند. در این سایت، نام کاربری و رمز عبور بخش مدیریت به‌ترتیب cmsADMIN و cmsPASSWORD است که نسبت‌به بزرگی و کوچکی حروف حساس است و این اطلاعات بصورت کدشده با الگوریتم MD5 در فایل config.php ذخیره شده‌اند. اگر قصد تغییر این اطلاعات را دارید، کافی‌است مقدار ثابت‌های UN و PW را در فایل config.php (یعنی خطوط 009 و 010) تغییر دهید. برای مثال، اگر می‌خواهید رمز عبور را 123456 بگذارید، این دستور را در یک صفحه قرار دهید و صفحه را اجرا کنید:

```
echo md5('123456');
```

تا با این خروجی مواجه‌شوید:

```
e10adc3949ba59abbe56e057f20f883e
```

حال کافی‌است این کد را جایگزین کد موجود درمقابل ثابت PW نمایید. ضمناً بعد از ورود نام کاربری و رمز عبور صحیح، برای مدیریت یک Session ایجاد می‌شود تا در صفحات بعدی نیاز به دریافت مجدد نام کاربری و رمز عبور نباشد. بعلاوه در سایر صفحات، ابتدا وجود Session مربوطه بررسی می‌شود و درصورت عدم تعریف Session، کاربر به صفحه ورود (Login) هدایت می‌شود.

برای ورود به بخش مدیریت، از آنجا که فایل‌های بخش مدیریت در پوشه management قراردارد، کافی‌است به‌انتهای آدرس سایت، عبارت management/ را اضافه‌کنیم. برای مثال، اگر آدرس صفحه اصلی سایت http://localhost/cms باشد، آدرس صفحه اصلی بخش مدیریت به‌صورت http://localhost/cms/management خواهد بود. حال به بررسی فایل‌های موجود در این صفحه، به‌ترتیبی که کاربر با آنها مواجه خواهد شد می‌پردازیم.

صفحه ورود به بخش مدیریت management/index.php

از آنجا که نام این فایل، index.php است، قاعدتاً اولین صفحه‌ای که با واردکردن آدرس بخش مدیریت با آن مواجه خواهیم‌شد، این فایل خواهد بود. ابتدا کد فایل را ملاحظه‌کنید:

```
001. <?php
002.     //Copyright محمد مصطفي شهركي @ http://www.ncis.ir
003.     require_once '../config.php';
004.     session_start();
005.     if(isset($_SESSION['manager'])) {
006.         unset($_SESSION['manager']);
007.     }
008. ?>
009. <!doctype html>
010. <html dir="rtl">
011. <head>
012. <meta charset="utf-8"/>
013. <title><?php echo TITLE; ?></title>
014. <link href="../style.css" rel="stylesheet" type="text/css"/>
015. </head>
016. <body style="background-image: url(../robot.jpg); background-repeat: no-repeat;">
017. <form action="manager.php" method="post">
018. <table align="center" border="0px" width="300px">
019. <tr align="right" valign="middle">
020. <th width="100px"><label for="un">نام کاربري</label></th>
021. <td><input class="transparent" id="un" maxlength="50" name="un" style="width: 100%;" type="text"/></td>
022. </tr>
```

```
<tr align="right" valign="middle">
<th><label for="pw">رمز عبور</label></th>
<td><input class="transparent" id="pw" maxlength="50" name="pw" style="width: 100%;" type="password"/></td>
</tr>
<tr align="center" valign="middle">
<td colspan="2"><input style="width: 100%;" type="submit" value="ورود"/></td>
</tr>
</table>
</form>
</body>
</html>
```

تغییرات کد نسبت به جلسه قبل:

- حذف دستورات ob_start و ob_end_flush از ابتدا و انتهای کد به‌دلیل عدم ضرورت
- تبدیل اسامی تگ‌ها به حروف کوچک
- مرتب‌شدن خاصیت‌های تگ‌ها به‌ترتیب الفبا
- حذف پرانتزهای دستور require_once در خط 003
- اضافه‌شدن شرط خط 004 به‌منظور جلوگیری از نمایش پیغام هشدار درصورت عدم‌تعریف متغیر Session مربوطه
- اضافه‌شدن تگ doctype در خط 009
- کوتاه‌شدن تگ meta در خط 012
- جایگزینی متغیر $title با ثابت TITLE در خط 013
- اضافه‌شدن عبارت px به اعداد جهت رعایت استانداردهای HTML در تعیین اندازه عناصر برحسب پیکسل
- حذف خاصیت width از عناصر th به‌جز سطر اول به‌دلیل عدم ضرورت

توضیح کد:

بخش HTML این‌کد نیازی به توضیح ندارد و صرفاً یک فرم ورود ساده ایجاد می‌کند که نام کاربری و رمز عبور را بعد از دریافت از کاربر، به روش Post برای صفحه manager.php ارسال می‌کند. اما در بخش PHP آن، بعد از ضمیمه‌کردن فایل config.php که در پوشه والد پوشه management قرار دارد (برای اشاره به پوشه والد از .. استفاده شده است) Session در خط 004 شروع می‌شود و در خط 005، وجود متغیر Session به‌نام manager بررسی می‌شود و درصورت وجود این متغیر، توسط تابع unset در خط 006 آنرا حذف می‌کنیم. علت این مسئله، امنیت بیشتر است: اگر یک‌نفر وارد سیستم شده‌باشد و بدون بستن مرورگر یا خروج از سایت، صرفاًصفحه مدیریت را بسته و سیستم را رها کرده‌باشد، با واردشدن فرد بعدی و واردکردن آدرس صفحه مدیریت، Session مدیر قبلی ازبین می‌رود و کاربر باید مجدداً نام کاربری و رمز عبور را واردکند.

فایل مقصد صفحه ورود management/manager.php

بدون مقدمه به‌سراغ کد این فایل می‌رویم:

```
<?php
    //Copyright محمد مصطفی شهرکی @ http://www.ncis.ir
    require_once '../config.php';
    session_start();
    if(isset($_SESSION['manager'])) {
        unset($_SESSION['manager']);
    }
?>
<!doctype html>
<html dir="rtl">
<head>
<meta charset="utf-8"/>
<title><?php echo TITLE; ?></title>
<link href="../style.css" rel="stylesheet" type="text/css"/>
</head>
<body>
<?php
    $flag = true;
    $vars = array('un', 'pw');
    foreach($vars as $var) {
        if(!isset($_POST[$var]) || $_POST[$var] == '') {
            $flag = false;
        }
    }
    $_SESSION['manager'] = ($flag && md5($_POST['un']) == UN && md5($_POST['pw']) == PW) ? true : false;
    header('location: management.php');
    exit();
?>
</body>
</html>
```

تغییرات کد نسبت به جلسه قبل:

- حذف دستورات ob_start و ob_end_flush از ابتدا و انتهای کد به‌دلیل عدم ضرورت
- تبدیل اسامی تگ‌ها به حروف کوچک
- مرتب‌شدن خاصیت‌های تگ‌ها به‌ترتیب الفبا
- حذف پرانتزهای دستور require_once در خط 003
- اضافه‌شدن شرط خط 004 به‌منظور جلوگیری از نمایش پیغام هشدار درصورت عدم‌تعریف متغیر Session مربوطه
- اضافه‌شدن تگ doctype در خط 009
- کوتاه‌شدن تگ meta در خط 012
- جایگزینی متغیر $title با ثابت TITLE در خط 013

توضیح کد:

ابتدا در تگ PHP ابتدای فایل، بعد از ضمیمه‌شدن فایل config.php، متغیر manager موجود در Session مدیر قبلی (درصورت وجود) ازبین می‌رود و سپس، درخطوط 018 تا 024 ارسال‌شدن صحیح پارامترهای un و pw به‌روش Post برای صفحه بررسی می‌شود. سپس در خط 025، متغیر manager در Session ایجاد می‌شود و مقدار آن، درصورتی‌که پارامترها به‌درستی ارسال شده‌باشند ($flag برابر با true باشد) و کد md5 نام کاربری و رمز عبور واردشده توسط کاربر به‌ترتیب با ثابت‌های UN و PW موجود در فایل config.php برابر باشند، true و درغیر اینصورت false خواهد بود. سپس در خط 026 کاربر به صفحه management.php هدایت می‌شود.

صفحه اصلی بخش مدیریت management/management.php

این صفحه، مشابه صفحه index.php در قسمت کاربری عمل می‌کند و وظیفه آن صرفاً قاب‌بندی صفحه است:

```
001. <?php
002.     //Copyright محمد مصطفي شهركي @ http://www.ncis.ir
003.     require_once '../config.php';
004.     session_start();
005.     if(!isset($_SESSION['manager']) || $_SESSION['manager'] != true) {
006.         header('location: index.php');
007.         exit();
008.     }
009. ?>
010. <!doctype html>
011. <html dir="rtl">
012. <head>
013. <meta charset="utf-8"/>
014. <title><?php echo TITLE; ?></title>
015. </head>
016. <frameset rows="100px,*">
017.     <frame frameborder="0" name="topFrame" noresize="noresize" scrolling="no" src="../top.php"/>
018.     <frameset cols="*,200px">
019.         <frame frameborder="0" name="mainFrame" noresize="noresize" scrolling="yes" src="main.php"/>
020.         <frame frameborder="0" name="linksFrame" noresize="noresize" scrolling="no" src="links.php"/>
021.     </frameset>
022. </frameset>
023. <noframes>
024. <body>
025. خطا: مرورگر شما از قاب بندي پشتيباني نمي كند.
026. </body>
027. </noframes>
028. </html>
```

تغییرات کد نسبت به جلسه قبل:

- حذف دستورات ob_start و ob_end_flush از ابتدا و انتهای کد به‌دلیل عدم ضرورت
- تبدیل اسامی تگ‌ها به حروف کوچک
- مرتب‌شدن خاصیت‌های تگ‌ها به‌ترتیب الفبا
- حذف پرانتزهای دستور require_once در خط 003
- تکمیل شرط خط 005 جهت امنیت بیشتر
- اضافه‌شدن تگ doctype در خط 010
- کوتاه‌شدن تگ meta در خط 013
- جایگزینی متغیر $title با ثابت TITLE در خط 014
- اضافه‌شدن عبارت px به اعداد جهت رعایت استاندارد HTML در تعیین اندازه عناصر برحسب پیکسل

توضیح کد:

از آنجا که قسمت HTML این فایل کاملاً مشابه فایل index.php در بخش کاربری است، نیاز به توضیح ندارد و و تنها نکته مهم در آن، نحوه اشاره به‌فایل top.php است که در پوشه والد قرار دارد (top.php/../)؛ اما در بخش PHP آن، ارائه اندکی توضیح مفید به‌نظر می‌رسد. پس‌از ضمیمه‌کردن فایل config.php و اجرای Session در خطوط 003 و 004، توسط شرط موجود در خطوط 005 تا 008، درصورتی‌که متغیر Session به‌نام manager ایجاد نشده‌باشد یا حاوی مقداری غیر از true باشد کاربر به‌صفحه index.php هدایت خواهد شد. درنتیجه، تنها راه مشاهده مابقی صفحه، وجود متغیر مربوطه و مقداردهی آن با true است که این‌موضوع نیز در صفحه مقصد فرم ورود و پس‌از بررسی صحت نام کاربری و رمز عبور واردشده توسط کاربر، انجام می‌شود. همان‌طور که مشخص‌است، برای قسمت بالای صفحه، از همان فایل top.php موجود در پوشه والد استفاده شده‌است که نیاز به توضیح مجدد ندارد.

صفحه لینک‌های سمت راست management/links.php

این صفحه، وظیفه نمایش لینک‌های بخش مدیریت را برعهده دارد و کد آن به‌صورت زیر است:

```
001. <?php
002.     //Copyright محمد مصطفي شهركي @ http://www.ncis.ir
003.     require_once '../config.php';
004.     if(!isset($_SERVER['HTTP_REFERER']) || $_SERVER['HTTP_REFERER'] != URL.'/management/management.php') {
005.         header('location: management.php');
006.         exit();
007.     }
008.     session_start();
009.     if(!isset($_SESSION['manager']) || $_SESSION['manager'] != true) {
010.         echo 'Illegal Access - <a href="'.URL.'" target="_top">Go to main page</a>'."\n";
011.         exit();
012.     }
013. ?>
014. <!doctype html>
015. <html dir="rtl">
016. <head>
017. <meta charset="utf-8"/>
018. <title><?php echo TITLE; ?></title>
019. <link href="../style.css" rel="stylesheet" type="text/css"/>
020. <base target="mainFrame"/>
021. </head>
022. <body>
023. <ul class="link">
024. <li><a href="<?php echo URL; ?>/management/main.php" target="_top">صفحه اصلي</a></li>
025. <li><a href="<?PHP echo URL; ?>/management/article_new.php">درج مطلب جديد</a></li>
026. <li><a href="<?PHP echo URL; ?>/management/article_edit.php">مديريت مطالب</a></li>
027. <li><a href="<?PHP echo URL; ?>/management/comment_new.php">درج نظر جديد</a></li>
028. <li><a href="<?PHP echo URL; ?>/management/comment_edit.php">مديريت نظرات</a></li>
029. <li><a href="<?PHP echo URL; ?>/management/logout.php" target="_top">خروج</a></li>
030. </ul>
031. </body>
032. </html>
```

تغییرات کد نسبت به جلسه قبل:

- حذف دستورات ob_start و ob_end_flush از ابتدا و انتهای کد به‌دلیل عدم ضرورت
- تبدیل اسامی تگ‌ها به حروف کوچک
- مرتب‌شدن خاصیت‌های تگ‌ها به‌ترتیب الفبا
- حذف پرانتزهای دستور require_once در خط 003
- اضافه‌شدن خطوط 004 تا 007 جهت جلوگیری از دسترسی مستقیم به فایل و ایجاد اجبار جهت بازشدن ازطریق قاب‌بندی صفحه management.php
- تکمیل شرط خط 009 جهت امنیت بیشتر
- تغییر نحوه هدایت کاربر به صفحه اصلی جهت رفع اشکالات احتمالی (توضیح کد را ملاحظه کنید)
- اضافه‌شدن تگ doctype در خط 014
- کوتاه‌شدن تگ meta در خط 017
- جایگزینی متغیر $title با ثابت TITLE در خط 018
- حذف تگ b به‌دلیل عدم‌نیاز (انجام وظیفه مشابه آن با کمک CSS)
- جایگزینی متغیر $url با ثابت URL در خطوط 024 تا 029

توضیح کد:

در پروتکل HTTP، هرگاه یک صفحه، صفحه دیگری را درخواست نماید، صفحه درخواست‌کننده به‌عنوان مراجعه‌کننده (Referer) شناخته می‌شود که این درخواست می‌تواند

به هر شکل ممکن باشد (قاب‌بندی، کلیک روی یک لینک به صفحه و...). در PHP، مراجعه‌کننده به یک صفحه را می‌توان ازطریق متغیر PHP_REFERER که در آرایه فوق سراسری (Super Global) به‌نام $_SERVER قراردارد، بررسی‌نمود. بنابراین، هرگاه یک صفحه مستقیماً ازطریق تایپ نشانی آن در مرورگر درخواست‌شود، طبیعتاً چنین متغیری وجود نخواهد داشت. در خطوط 004 تا 007، این مسئله بررسی می‌شود و اگر این صفحه (links.php) مستقیماً یا ازطریق صفحه‌ای غیر از management.php درخواست شده باشد، کاربر به صفحه management.php یعنی صفحه اصلی بخش مدیریت هدایت خواهد شد. در شرط خط 009 نیز درصورتی‌که برای کاربر متغیر manager ازنوع Session ایجاد نشده‌باشد یا حاوی مقداری غیر از true باشد، یک پیغام مبنی‌بر دسترسی غیرمجاز همراه با یک لینک به صفحه اصلی سایت (بخش کاربری) به وی نشان‌داده می‌شود. علت استفاده از لینک، امکان تنظیم خاصیت target آن برروی _top است تا اگر کاربر صفحه links.php را در یک قاب بازکند، بازهم باکلیک برروی لینک، صفحه اصلی سایت، تمام پنجره مرورگر را بپوشاند. مابقی کدهای صفحه نیز HTML هستند و نیازی به توضیح ندارند.

صفحه اصلی سمت چپ management/main.php

این صفحه، محتوای خاصی ندارد و صرفاً کاربر را به کلیک‌کردن برروی یکی از گزینه‌های سمت راست برای استفاده از بخش مدیریت دعوت می‌کند:

```
001. <?php
002.     //Copyright محمدمصطفی شهرکی@ http://www.ncis.ir
003.     require_once '../config.php';
004.     if(!isset($_SERVER['HTTP_REFERER']) || $_SERVER['HTTP_REFERER'] != URL.'/management/management.php') {
005.         header('location: management.php');
006.         exit();
007.     }
008.     session_start();
009.     if(!isset($_SESSION['manager']) || $_SESSION['manager'] != true) {
010.         echo 'Illegal Access - <a href="'.URL.'" target="_top">Go to main page</a>'."\n";
011.         exit();
012.     }
013. ?>
014. <!doctype html>
015. <html dir="rtl">
016. <head>
017. <meta charset="utf-8"/>
018. <title><?php echo TITLE; ?></title>
019. <link href="../style.css" rel="stylesheet" type="text/css"/>
020. </head>
021. <body style="background-image: url(../robot.jpg); background-repeat: no-repeat;">
022. <table border="0px" cellpadding="0px" cellspacing="0px" style="position: absolute; left: 0px; height: 100%;
     top: 0px; width: 100%;">
023. <tr align="center" valign="middle">
024. <td>
025. <marquee behavior="alternate" direction="left" height="10%" scrolldelay="1" scrollamount="1" width="100%">
026. <a href="management.php" target="_top">(( یکی از لینکهای سمت راست را انتخاب کنید ))</a>
027. </marquee>
028. </td>
029. </tr>
030. </table>
031. </body>
032. </html>
```

تغییرات کد نسبت به جلسه قبل:

- حذف دستورات ob_start و ob_end_flush از ابتدا و انتهای کد به‌دلیل عدم ضرورت
- تبدیل اسامی تگ‌ها به حروف کوچک
- مرتب‌شدن خاصیت‌های تگ‌ها به‌ترتیب الفبا
- حذف پرانتزهای دستور require_once در خط 003
- اضافه‌شدن خطوط 004 تا 007 جهت جلوگیری از دسترسی مستقیم به فایل و ایجاد اجبار جهت بازشدن ازطریق قاب‌بندی صفحه management.php
- تکمیل شرط خط 009 جهت امنیت بیشتر
- تغییر نحوه هدایت کاربر به صفحه اصلی جهت رفع اشکالات احتمالی (توضیح کد را ملاحظه کنید)
- اضافه‌شدن تگ doctype در خط 014
- کوتاه‌شدن تگ meta در خط 017
- جایگزینی متغیر $title با ثابت TITLE در خط 018
- تغییر روش تعیین تصویر پس‌زمینه به ساختار CSS در خط 021 جهت رعایت استانداردهای جدید HTML
- تغییر روش تعیین اندازه‌های جدول به‌صورت تمام‌صفحه به ساختار CSS در خط 022 جهت رعایت استانداردهای جدید HMTL5
- اضافه‌شدن عبارت px به اعداد جهت رعایت استاندارد HTML در تعیین اندازه عناصر برحسب پیکسل

توضیح کد:

بخش HTML این کد نیاز به توضیح ندارد و بخش PHP آن نیز کاملاً مشابه فایل links.php است و لذا، از توضیح مجدد آن خودداری می‌کنیم.

صفحه درج مطلب جدید management/article_new.php

همان‌طور که از عنوان این فایل مشخص‌است، مدیر سایت برای درج یک مقاله جدید، از آن استفاده می‌کند. در این جلسه، تغییرات ایجادشده در بخش مدیریت مقالات (درج، ویرایش، حذف) نسبتاً بیشتر از سایر بخش‌ها است. برای مثال، قابلیت آپلود (Upload) فایل مقاله به وب‌سایت نیز در صفحات درج و ویرایش و مدیریت فایل‌های آپلودشده (حذف فایل قبلی مقاله درصورت آپلود فایل جدید در صفحه ویرایش) جزو ویژگی‌های جدیدی است که در جلسه قبل وجود نداشتند. کد این فایل به‌صورت زیر است:

```
001. <?php
002.     //Copyright محمد مصطفي شهركي @ http://www.ncis.ir
003.     require_once '../config.php';
004.     require_once '../db.php';
005.     if(!isset($_SERVER['HTTP_REFERER']) || $_SERVER['HTTP_REFERER'] != URL.'/management/links.php') {
006.         header('location: main.php');
007.         exit();
008.     }
009.     session_start();
010.     if(!isset($_SESSION['manager']) || $_SESSION['manager'] != true) {
011.         echo 'Illegal Access - <a href="'.URL.'" target="_top">Go to main page</a>'."\n";
012.         exit();
013.     }
014. ?>
015. <!doctype html>
016. <html dir="rtl">
017. <head>
018. <meta charset="utf-8"/>
019. <title><?php echo TITLE; ?></title>
020. <link href="../style.css" rel="stylesheet" type="text/css"/>
021. </head>
022. <body style="background-image: url(../robot.jpg); background-repeat: no-repeat;">
023. <?php
024.     $articles = new Articles();
025.     echo '<span class="red">شماره مطلب: '.$articles->GetNewID().'</span><br/>'."\n";
026. ?>
027. <form action="article_result.php?action=new" enctype="multipart/form-data" method="post">
028. <table border="0px" cellpadding="0px" cellspacing="2px" width="300px">
029. <tr align="right" valign="middle">
030. <th width="100px"><label for="title">عنوان</label></th>
031. <td><input class="transparent" id="title" maxlength="255" name="title" style="width: 100%;"
     type="text"/></td>
032. </tr>
033. <tr align="right" valign="middle">
034. <th><label for="abstract">چکیده</label></th>
035. <td><input class="transparent" id="abstract" maxlength="255" name="abstract" style="width: 100%;"
     type="text"/></td>
036. </tr>
037. <tr align="right" valign="middle">
038. <th><label for="body">ادامه مطلب</label></th>
039. <td><textarea class="transparent" id="body" name="body" rows="5" style="width: 100%;"></textarea></td>
040. </tr>
041. <tr align="right" valign="middle">
042. <th><label for="file">فایل PDF</label></th>
043. <td>
044. <input name="MAX_FILE_SIZE" type="hidden" value="1048576"/>
045. <input class="transparent" id="file" name="file" size="15" type="file"/>
046. </td>
047. </tr>
048. <tr align="right" valign="middle">
049. <td colspan="2"><input style="width: 100%;" type="submit" value="ثبت"/></td>
050. </tr>
051. </table>
052. </form>
053. </body>
054. </html>
```

تغییرات کد نسبت به جلسه قبل:

- حذف دستورات ob_start و ob_end_flush از ابتدا و انتهای کد به‌دلیل عدم ضرورت
- تبدیل اسامی تگ‌ها به حروف کوچک
- مرتب‌شدن خاصیت‌های تگ‌ها به‌ترتیب الفبا
- حذف پرانتزهای دستور require_once در خطوط 003 و 004
- اضافه‌شدن خطوط 005 تا 008 جهت جلوگیری از دسترسی مستقیم به فایل و ایجاد اجبار جهت بازشدن ازطریق لینک صفحه links.php
- هدایت کاربر به صفحه main.php به‌جای management.php در خط 006

- تکمیل شرط خط 010 جهت امنیت بیشتر
- تغییر نحوه هدایت کاربر به صفحه اصلی جهت رفع اشکالات احتمالی
- اضافه‌شدن تگ doctype در خط 015
- کوتاه‌شدن تگ meta در خط 018
- جایگزینی متغیر $title با ثابت TITLE در خط 019
- تغییر روش تعیین تصویر پس‌زمینه به ساختار CSS در خط 022 جهت رعایت استانداردهای جدید HTML
- جایگزینی تگ font با تگ span در خط 025 به‌منظور رعایت استانداردهای جدید HTML
- اضافه‌شدن خاصیت enctype با مقدار multipart/form-data به تگ form جهت ایجاد امکان آپلود فایل توسط فرم
- اضافه‌شدن عبارت px به اعداد جهت رعایت استاندارد HTML در تعیین اندازه عناصر برحسب پیکسل
- کم‌شدن پهنای جدول از ۵۰۰ پیکسل به ۳۰۰ پیکسل در خط 028 جهت ایجاد ظاهر مناسب‌تر برای فرم
- حذف خاصیت width تگ td در خط 031 به‌دلیل عدم ضرورت
- تغییر نحوه مشخص‌کردن فایل در خطوط 042 تا 046 از اعلام نام فایل به انتخاب فایل PDF با اندازه حداکثر ۱ مگابایت جهت آپلود (توضیح کد را ملاحظه‌کنید)

توضیح کد:

بعد از ضمیمه‌شدن فایل‌های config.php و db.php از پوشه والد، ابتدا وجود صفحه links.php بعنوان مراجعه‌کننده (Referer) بررسی می‌شود و اگر چنین نباشد، به‌معنای بازشدن صفحه با روشی غیر از کلیک‌کردن روی لینک موجود در صفحه links.php بوده‌است و درنتیجه، کاربر به صفحه main.php هدایت می‌شود. در ادامه، اگر متغیر manager از نوع Session برای کاربر تعریف نشده باشد یا حاوی مقداری غیر از true باشد، پیغام دسترسی غیرمجاز همراه با یک لینک جهت بازگشت به صفحه اصلی سایت (بخش کاربری) برای کاربر به‌نمایش در می‌آید. در خط 024 یک شئ به‌نام $articles از کلاس Articles ایجاد می‌شود و در خط 025 با فراخوانی تابع GetNewID، شماره مقاله جدید به‌دست آمده و به مدیر نشان‌داده می‌شود تا بداند مطلب جدید، چندمین مطلب موجود در سایت خواهد بود. نکته مهم در این فایل، نحوه ایجاد امکان آپلود فایل برای کاربر است. برای اینکه کاربران بتوانند فایل‌های موردنظر خود را آپلود نمایند، موارد زیر ضروری است:

- روش ارسال فرم، Post باشد (method="post")
- خاصیت enctype برای فرم ذکر شود و مقدار آن، multipart/form-data باشد (enctype="multipart/form-data")
- برای آنکه کاربر بتواند فایل خود را انتخاب‌کند، از تگ input با خاصیت type برابر با file (type="file") استفاده شود
- برای آنکه بتوانیم حداکثر اندازه مجاز فایل را تعیین‌کنیم، از تگ زیر استفاده می‌کنیم:

```
<input name="MAX_FILE_SIZE" type="hidden" value="1048576"/>
```

که در تگ فوق، مقدار value حداکثر اندازه مجاز برحسب بایت خواهد بود (در مثال فوق، 1MB = 1024*1024B = 1048576B).

البته تعیین حداکثر اندازه برای آپلود فایل اختیاری است و اگر این‌مقدار ذکر نشود، مقدار تعیین‌شده در تنظیمات PHP (فایل php.ini) در سطر زیر:

```
upload_max_filesize = 2M
```

درنظر گرفته می‌شود که بطور پیش‌فرض، ۲ مگابایت خواهد بود. بعلاوه، مقدار MAX_FILE_SIZE نمی‌تواند بیشتر از upload_max_filesize باشد.

ضمناً در کد فوق، مقصد فرم فایل article_result.php است که پارامتر action با مقدار new ازطریق آدرس (متد Get) برای آن ارسال می‌شود. بنابراین، همانطور که ملاحظه می‌کنید، می‌توان برای یک صفحه بطور همزمان پارامترهای مختلف را به‌روش Post و Get ارسال‌نمود. در این صفحه، پارامتر action با روش Get و اطلاعات واردشده توسط مدیر درون فرم با روش Post برای صفحه article_result.php ارسال می‌شود. از آنجا که صفحه article_result.php کارهای دیگری نیز بجز درج مطلب جدید انجام خواهد داد، توضیح آنرا بعد از فایل acrticle_edit.php ارائه خواهیم‌کرد.

صفحه مدیریت مطالب management/article_edit.php

این فایل، جهت مدیریت مطالب موجود به‌کار می‌رود و توسط آن، مدیر می‌تواند علاوه‌بر مشاهده فهرست خلاصه‌ای از مطالب موجود، اقدام به ویرایش و حذف/بازیابی آنها نماید.

کد این فایل به‌صورت زیر است:

```
001. <?php
002.     //Copyright محمدمصطفی‌شهرکی@ http://www.ncis.ir
003.     require_once '../config.php';
004.     require_once '../db.php';
005.     $referers = array(URL.'/management/article_edit.php', URL.'/management/links.php');
006.     if(!isset($_SERVER['HTTP_REFERER']) || !in_array($_SERVER['HTTP_REFERER'], $referers)) {
007.         header('location: main.php');
008.         exit();
009.     }
010.     session_start();
011.     if(!isset($_SESSION['manager']) || $_SESSION['manager'] != true) {
012.         echo 'Illegal Access - <a href="'.URL.'" target="_top">Go to main page</a>'."\n";
013.         exit();
014.     }
015. ?>
016. <!doctype html>
017. <html dir="rtl">
```

```
018. <head>
019. <meta charset="utf-8"/>
020. <title><?php echo TITLE; ?></title>
021. <link href="../style.css" rel="stylesheet" type="text/css"/>
022. </head>
023. <body style="background-image: url(../robot.jpg); background-repeat: no-repeat;">
024. <?php
025.     $articles = new Articles();
026.     if(!isset($_GET['id']) || $_GET['id'] == '') {
027.         $result = $articles->SelectAll();
028.         if($result !== false && mysql_num_rows($result) > 0) {
029. ?>
030. <table border="1px" cellpadding="2px" cellspacing="0px" width="100%">
031. <tr align="center" valign="middle">
032. <th width="5%">ردیف</th>
033. <th width="10%">عنوان</th>
034. <th width="15%">چکیده</th>
035. <th width="25%">ادامه مطلب</th>
036. <th width="10%">نام فایل</th>
037. <th width="10%">بازدید</th>
038. <th width="10%">وضعیت</th>
039. <th width="15%">عملیات</th>
040. </tr>
041. <?php
042.             while($row = mysql_fetch_assoc($result)) {
043.                 echo '<tr align="center" valign="middle">'."\n";
044.                 echo '<td>'.($row['id'] != '' ? $row['id'] : ' ').'</td>'."\n";
045.                 echo '<td>'.($row['title'] != '' ? $row['title'] : ' ').'</td>'."\n";
046.                 echo '<td style="text-align: justify;">'.($row['abstract'] != '' ? $row['abstract'] :
                        ' ').'</td>'."\n";
047.                 echo '<td style="text-align: justify;">'.($row['body'] != '' ? $row['body'] :
                        ' ').'</td>'."\n";
048.                 echo '<td>'.($row['filename'] != '' ? $row['filename'] : ' ').'</td>'."\n";
049.                 echo '<td>'.($row['visits'] != '' ? $row['visits'] : ' ').'</td>'."\n";
050.                 echo '<td>'.($row['status'] == 'normal' ? 'عادي' : 'حذف شده').'</td>'."\n";
051.                 echo '<td>';
052.                 echo '<a href="article_edit.php?id='.$row['id'].'">[ویرایش]</a>';
053.                 echo '   ';
054.                 echo '<a href="article_result.php?action='.($row['status'] == 'normal' ? 'delete' :
                        'recover').'&id='.$row['id'].'">['.($row['status']=='normal' ? 'حذف' :
                        'بازیابي').']</a>';
055.                 echo '</td>'."\n";
056.                 echo '</tr>'."\n";
057.             }
058. ?>
059. </table>
060. <?php
061.         }
062.         else {
063.             echo 'هیچ مطلبي پیدا نشد.<br/>'."\n";
064.         }
065.     }
066.     else {
067.         $result = $articles->SelectRow($_GET['id']);
068.         if($result !== false && mysql_num_rows($result) > 0) {
069.             $row = mysql_fetch_assoc($result);
070. ?>
071. <form action="article_result.php?action=edit" enctype="multipart/form-data" method="post">
072. <input name="id" type="hidden" value="<?php echo $row['id']; ?>"/>
073. <table border="0px" cellpadding="0px" cellspacing="2px" width="300px">
074. <tr align="right" valign="middle">
075. <th width="100px"><label for="title">عنوان</label></th>
076. <td><input class="transparent" id="title" maxlength="255" name="title" style="width: 100%;" type="text"
     value="<?php echo $row['title']; ?>"/></td>
077. </tr>
078. <tr align="right" valign="middle">
079. <th><label for="abstract">چکیده</label></th>
080. <td><input class="transparent" id="abstract" maxlength="255" name="abstract" style="width: 100%;"
     type="text" value="<?php echo $row['abstract']; ?>"/></td>
081. </tr>
082. <tr align="right" valign="middle">
083. <th><label for="body">ادامه مطلب</label></th>
084. <td><textarea class="transparent" id="body" name="body" rows="5" style="width: 100%;">
     <?php echo $row['body']; ?></textarea></td>
085. </tr>
086. <tr align="right" valign="middle">
087. <th><label for="filename">فایل PDF</label></th>
088. <td>
089. <input name="MAX_FILE_SIZE" type="hidden" value="1048576"/>
```

```
090. <input class="transparent" id="file" name="file" size="15" type="file"/>
091. </td>
092. </tr>
093. <tr align="right" valign="middle">
094. <th><label for="visits">بـازدیـد</label></th>
095. <td><input class="transparent" id="visits" maxlength="255" name="visits" style="width: 100%;" type="text"
     value="<?php echo $row['visits']; ?>"/></td>
096. </tr>
097. <tr align="right" valign="middle">
098. <td colspan="2"><input style="width: 100%;" type="submit" value="ثـبت"/></td>
099. </tr>
100. </table>
101. </form>
102. <?php
103.         }
104.     }
105. ?>
106. </body>
107. </html>
```

تغییرات کد نسبت به جلسه قبل:

- حذف دستورات ob_start و ob_end_flush از ابتدا و انتهای کد به‌دلیل عدم ضرورت
- تبدیل اسامی تگ‌ها به حروف کوچک
- مرتب‌شدن خاصیت‌های تگ‌ها به‌ترتیب الفبا
- حذف پرانتزهای دستور require_once در خطوط 003 و 004
- اضافه‌شدن خطوط 005 تا 009 جهت جلوگیری از دسترسی مستقیم و ایجاد اجبار جهت بازشدن ازطریق صفحات links.php، و article_edit.php
- هدایت کاربر به صفحه main.php به‌جای management.php در خط 007
- تکمیل شرط خط 011 جهت امنیت بیشتر
- تغییر نحوه هدایت کاربر به صفحه اصلی جهت رفع اشکالات احتمالی
- اضافه‌شدن تگ doctype در خط 016
- کوتاه‌شدن تگ meta در خط 019
- جایگزینی متغیر $title با ثابت TITLE در خط 020
- تغییر روش تعیین تصویر پس‌زمینه به ساختار CSS در خط 023 جهت رعایت استانداردهای جدید HTML
- اضافه‌شدن خاصیت enctype با مقدار multipart/form-data به تگ form در خط 071 جهت ایجاد امکان آپلود فایل توسط فرم
- اضافه‌شدن عبارت px به اعداد جهت رعایت استاندارد HTML در تعیین اندازه عناصر برحسب پیکسل
- کم‌شدن پهنای جدول از ۵۰۰ پیکسل به ۳۰۰ پیکسل در خط 073 جهت ایجاد ظاهر مناسب‌تر برای فرم
- حذف خاصیت width تگ td در خط 076 به‌دلیل عدم ضرورت
- تغییر نحوه مشخص‌کردن فایل در خطوط 087 تا 091 از اعلام نام فایل به انتخاب فایل PDF با اندازه حداکثر ۱ مگابایت جهت آپلود (توضیح کد را ملاحظه‌کنید)

توضیح کد:

این فایل توسط لینک‌ها و فرم‌های موجود در صفحات links.php و article_edit.php قابل ارجاع است. بنابراین، ابتدا در خط 005 نشانی این دو فایل در یک آرایه به‌نام $referers قرار می‌گیرد و سپس، در خط 006، بررسی می‌شود که اگر صفحه جاری دارای یک مراجعه‌کننده نباشد یا مراجعه‌کننده آن، جزو آرایه $referers نباشد (برابر با یکی از مقادیر آن نباشد)، کاربر به صفحه main.php هدایت‌شود. در ادامه نیز درصورتی‌که برای کاربر متغیر manager از نوع Session و با مقدار true ایجاد نشده‌باشد (کاربر وارد سایت نشده یا اصطلاحاً Login نکرده‌باشد)، ضمن نمایش پیغام خطای دسترسی غیرمجاز، لینک صفحه اصلی سایت به وی نشان‌داده می‌شود در خط 025 یک شئ به‌نام $articles از کلاس Articles ایجاد می‌شود. صفحه ویرایش مطالب به‌نحوی طراحی شده‌است که اگر شماره مطلب برای آن ارسال نشده‌باشد، فهرست تمامی مطالب را نشان می‌دهد و درصورت دریافت شماره مطلب، اطلاعات همان مطلب را درقالب یک فرم جهت ویرایش به مدیر نشان می‌دهد. با توجه به این روش طراحی، در خط 026 ارسال‌شدن پارامتر id ازطریق آدرس صفحه (روش Get) و خالی‌نبودن آن بررسی می‌شود و درصورت عدم‌ارسال یا خالی‌بودن این پارامتر، با فراخوانی تابع SelectAll از شئ $articles در خط 027، تمامی مطالب از جدول موجود در پایگاه داده‌ها استخراج می‌شوند و در متغیر $result قرار می‌گیرند. سپس در خط 028 این متغیر بررسی می‌شود که اولاً مخالف false باشد (دستور MySQL اجرا شده روی پایگاه داده‌ها دارای خطا نباشد) و ثانیاً تعداد رکوردهای بازگردانده‌شده بیشتر از صفر باشد (جدول مطالب خالی نباشد) و درصورت برقراری شرایط فوق، یک جدول برای نمایش مطالب ایجاد می‌شود و در سطر اول آن، عناوین ستون‌ها ذکر می‌گردد. نکته قابل‌توجه، حلقه while موجود در خطوط 042 تا 057 است که به‌کمک دستور mysql_fetch_assoc، هربار یک رکورد را از متغیر $result استخراج‌کرده و درداخل متغیر $row قرار می‌دهد. از آنجا که دستور فوق با رسیدن به آخرین رکورد و عدم‌وجود رکورد دیگر برای استخراج، مقدار false باز می‌گرداند، طبیعتاً بعد از رسیدن به آخرین رکورد متغیر $row حاوی مقدار false خواهد بود و به‌طور خودکار از حلقه خارج خواهیم شد. در درون حلقه نیز ابتدا در خط 043 یک سطر جدید به جدول اضافه می‌شود. سپس در

خطوط 044 تا 049 شماره ردیف، عنوان، چکیده، ادامه مطلب، نام فایل و تعداد بازدید مقاله در سطر مربوطه درج می‌شوند. شرط‌های موجود در این خطوط نیز برای کنترل نحوه ایجاد خانه بدون محتوا درنظر گرفته شده‌است (در HTML اگر بخواهیم یک خانه بدون محتوا داشته‌باشیم، برای ترسیم صحیح آن، باید درون آن خانه حداقل یک کارکتر فاصله جداناپذیر یعنی   – مخفف Non-Breakable Space – داشته‌باشیم، درغیر این‌صورت، آن خانه ترسیم نمی‌شود). در سطر 050 وضعیت مطلب برحسب محتوای فیلد status آن، «عادی» (`status`='normal') یا «حذف‌شده» (`status`='deleted') درنظر گرفته و نمایش‌داده می‌شود. در خانه مربوط‌به عملیات نیز نخست در خط 052، یک لینک به صفحه article_edit.php همراه‌با شماره ردیف مقاله که بعنوان پارامتر id ازطریق آدرس ارسال می‌شود، به‌نمایش در می‌آید و در ادامه فاصله کافی توسط خط 053 ایجاد می‌گردد. در خط 054 نیز برحسب مقدار فیلد status، یکی از لینک‌های مربوط‌به عملیات «حذف» یا «بازیابی» ظاهر می‌شود. برای مثال اگر فیلد status مطلب شماره ۱ (`id`='1') برابر با normal باشد (`status`='normal') به‌معنای عادی‌بودن آن مطلب است و درنتیجه، لینک مربوط‌به حذف باید ظاهر شود:

```
<a href="article_result.php?action=delete&id=1>حذف</a>
```

و درغیر این‌صورت (`status`='deleted')، مطلب مذکور از نوع حذف‌شده است و باید لینک مربوط‌به بازیابی ظاهر گردد:

```
<a href="article_result.php?action=recover&id=1>بازیابی</a>
```

قسمت else مربوط‌به شرط نیز در خطوط 062 تا 064 درصورتی اجرا می‌گردد که هیچ رکوردی پیدا نشود (جدول خالی باشد) یا دستور MySQL اجراشده دارای خطا باشد و درهرصورت، پیغام خطای مناسب به مدیر نشان داده خواهد شد؛ اما قسمت else موجود در خطوط 066 تا 104 مربوط‌به دستور if تعریف‌شده در خط 026 است و زمانی اجرا خواهد شد که پارامتر id ازطریق آدرس (روش Get) برای صفحه ارسال شده باشد (لینک ویرایش مطلب که در خط 052 ساخته می‌شود را نگاه‌کنید). در این قسمت ابتدا در خط 067 تابع SelectRow از شئ $articles فراخوانی می‌شود و شماره مطلب که به‌روش Get برای صفحه ارسال شده است، برای تابع فوق ارسال می‌شود تا فقط همان مطلب از جدول در پایگاه داده‌ها استخراج‌شود و در متغیر $result قرارگیرد. حال ابتدا در خط 068 عدم‌وجود خطا در دستور MySQL و همچنین خالی‌نبودن جدول بررسی می‌شود و درصورت برقراری شرایط فوق، در خط 069 تنها رکورد موجود در $result توسط تابع mysql_fetch_assoc استخراج می‌شود و درون متغیر $row قرار می‌گیرد. در ادامه نیز فرمی مشابه فایل article_new.php به مدیر نشان‌داده خواهد شد، با این تفاوت که مقادیر موجود در پایگاه داده‌ها از قبل بعنوان مقادیر قبلی درون عناصر مربوطه (عنوان، چکیده و...) نوشته می‌شود. مقصد این فرم نیز فایل article_result.php است، با این تفاوت که عمل مربوطه، edit خواهد بود (پارامتر action که ازطریق آدرس و به‌روش Get ارسال می‌شود، برابر با edit است).

صفحه مقصد عملیات برروی مطالب management/article_result.php

کد این صفحه، بیشترین اهمیت را در بین صفحاتی از سایت دارد که با مقالات سروکار دارند، زیرا کلیه اعمال شامل درج مقاله جدید، ویرایش مقاله موجود و همچنین حذف و بازیابی مقالات برعهده این فایل است. درحقیقت سایر فایل‌ها صرفاً اطلاعات موردنیاز را از مدیر دریافت می‌کنند و این فایل، عملیات اصلی را برحسب نوع آن که توسط پارامتر action ازطریق آدرس (به‌روش Get) برای فایل ارسال شده‌است، انجام می‌دهد. بنابراین، پیشنهاد می‌کنیم کد این فایل و توضیحات آنرا به‌دقت مطالعه‌کنید:

```
001. <?php
002.     //Copyright محمدمصطفی‌شهرکی@ http://www.ncis.ir
003.     require_once '../config.php';
004.     require_once '../db.php';
005.     $referers = array(URL.'/management/article_edit.php', URL.'/management/article_new.php');
006.     if(!isset($_SERVER['HTTP_REFERER']) || !in_array(substr($_SERVER['HTTP_REFERER'], 0, strlen(URL) + 28),
            $referers)) {
007.         header('location: main.php');
008.         exit();
009.     }
010.     session_start();
011.     if(!isset($_SESSION['manager']) || $_SESSION['manager'] != true) {
012.         echo 'Illegal Access - <a href="'.URL.'" target="_top">Go to main page</a>'."\n";
013.         exit();
014.     }
015. ?>
016. <!doctype html>
017. <html dir="rtl">
018. <head>
019. <meta charset="utf-8"/>
020. <title><?php echo TITLE; ?></title>
021. <link href="../style.css" rel="stylesheet" type="text/css"/>
022. </head>
023. <body style="background-image: url(../robot.jpg); background-repeat: no-repeat;">
024. <?php
025.     if(isset($_GET['action']) && $_GET['action'] != '') {
026.     $articles = new Articles();
027.         switch(strtolower($_GET['action'])) {
028.             case 'new':
029.                 $flag = true;
030.                 $result = 0;
031.                 if(!(isset($_FILES['file'])) || $_FILES['file']['error'] != 0) {
032.                     $flag = false;
033.                 }
034.                 $vars = array('abstract', 'body', 'title');
035.                 foreach($vars as $var) {
036.                     if(!isset($_POST[$var]) || $_POST[$var] == '') {
037.                         $flag = false;
```

```
038.                     }
039.                 }
040.                 if($flag && move_uploaded_file($_FILES['file']['tmp_name'],
                                                    '../files/'.$_FILES['file']['name'])) {
041.                     $result = $articles->Insert($_POST['title'], $_POST['abstract'], $_POST['body'],
                                                     $_FILES['file']['name']);
042.                 }
043.                 echo 'مطلب'.($result > 0 ? 'با موفقیت ثبت شد' : 'ثبت نشد').'.<br/>'."\n";
044.                 break;
045.             case 'edit':
046.                 $flag = true;
047.                 $result = 0;
048.                 if(!(isset($_FILES['file'])) || $_FILES['file']['error'] != 0) {
049.                     $flag = false;
050.                 }
051.                 $vars = array('abstract', 'body', 'id', 'title', 'visits');
052.                 foreach($vars as $var) {
053.                     if(!isset($_POST[$var]) || $_POST[$var] == '') {
054.                         $flag = false;
055.                     }
056.                 }
057.                 if($flag && move_uploaded_file($_FILES['file']['tmp_name'],
                                                    '../files/'.$_FILES['file']['name'])) {
058.                     $article = $articles->SelectRow($_POST['id']);
059.                     if($article !== false && mysql_num_rows($article) > 0) {
060.                         $article = mysql_fetch_assoc($article);
061.                         if(file_exists('../files/'.$article['filename'])) {
062.                             unlink('../files/'.$article['filename']);
063.                         }
064.                     }
065.                     $result = $articles->Update($_POST['id'], $_POST['title'], $_POST['abstract'],
                                                    $_POST['body'], $_FILES['file']['name'],$_POST['visits']);
066.                 }
067.                 echo 'مطلب'.($result > 0 ? 'با موفقیت ویرایش شد' : 'ویرایش نشد').'.<br/>'."\n";
068.                 break;
069.             case 'delete':
070.                 $result = 0;
071.                 if(isset($_GET['id']) && $_GET['id'] != '' && is_numeric($_GET['id'])) {
072.                     $result = $articles->Delete($_GET['id']);
073.                 }
074.                 echo 'مطلب'.($result > 0 ? 'با موفقیت حذف شد' : 'حذف نشد').'.<br/>'."\n";
075.                 break;
076.             case 'recover':
077.                 $result = 0;
078.                 if(isset($_GET['id']) && $_GET['id'] != '' && is_numeric($_GET['id'])) {
079.                     $result = $articles->Recover($_GET['id']);
080.                 }
081.                 echo 'مطلب'.($result > 0 ? 'با موفقیت بازیابی شد' : 'بازیابی نشد').'.<br/>'."\n";
082.                 break;
083.         }
084.     }
085. ?>
086. </body>
087. </html>
```

تغییرات کد نسبت به جلسه قبل:

- حذف دستورات ob_start و ob_end_flush از ابتدا و انتهای کد به‌دلیل عدم ضرورت
- تبدیل اسامی تگ‌ها به حروف کوچک
- مرتب‌شدن خاصیت‌های تگ‌ها به‌ترتیب الفبا
- حذف پرانتزهای دستور require_once در خطوط 003 و 004
- اضافه‌شدن خطوط 005 تا 009 جهت جلوگیری از دسترسی مستقیم و ایجاد اجبار جهت بازشدن ازطریق صفحات article_edit.php، و article_new.php
- هدایت کاربر به صفحه main.php به‌جای management.php در خط 007
- تکمیل شرط خط 011 جهت امنیت بیشتر
- تغییر نحوه هدایت کاربر به صفحه اصلی جهت رفع اشکالات احتمالی
- اضافه‌شدن تگ doctype در خط 016
- کوتاه‌شدن تگ meta در خط 019
- جایگزینی متغیر $title با ثابت TITLE در خط 020
- تغییر روش تعیین تصویر پس‌زمینه به ساختار CSS در خط 023 جهت رعایت استانداردهای جدید HTML

- اضافه‌شدن شرط موجود در خط 025 جهت جلوگیری از بروز خطا درصورت عدم ارسال پارامتر action به روش Get برای صفحه
- اضافه‌شدن متغیر کنترلی $flag در خطوط 029 و 046 با مقدار اولیه true جهت بررسی ارسال صحیح تمامی پارامترهای لازم برای انجام عملیات روی مطالب
- اضافه‌شدن خطوط 030 تا 040 جهت کنترل صحت ارسال پارامترهای لازم و انجام عملیات آپلود فایل در زمان ایجاد مطلب جدید
- اضافه‌شدن خطوط 047 تا 064 جهت کنترل صحت ارسال پارامترهای لازم و انجام عملیات آپلود فایل و حذف فایل قبلی در زمان ویرایش مطلب موجود
- اضافه‌شدن خطوط 070 و 071 جهت کنترل صحت ارسال شماره مطلب به روش Get برای صفحه در زمان حذف مطلب
- اضافه‌شدن خطوط 077 و 078 جهت کنترل صحت ارسال شماره مطلب به روش Get برای صفحه در زمان بازیابی مطلب

توضیح کد:

در خط 006، ابتدا وجود مراجعه‌کننده صفحه بررسی می‌شود و درصورت وجود، این مراجعه‌کننده باید در آرایه تعریف‌شده در خط 005 موجود باشد. علت استفاده از تابع substr، محدودکردن مراجعه‌کننده است. برای درک بهتر، به آدرس صفحه ویرایش مطلب شماره ۲ که اطلاعات را پس‌از دریافت ازطریق فرم، به این صفحه می‌فرستد دقت‌کنید:

http://localhost/cms**/management/article_edit.php?id=2**

در آدرس فوق که مراجعه‌کننده صفحه جاری خواهد بود، قسمتی که با قلم معمولی نوشته شده‌است، برابر با ثابت URL می‌باشد و قسمت پررنگ‌شده، حاوی آدرس صفحه مربوطه‌به ویرایش مطلب به‌همراه شماره مطلب است که به‌روش Get برای این صفحه ارسال شده‌است. از آنجا که قسمت ?id=2 که به‌جای عدد ۲ در آن، شماره مطلب مربوطه قرار می‌گیرد، برحسب مطالب مختلف، متفاوت خواهد بود و ما فقط می‌خواهیم از این‌بابت مطمئن‌شویم که صفحه article_edit.php مراجعه‌کننده باشد و به‌قسمت متغیر کاری نداریم، از ابتدای آدرس مراجعه‌کننده، به‌اندازه طول ثابت URL بعلاوه ۲۸ کارکتر (به‌اندازه طول رشته /management/article_edit.php) جدا می‌کنیم و وجود آنرا در آرایه بررسی می‌نماییم. بدین‌ترتیب، قسمت متغیر نادیده گرفته خواهد شد. این راه‌حل درصورتی‌که مراجعه‌کننده صفحه، فایل article_new.php باشد نیز مشکلی ایجاد نمی‌کند، زیرا طول رشته /management/article_new.php برابر با ۲۷ کارکتر است و اگر از ابتدای رشته به‌اندازه ۲۸ کارکتر جدا کنیم، با پیغام خطا مواجه نخواهیم‌شد و کل رشته بعنوان خروجی برگردانده می‌شود. در خطوط 010 تا 014 نیز درصورت عدم‌وجود متغیر manager از نوع Session، پیغام خطای مناسب همراه با لینک بازگشت به صفحه اصلی ظاهر خواهد شد و درنتیجه، بقیه کد تنها درصورتی اجرا می‌شود که مدیر به‌درستی (با واردکردن نام کاربری و رمز عبور) وارد سایت شده‌باشد. در خط 025 ارسال صحیح پارامتر action به‌روش Get بررسی می‌شود و درنتیجه، عملیات موردنظر برروی مطالب تنها درصورت صحت ارسال این پارامتر انجام خواهد شد. در خط 026 یک شئ به‌نام $articles از کلاس Articles ایجاد می‌شود. در خط 027 توسط یک ساختار switch، مقدار این پارامتر پس‌از تبدیل به حروف کوچک، مورد بررسی قرار می‌گیرد. در خط 028 اگر پارامتر action برابر با new باشد، به‌معنای عمل درج مقاله جدید خواهد بود و لذا، ابتدا در خط 029 متغیر کنترلی $flag با مقدار اولیه true تعریف‌شده و در خط 030 متغیر $result با مقدار اولیه صفر ایجاد می‌شود. در زمان آپلود فایل در PHP، اگر فایل به‌درستی آپلود شده باشد (اندازه آن بیشتر از حد مجاز نباشد)، اتفاقات زیر بطور پیش‌فرض رخ می‌دهد:

- فایل با یک اسم موقت به پوشه tmp/ روی سرور منتقل می‌شود.
- در آرایه فوق‌سراسری $_FILES یک عنصر اضافه می‌شود که نام آن، برابر با نام عنصر input با خاصیت "type="file در فرم مبدأ خواهد بود.
- عنصر مذکور، به‌نوبه خود یک آرایه و حاوی عناصر زیر است:

error	اگر در زمان آپلود خطا رخ دهد، شماره خطا و در غیر اینصورت، عدد صفر در این عنصر قرار می‌گیرد
name	نام فایل اصلی روی کامپیوتر کاربر
size	اندازه فایل برحسب بایت
tmp_name	مسیر و نام موقت فایل آپلود شده روی سرور
type	نوع فایل آپلود شده

بنابراین، اگر عنصر $_FILES['file'] وجود نداشته‌باشد، به‌معنای عدم‌انتخاب یک فایل در فرم بوده‌است و همچنین اگر عنصر $_FILES['file']['error'] مخالف صفر باشد، نشان‌دهنده وجود خطا در آپلود فایل می‌باشد. این دو مورد در شرط موجود در خط 031 بررسی می‌شوند و درصورت برقراری هرکدام از این حالت‌ها، متغیر کنترلی $flag برابر با false خواهد شد. در خطوط 034 تا 039 نیز ارسال صحیح پارامترهای title، abstract و body به‌روش Post بررسی می‌شود و درصورت عدم‌ارسال یا ارسال رشته خالی، متغیر کنترلی $flag باز هم برابر با false خواهد شد. در خط 040، ابتدا متغیر $flag بررسی می‌شود و اگر برابر با true باشد، تابع move_uploaded_file فراخوانی می‌شود. این تابع، دارای دو پارامتر است. پارامتر اول، مسیر و نام موقت فایل آپلود شده روی سرور است که طبیعتاً برای دسترسی به‌آن عنصر tmp_name فایل آپلود شده مورد استفاده قرار می‌گیرد. پارامتر دوم تابع فوق نیز مسیر و نام فایل آپلود شده را جهت انتقال مشخص می‌کند که در کد ما، باید فایل مربوطه با نام اصلی در پوشه files واقع‌در پوشه والد بخش مدیریت ذخیره‌شود. خروجی تابع move_uploaded_file درصورت انتقال موفقیت‌آمیز فایل، مقدار true و درصورت عدم‌موفقیت در انتقال فایل، مقدار false خواهد بود. دقت‌کنید که اگر فایل موقت را با استفاده از این تابع، به مسیر دلخواه منتقل نکنید، با پایان‌یافتن کد، بطور خودکار ازبین خواهد رفت و قابل دسترسی نخواهد بود. حال اگر خروجی این‌تابع نیز true باشد، در خط 041 با فراخوانی تابع Insert از شئ $articles و ارسال پارامترهای عنوان چکیده، ادامه مطلب و نام فایل، عمل ثبت این اطلاعات در پایگاه داده‌ها نیز انجام می‌شود و خروجی آن که تعداد رکوردهای تحت‌تأثیر خواهد بود، در متغیر $result قرار می‌گیرد. بنابراین، در خط 043 می‌توان برحسب مقدار این متغیر، پیغام ثبت موفقیت‌آمیز مطلب ($result مخالف صفر) یا عدم ثبت مطلب ($result برابر با صفر) را نمایش‌داد.

در خطوط 046 تا 057 عملیات مشابهی جهت کنترل صحت ارسال اطلاعات لازم برای ویرایش مطلب و همچنین آپلود فایل انجام می‌شود. در زمان ویرایش مطلب باید علاوه‌بر آپلود فایل جدید، فایل قبلی را نیز حذف‌کنیم. برای این‌کار، در خط 058 با فراخوانی تابع SelectRow از شئ $articles و ارسال شماره مطلب برای آن، مطلب مذکور از پایگاه داده‌ها استخراج‌شده و در متغیر $article (بدون s) قرار می‌گیرد. حال در خط 059 اگر این متغیر مخالف false باشد (دستور MySQL بدون خطا باشد) و همچنین تعداد رکوردهای موجود در آن از صفر بیشتر باشد (جدول مطالب در پایگاه داده‌ها خالی نباشد و مطلبی با آن شماره پیدا شود)، تنها رکورد موجود در متغیر $article توسط تابع mysql_fetch_assoc استخراج‌شده و به‌منظور صرفه‌جویی در مصرف حافظه، مجدداً در متغیر $article قرار می‌گیرد. نکته مهم در اینجا آن‌است که مقدار قبلی متغیر $article را با مقدار فعلی آن اشتباه نگیرید: مقدار قبلی، یک جدول حاوی یک رکورد بود و مقدار فعلی، یک رکورد است که می‌توانیم به فیلدهای آن دسترسی پیدا کنیم. شرط موجود در خط 061، ابتدا وجود فایلی را که نام آن در فیلد filename رکورد جاری ذخیره شده‌است، بررسی می‌کند و درصورت وجود این فایل در پوشه files واقع در پوشه والد، آنرا توسط خط 062 و به‌کمک تابع unlink حذف می‌کند. در خط 065 نیز تابع Update از شئ $articles فراخوانی می‌شود و تعداد رکوردهای تحت تأثیر آن در متغیر $result ذخیره می‌شود و در خط 067 نیز برحسب مقدار این متغیر، پیغام ویرایش موفقیت‌آمیز یا عدم ویرایش مطلب موردنظر، به مدیر اعلام می‌شود.

در خطوط 070 تا 074، ضمن بررسی صحت ارسال شماره مطلب ازطریق آدرس (روش Get) در خط 071 (شامل ارسال‌شدن، خالی‌نبودن و عددی‌بودن پارامتر id)، تابع Delete از شئ $articles در خط 072 فراخوانی می‌شود تا مطلب مذکور، حذف منطقی گردد (مطلب واقعاً حذف نمی‌شود و فقط فیلد status آن با مقدار deleted مقداردهی می‌شود) و در خط 074 برحسب مقدار متغیر $result، پیغام مناسب درخصوص حذف مطلب به مدیر اعلام می‌گردد.

در خطوط 077 تا 081، به روش مشابه، عمل بازیابی مطلب موردنظر با فراخوانی تابع Recover از شئ $articles انجام می‌شود.

صفحه درج نظر جدید management/comment_new.php

اگر مدیر سایت شخصاً تمایل به درج نظر جدیدی داشته‌باشد، می‌تواند از این‌قسمت استفاده‌کند. کد این فایل به‌صورت زیر است:

```
001. <?php
002.     //Copyright محمد مصطفي شهركي @ http://www.ncis.ir
003.     require_once '../config.php';
004.     require_once '../db.php';
005.     if(!isset($_SERVER['HTTP_REFERER']) || $_SERVER['HTTP_REFERER'] != URL.'/management/links.php') {
006.         header('location: main.php');
007.         exit();
008.     }
009.     session_start();
010.     if(!isset($_SESSION['manager']) || $_SESSION['manager'] != true) {
011.         echo 'Illegal Access - <a href="'.URL.'" target="_top">Go to main page</a>'."\n";
012.         exit();
013.     }
014. ?>
015. <!doctype html>
016. <html dir="rtl">
017. <head>
018. <meta charset="utf-8"/>
019. <title><?php echo TITLE; ?></title>
020. <link href="../style.css" rel="stylesheet" type="text/css"/>
021. </head>
022. <body style="background-image: url(../robot.jpg); background-repeat: no-repeat;">
023. <form action="comment_result.php?action=new" method="post">
024. <table border="0px" cellpadding="0px" cellspacing="2px" width="500px">
025. <tr align="right" valign="middle">
026. <th width="100px"><label for="aid">مطلب</label></th>
027. <td>
028. <select class="transparent" id="aid" name="aid" style="width: 100%;">
029. <?php
030.     $articles = new Articles();
031.     $result = $articles->SelectAll();
032.     if($result !== false && mysql_num_rows($result) > 0) {
033.         while($row = mysql_fetch_assoc($result)) {
034.             echo '<option value="'.$row['id'].'">'.$row['id'].' - '.$row['title']
                      .'</option>'."\n";
035.         }
036.     }
037. ?>
038. </select>
039. </td>
040. </tr>
041. <tr align="right" valign="middle">
042. <th><label for="name">نام</label></th>
043. <td><input class="transparent" id="name" maxlength="255" name="name" style="width: 100%;"
     type="text"/></td>
044. </tr>
045. <tr align="right" valign="middle">
046. <th><label for="body">متن</label></th>
047. <td><textarea class="transparent" id="body" name="body" rows="5" style="width: 100%;"></textarea></td>
048. </tr>
049. <tr align="right" valign="middle">
```

```
<th><label for="status">حالت</label></th>
<td>
<label for="public">عمومي </label>
<input checked="checked" class="transparent" id="public" name="status" type="radio" value="public"/>
 
<label for="private">خصوصي </label>
<input class="transparent" id="private" name="status" type="radio" value="private"/>
</td>
</tr>
<tr align="right" valign="middle">
<td colspan="2"><input style="width: 100%;" type="submit" value="ثبت"/></td>
</tr>
</table>
</form>
</body>
</html>
```

تغییرات کد نسبت به جلسه قبل:

- حذف دستورات ob_start و ob_end_flush از ابتدا و انتهای کد به‌دلیل عدم ضرورت
- تبدیل اسامی تگ‌ها به حروف کوچک
- مرتب‌شدن خاصیت‌های تگ‌ها به‌ترتیب الفبا
- حذف پرانتزهای دستور require_once در خطوط 003 و 004
- اضافه‌شدن خطوط 005 تا 008 جهت جلوگیری از دسترسی مستقیم به فایل و ایجاد اجبار جهت بازشدن ازطریق لینک صفحه links.php
- هدایت کاربر به صفحه main.php به‌جای management.php در خط 006
- تکمیل شرط خط 010 جهت امنیت بیشتر
- تغییر نحوه هدایت کاربر به صفحه اصلی جهت رفع اشکالات احتمالی
- اضافه‌شدن تگ doctype در خط 015
- کوتاه‌شدن تگ meta در خط 018
- جایگزینی متغیر $title با ثابت TITLE در خط 019
- تغییر روش تعیین تصویر پس‌زمینه به ساختار CSS در خط 022 جهت رعایت استانداردهای جدید HTML
- اضافه‌شدن عبارت px به اعداد جهت رعایت استاندارد HTML در تعیین اندازه عناصر برحسب پیکسل
- حذف خاصیت width تگ td در خط 027 به‌دلیل عدم ضرورت

توضیح کد:

خطوط 001 تا 014 این کد، دقیقاً مشابه فایل article_new.php هستند و لذا، از توضیح مجدد آن خودداری می‌کنیم. از آنجا که در صفحه درج نظر جدید که در اختیار مدیر قرار دارد (صفحه جاری)، امکان درج نظر برای تمامی مطالب موجود برای مدیر درنظر گرفته شده‌است، باید فهرست مناسبی از مطالب را در اختیار مدیر قرار دهیم. برای این‌کار، از یک فهرست انتخاب (عنصر select) استفاده کرده‌ایم و مطالب را درون آن (توسط تگ option) قرار داده‌ایم تا مدیر بتواند یکی‌از آنها را برحسب تمایل، انتخاب‌کند. به‌منظور استخراج مطالب، در خط 030 یک شئ به‌نام $articles از کلاس Articles ایجاد می‌شود و در خط 031 تمامی مطالب موجود با فراخوانی تابع SelectAll شئ $articles استخراج‌شده و در متغیر $result قرار می‌گیرد. در خط 032 صحت دستور MySLQ و همچنین خالی‌نبودن جدول بررسی می‌شود و درصورت برقراری شرایط فوق، در خطوط 033 تا 035 به‌کمک یک حلقه while، تمامی رکوردهای موجود در متغیر $result به‌ترتیب استخراج و توسط تگ option در داخل تگ select درج می‌شوند. خاصیت value هر عنصر، شماره مطلب مربوطه و محتوای آن (که مدیر مشاهده می‌کند)، شماره مطلب بعلاوه عنوان آن می‌باشد که با یک خط تیره‌جدا شده‌اند. مقصد فرم موجود در این فایل، صفحه comment_result.php است که علاوه‌بر ارسال اطلاعات دریافت‌شده از کاربر به روش Post، پارامتر action را نیز با مقدار new به‌روش Get برای صفحه مذکور ارسال می‌کند. بقیه کد این فایل نیز HTML ساده است و نیاز به توضیح ندارد. از آنجا که فایل comment_result.php اعمال دیگری بجز درج نظر جدید نیز انجام می‌دهد، توضیح آنرا بعد از صفحه مدیریت نظرات ارائه خواهیم‌کرد.

صفحه مدیریت نظرات management/comment_edit.php

این صفحه، دو وظیفه کلی برعهده دارد: ۱- نمایش فهرستی از تمامی نظرات ثبت‌شده همراه‌با لینک‌هایی جهت ویرایش، تأیید/عدم‌تأیید و تغییر حالت نظر به خصوصی/عمومی و ۲- نمایش یک فرم جهت ویرایش اطلاعات مربوط‌به نظر شامل شماره مطلب مرتبط، نام فرد نظردهنده، متن نظر و نوع آن (خصوصی/عمومی). ابتدا کد فایل را ملاحظه‌کنید:

```
<?php
    //Copyright محمد مصطفي شهركي @ http://www.ncis.ir
    require_once '../config.php';
    require_once '../db.php';
    $referers = array(URL.'/management/comment_edit.php', URL.'/management/links.php');
    if(!isset($_SERVER['HTTP_REFERER']) || !in_array($_SERVER['HTTP_REFERER'], $referers)) {
        header('location: main.php');
        exit();
    }
```

```
010.     session_start();
011.     if(!isset($_SESSION['manager']) || $_SESSION['manager'] != true) {
012.         echo 'Illegal Access - <a href="'.URL.'" target="_top">Go to main page</a>'."\n";
013.         exit();
014.     }
015. ?>
016. <!doctype html>
017. <html dir="rtl">
018. <head>
019. <meta charset="utf-8"/>
020. <title><?php echo TITLE; ?></title>
021. <link href="../style.css" rel="stylesheet" type="text/css"/>
022. </head>
023. <body style="background-image: url(../robot.jpg); background-repeat: no-repeat;">
024. <?php
025.     $articles = new Articles();
026.     $comments = new Comments();
027.     if(!isset($_GET['id']) || $_GET['id'] == '') {
028.         $result = $comments->SelectAll();
029.         if($result !== false && mysql_num_rows($result) > 0) {
030. ?>
031. <table border="1px" cellpadding="2px" cellspacing="0px" width="100%">
032. <tr align="center" valign="middle">
033. <th width="5%">ردیف</th>
034. <th width="15%">مطلب</th>
035. <th width="15%">نام</th>
036. <th width="25%">متن</th>
037. <th width="10%">وضعیت</th>
038. <th width="10%">حالت</th>
039. <th width="20%">عملیات</th>
040. </tr>
041. <?php
042.             while($row = mysql_fetch_assoc($result)) {
043.                 echo '<tr align="center" valign="middle">'."\n";
044.                 echo '<td>'.($row['id'] != '' ? $row['id'] : ' ').'</td>'."\n";
045.                 $article = $articles->SelectRow($row['aid']);
046.                 $article = ($article !== false && mysql_num_rows($article) > 0) ?
                                mysql_fetch_assoc($article) : array('id' => '0', 'title' => 'نامشخص');
047.                 echo '<td style="text-align: justify">'.$article['id'].' - '
                        .$article['title'].'</td>'."\n";
048.                 echo '<td style="text-align: justify;">'.($row['name'] != '' ? $row['name'] :
                        ' ').'</td>'."\n";
049.                 echo '<td style="text-align: justify;">'.($row['body'] != '' ? $row['body'] :
                        ' ').'</td>'."\n";
050.                 echo '<td>'.($row['confirmed'] != '' ? ($row['confirmed'] == '0' ? 'تأیید منتظر' :
                        'تأیید شده') : 'نامشخص').'</td>'."\n";
051.                 echo '<td>'.($row['status'] != '' ? ($row['status'] == 'public' ? 'عمومي' : 'خصوصي') :
                        'نامشخص').'</td>'."\n";
052.                 echo '<td>';
053.                 echo '<a href="comment_edit.php?id='.$row['id'].'">[ویرایش]</a>';
054.                 echo '   ';
055.                 echo '<a href="comment_result.php?action='.($row['confirmed'] == '0' ? 'confirmed' :
                        'waiting').'&id='.$row['id'].'">['.($row['confirmed'] == '0' ? 'تأیید' :
                        'عدم تأیید').']</a>';
056.                 echo '   ';
057.                 echo '<a href="comment_result.php?action='.($row['status'] == 'public' ? 'private' :
                        'public').'&id='.$row['id'].'">['.($row['status'] == 'public' ? 'خصوصي' :
                        'عمومي').']</a>';
058.                 echo '</td>'."\n";
059.                 echo '</tr>'."\n";
060.             }
061. ?>
062. </table>
063. <?php
064.         }
065.         else {
066.             echo 'هیچ نظري پیدا نشد.<br/>'."\n";
067.         }
068.     }
069.     else {
070.         $result = $comments->SelectRow($_GET['id']);
071.         if($result !== false && mysql_num_rows($result) > 0) {
072.             $row = mysql_fetch_assoc($result);
073. ?>
074. <form action="comment_result.php?action=edit" method="post">
075. <input name="id" type="hidden" value="<?php echo $row['id']; ?>"/>
076. <table border="0px" cellpadding="0px" cellspacing="2px" width="500px">
077. <tr align="right" valign="middle">
078. <th width="100px"><label for="aid">مطلب</label></th>
```

```
079. <td>
080. <select class="transparent" id="aid" name="aid" style="width:100%">
081. <?php
082.             $articles = $articles->SelectAll();
083.             if($articles !== false && mysql_num_rows($articles) > 0) {
084.                 while($article = mysql_fetch_assoc($articles)) {
085.                     echo '<option value="'.$article['id'].'" '.($article['id'] == $row['aid'] ?
                            'selected="selected"' : '').'>'.$article['id'].' - '
                            .$article['title'].'</option>'."\n";
086.                 }
087.             }
088. ?>
089. </select>
090. </td>
091. </tr>
092. <tr align="right" valign="middle">
093. <th><label for="name">نـام</label></th>
094. <td><input class="transparent" id="name" name="name" maxlength="255" style="width: 100%;" type="text"
     value="<?php echo $row['name']; ?>"/></td>
095. </tr>
096. <tr align="right" valign="middle">
097. <th><label for="body">مـتن</label></th>
098. <td><textarea class="transparent" id="body" name="body" rows="5" style="width: 100%;">
     <?php echo $row['body']; ?></textarea></td>
099. </tr>
100. <tr align="right" valign="middle">
101. <th><label for="status">حـالـت</label></th>
102. <td>
103. <label for="public">عمومي </label>
104. <input <?php echo $row['status'] == 'public' ? 'checked="checked"' : ''; ?> class="transparent" id="public"
     name="status" type="radio" value="public"/>
105.  
106. <label for="private">خصوصي </label>
107. <input <?php echo $row['status'] != 'public' ? 'checked="checked"' : ''; ?> class="transparent"
     id="private" name="status" type="radio" value="private"/>
108. </td>
109. </tr>
110. <tr align="right" valign="middle">
111. <td colspan="2"><input style="width: 100%;" type="submit" value="ثـبت"/></td>
112. </tr>
113. </table>
114. </form>
115. <?php
116.         }
117.     }
118. ?>
119. </body>
120. </html>
```

تغییرات کد نسبت به جلسه قبل:

- حذف دستورات ob_start و ob_end_flush از ابتدا و انتهای کد به‌دلیل عدم ضرورت
- تبدیل اسامی تگ‌ها به حروف کوچک
- مرتب‌شدن خاصیت‌های تگ‌ها به‌ترتیب الفبا
- حذف پرانتزهای دستور require_once در خطوط 003 و 004
- اضافه‌شدن خطوط 005 تا 009 جهت جلوگیری از دسترسی مستقیم و ایجاد اجبار جهت بازشدن ازطریق صفحات links.php، و comment_edit.php
- هدایت کاربر به صفحه main.php به‌جای management.php در خط 007
- تکمیل شرط خط 011 جهت امنیت بیشتر
- تغییر نحوه هدایت کاربر به صفحه اصلی جهت رفع اشکالات احتمالی
- اضافه‌شدن تگ doctype در خط 016
- کوتاه‌شدن تگ meta در خط 019
- جایگزینی متغیر $title با ثابت TITLE در خط 020
- تغییر روش تعیین تصویر پس‌زمینه به ساختار CSS در خط 023 جهت رعایت استانداردهای جدید HTML
- تکمیل شرط‌های موجود در خطوط 029 و 083 جهت امنیت بیشتر و عدم نمایش پیغام خطای احتمالی درصورت بروز خطا در دستورات MySQL
- اضافه‌شدن عبارت px به اعداد جهت رعایت استاندارد HTML در تعیین اندازه عناصر برحسب پیکسل
- حذف خاصیت width تگ td در خط 079 به‌دلیل عدم ضرورت

توضیح کد:

ابتدا در خطوط 025 و 026 دو شئ به نام‌های $articles و $comments به‌ترتیب از کلاس‌های Articles و Comments ایجاد می‌شود. در این فایل، اگر شماره نظر ازطریق آدرس (روش Get) برای صفحه ارسال شده‌باشد، فقط همان نظر جهت ویرایش درقالب یک فرم ظاهر می‌شود و درغیر این‌صورت، تمامی نظرات در یک جدول همراه‌با لینک‌هایی جهت ویرایش، تأیید/عدم‌تأیید و تغییر حالت به خصوصی/عمومی در اختیار مدیر قرار می‌گیرد. در خط 027 ارسال‌شدن پارامتر id به‌روش Get بررسی می‌شود و درصورت عدم‌ارسال یا خالی‌بودن پارامتر ارسال‌شده، باید جدول حاوی کلیه نظرات ثبت‌شده به‌نمایش درآید. بدین‌منظور، در خط 028 با فراخوانی تابع SelectAll از شئ $comments، تمامی نظرات موجود در پایگاه داده‌ها استخراج‌شده و در متغیر $result قرار می‌گیرد. در خط 029 ابتدا عدم‌وجود خطا در دستور MySQL اجرا شده‌در تابع SelectAll و سپس، خالی‌نبودن جدول نظرات در پایگاه داده‌ها بررسی می‌شود و درصورت برقراری شرایط فوق، در خطوط 031 تا 040 یک جدول همراه‌با یک سطر حاوی عنوان ستون‌ها ایجاد می‌شود، اما جدول (تگ table) بسته نمی‌شود زیرا می‌خواهیم نظرات را استخراج‌کرده و درقالب سطرهای مختلف به آن اضافه‌کنیم. در خط 042 به‌کمک حلقه while، تمامی رکوردهای موجود در $result به‌ترتیب استخراج‌شده و در متغیر $row قرار می‌گیرند. در داخل حلقه، ابتدا در خط 043 یک سطر جدید به انتهای جدول اضافه می‌شود. در خط 044 شماره ردیف نظر استخراج‌شده (که در متغیر $row قرار دارد) به سطر جدید اضافه می‌شود. در خط 045 با فراخوانی تابع SelectRow از شئ $articles و ارسال شماره مطلب مربوط به نظر (فیلد aid) برای آن، مطلب مربوطه از پایگاه داده‌ها استخراج‌شده و در متغیر $article (بدون s) ذخیره می‌شود. به خط 046 دقت کنید: در این دستور، اگر خطایی در دستور MySQL نداشته‌باشیم و ضمناً جدول مطالب خالی نباشد، تنها رکورد موجود در متغیر $article استخراج‌شده و مجدداً درون متغیر $article ذخیره می‌شود، اما اگر شرایط فوق برقرار نباشد، یک آرایه با دو عنصر id و title به‌ترتیب با مقادیر صفر و «نامشخص» ایجاد می‌شود تا در متغیر $article قرارگیرد (یعنی مطلبی با شماره مشخص‌شده توسط فیلد aid نظر، پیدا نشده‌است). در خط 047، ابتدا فیلد id آرایه $article (یعنی شماره مطلب مربوطه) و سپس، یک خط تیره و درنهایت فیلد title متغیر $article (یعنی عنوان مطلب مربوطه) به‌صورت یک خانه به سطر جدید جدول اضافه می‌شود. در خط 048 نام نظردهنده و در خط 049 متن نظر در خانه‌های جداگانه به سطر موجود اضافه می‌شوند. در خط 050، اگر فیلد confirmed خالی نباشد، درصورت صفر بودن مقدار این فیلد، عبارت «منتظر تأیید» و درغیر این‌صورت عبارت «تأیید شده» و درصورت خالی‌بودن فیلد confirmed، عبارت «نامشخص» بعنوان خانه بعدی جدول، به سطر موجود افزوده می‌شود. خط 051 نیز اگر فیلد status خالی باشد، عبارت «نامشخص» و درصورت خالی‌نبودن، برحسب مقدار این فیلد، عبارت «عمومی» (مقدار public) یا «خصوصی» (مقدار private) را بعنوان خانه بعدی، به سطر موجود اضافه می‌کند. در خط 052 یک خانه جدید باز می‌شود. در خط 053 یک لینک جهت ویرایش مطلب در خانه بازشده‌درج می‌شود که مدیر را به صفحه comment_result.php هدایت‌کرده و شماره (فیلد id) نظر را نیز به‌روش Get برای آن‌صفحه می‌فرستد. در خط 054 فاصله کافی جهت خوانایی بهتر لینک‌ها، ایجاد می‌شود. لینک بعدی که توسط خط 055 ایجاد می‌شود، مربوط‌به تأیید/عدم‌تأیید نظر است که باز هم مدیر را به صفحه comment_result.php منتقل می‌کند، اما پارامتر action آن که به‌روش Get برای صفحه مذکور می‌فرستد و همچنین محتوا (متن) لینک، بستگی به مقدار فیلد confirmed دارد: اگر این‌مقدار صفر باشد، باید عمل «تأیید» با پارامتر action برابر با confirmed و درغیر این‌صورت، عمل «عدم‌تأیید» با پارامتر action برابر با waiting انجام‌شود. درخط 056 مجدداً فاصله کافی با لینک بعدی ایجاد می‌شود. در خط 057 به‌روش مشابه خط 055 لینک تبدیل مطلب به «خصوصی» (پارامتر action برابر با private) و یا «عمومی» (پارامتر action برابر با public) به‌ترتیب درصورت برابر بودن/نبودن فیلد status با مقدار private تولید می‌شود. ضمناً در هر دو خط 055 و 057 مشابه‌خط 053، شماره نظر (فیلد id) نیز به‌روش Get برای صفحه comment_result.php ارسال خواهد شد. در خطوط 058 و 059 نیز به‌ترتیب آخرین‌خانه سطر و خود سطر، بسته می‌شوند. همان‌گونه که ذکر شد، حلقه فوق تا زمانی که تمامی نظرات در جدول موجود درج‌شوند، تکرار خواهد شد و بعد از پایان حلقه، توسط خط 062 جدول ایجادشده بسته می‌شود. در قسمت else موجود در خطوط 065 تا 067، درصورت وجود خطا در دستور MySQL یا خالی‌بودن جدول، پیغام «هیچ نظری یافت نشد.» به مدیر نشان‌داده خواهد شد. قسمت else خط 069 درصورتی اجرا می‌شود که پارامتر id ازطریق آدرس (روش Get) برای این‌صفحه فرستاده شده‌باشد. در خط 071 عدم‌وجود خطا در دستور MySQL و خالی‌نبودن جدول نظرات بررسی می‌شود و درصورت برقراری شرایط فوق، در خط 072 تنها رکورد موجود در متغیر $result استخراج‌شده و در متغیر $row قرار می‌گیرد. سپس در خط 074 یک فرم با مقصد comment_result.php ایجاد می‌شود که علاوه‌بر ارسال اطلاعات دریافت‌شده از کاربر به روش Post، پارامتر action را نیز با مقدار edit برای صفحه مذکور می‌فرستد. در خط 075 توسط یک تگ input مخفی (خاصیت type برابر با hidden)، شماره نظر (فیلد id) به فرم اضافه می‌شود تا همراه با سایر اطلاعات، به صفحه مقصد فرم ارسال گردد. در ادامه، در خطوط 082 تا 087، با فراخوانی تابع SelectAll شئ $articles و به‌کمک یک حلقه while، تمامی مقالات درون تگ select که توسط خط 080 بازشده و در خط 089 بسته می‌شود، به‌نمایش در می‌آیند و مقاله‌ای که شماره آن در فیلد aid نظر ثبت شده‌است، از قبل انتخاب می‌شود (خاصیت selected="selected" برای آن تعیین می‌شود). در خطوط 104 و 107 نیز گزینه‌های عمومی/خصوصی ایجاد می‌شوند و درصورت برابربودن فیلد status نظر با public، گزینه اول (عمومی) و درغیر این‌صورت، گزینه دوم (خصوصی) انتخاب می‌شود. سایر خطوط این کد نیز HTML ساده بوده و نیاز به توضیح خاصی ندارد.

صفحه مقصد عملیات برروی نظرات management/comment_result.php

این صفحه، بیشترین اهمیت را در بین صفحات مرتبط با نظرات دارد، زیرا کلیه اعمال شامل درج نظر جدید، ویرایش نظر موجود، تأیید/عدم‌تأیید و همچنین تعیین نوع نظر (عمومی/خصوصی) برعهده این فایل است. درواقع سایر فایل‌ها صرفاً وظیفه دریافت اطلاعات موردنیاز را از مدیر دارند و این فایل، عملیات اصلی را برحسب نوع آن‌که توسط پارامتر action ازطریق آدرس (به‌روش Get) برای فایل ارسال شده‌است، انجام می‌دهد. بنابراین، بهتر است به‌دقت کد این فایل را با دقت بیشتری مطالعه و بررسی نمایید:

```
001. <?php
002.     //Copyright محمدمصطفی‌بشهرکی@ http://www.ncis.ir
003.     require_once '../config.php';
004.     require_once '../db.php';
005.     $referers = array(URL.'/management/comment_edit.php', URL.'/management/comment_new.php');
```

```
006.     if(!isset($_SERVER['HTTP_REFERER']) || !in_array(substr($_SERVER['HTTP_REFERER'], 0, strlen(URL) + 28),
            $referers)) {
007.         header('location: main.php');
008.         exit();
009.     }
010.     session_start();
011.     if(!isset($_SESSION['manager']) || $_SESSION['manager'] != true) {
012.         echo 'Illegal Access - <a href="'.URL.'" target="_top">Go to main page</a>'."\n";
013.         exit();
014.     }
015. ?>
016. <!doctype html>
017. <html dir="rtl">
018. <head>
019. <meta charset="utf-8"/>
020. <title><?php echo TITLE; ?></title>
021. <link href="../style.css" rel="stylesheet" type="text/css"/>
022. </head>
023. <body style="background-image: url(../robot.jpg); background-repeat: no-repeat;">
024. <?php
025.     if(isset($_GET['action']) && $_GET['action'] != '') {
026.         $comments = new Comments();
027.         switch(strtolower($_GET['action'])) {
028.             case 'new':
029.                 $flag = true;
030.                 $result = 0;
031.                 $vars = array('body', 'id', 'name', 'status');
032.                 foreach($vars as $var) {
033.                     if(!isset($_POST[$var]) || $_POST[$var] == '') {
034.                         $flag = false;
035.                     }
036.                 }
037.                 if($flag) {
038.                     $result = $comments->Insert($_POST['aid'], $_POST['name'], $_POST['body'],
                                                    $_POST['status']);
039.                 }
040.                 echo 'نظر'.($result > 0 ? 'با موفقیت ثبت شد' : 'ثبت نشد').'.<br/>'."\n";
041.                 break;
042.             case 'edit':
043.                 $flag = true;
044.                 $result = 0;
045.                 $vars = array('aid', 'body', 'id', 'status');
046.                 foreach($vars as $var) {
047.                     if(!isset($_POST[$var]) || $_POST[$var] == '') {
048.                         $flag = false;
049.                     }
050.                 }
051.                 if($flag) {
052.                     $result = $comments->Update($_POST['id'], $_POST['aid'], $_POST['body'],
                                                    $_POST['status']);
053.                 }
054.                 echo 'نظر'.($result > 0 ? 'با موفقیت ویرایش شد' : 'ویرایش نشد').'.<br/>'."\n";
055.                 break;
056.             case 'confirmed':
057.                 $result = 0;
058.                 if(isset($_GET['id']) && $_GET['id'] != '' && is_numeric($_GET['id'])) {
059.                     $result = $comments->Set($_GET['id'], 'confirmed');
060.                 }
061.                 echo 'نظر'.($result > 0 ? 'با موفقیت تأیید شد' : 'تأیید نشد').'.<br/>'."\n";
062.                 break;
063.             case 'waiting':
064.                 $result = 0;
065.                 if(isset($_GET['id']) && $_GET['id'] != '' && is_numeric($_GET['id'])) {
066.                     $result = $comments->Set($_GET['id'], 'waiting');
067.                 }
068.                 echo 'نظر'.($result > 0 ? 'با موفقیت منتظر تأیید شد' : 'منتظر تأیید نشد').'.<br/>'."\n";
069.                 break;
070.             case 'private':
071.                 $result = 0;
072.                 if(isset($_GET['id']) && $_GET['id'] != '' && is_numeric($_GET['id'])) {
073.                     $result = $comments->Set($_GET['id'], 'private');
074.                 }
075.                 echo 'نظر'.($result > 0 ? 'با موفقیت خصوصی شد' : 'خصوصی نشد').'.<br/>'."\n";
076.                 break;
077.             case 'public':
078.                 $result = 0;
079.                 if(isset($_GET['id']) && $_GET['id'] != '' && is_numeric($_GET['id'])) {
080.                     $result = $comments->Set($_GET['id'], 'public');
081.                 }
```

```
082.                 echo 'نظر'.($result > 0 ? 'با موفقیت عمومی شد' : 'عمومی نشد').'.<br/>'."\n";
083.                 break;
084.         }
085.     }
086. ?>
087. </body>
088. </html>
```

تغییرات کد نسبت به جلسه قبل:

- حذف دستورات ob_start و ob_end_flush از ابتدا و انتهای کد به‌دلیل عدم ضرورت
- تبدیل اسامی تگ‌ها به حروف کوچک
- مرتب‌شدن خاصیت‌های تگ‌ها به‌ترتیب الفبا
- حذف پرانتزهای دستور require_once در خطوط 003 و 004
- اضافه‌شدن خطوط 005 تا 009 جهت جلوگیری از دسترسی مستقیم و ایجاد اجبار جهت بازشدن ازطریق صفحات article_edit.php، و article_new.php
- هدایت کاربر به صفحه main.php به‌جای management.php در خط 007
- تکمیل شرط خط 011 جهت امنیت بیشتر
- تغییر نحوه هدایت کاربر به صفحه اصلی جهت رفع اشکالات احتمالی
- اضافه‌شدن تگ doctype در خط 016
- کوتاه‌شدن تگ meta در خط 019
- جایگزینی متغیر $title با ثابت TITLE در خط 020
- تغییر روش تعیین تصویر پس‌زمینه به ساختار CSS در خط 023 جهت رعایت استانداردهای جدید HTML
- اضافه‌شدن شرط موجود در خط 025 جهت جلوگیری از بروز خطا درصورت عدم ارسال پارامتر action به روش Get برای صفحه
- اضافه‌شدن متغیر کنترلی $flag در خطوط 029 و 043 با مقدار اولیه true جهت بررسی ارسال صحیح تمامی پارامترهای لازم برای انجام عملیات روی مطالب
- اضافه‌شدن خطوط 030 تا 037 جهت کنترل صحت ارسال پارامترهای لازم در زمان ایجاد نظر جدید
- اضافه‌شدن خطوط 044 تا 051 جهت کنترل صحت ارسال پارامترهای لازم در زمان ویرایش نظر موجود
- اضافه‌شدن خطوط 057 و 058 جهت کنترل صحت ارسال شماره نظر به روش Get برای صفحه در زمان تأیید نظر
- اضافه‌شدن خطوط 064 و 065 جهت کنترل صحت ارسال شماره نظر به روش Get برای صفحه در زمان عدم‌تأیید نظر
- اضافه‌شدن خطوط 071 و 072 جهت کنترل صحت ارسال شماره نظر به روش Get برای صفحه در زمان تغییر نوع نظر به خصوصی
- اضافه‌شدن خطوط 078 و 079 جهت کنترل صحت ارسال شماره نظر به روش Get برای صفحه در زمان تغییر نوع نظر به عمومی

توضیح کد:

از آنجا که اکثر کدهای این فایل، مشابه فایل article_result.php است، تنها به ذکر توضیح کدهای متفاوت اکتفا می‌کنیم (برای مشاهده توضیحات سایر کدها به بخش توضیح کد فایل مذکور مراجعه نمایید). در دستورات موجود در خطوط 005 تا 009، درصورتی‌که کاربر به‌صورت مستقیم (تایپ آدرس در مرورگر) یا ازطریق لینکی درصفحاتی به‌جز comment_edit.php یا comment_new.php وارد صفحه شده‌باشد، وی را به صفحه اصلی بخش مدیریت هدایت می‌کند. در خطوط 010 تا 014 نیز کاربر درصورت دسترسی غیرمجاز (عدم ورود به سایت با نام کاربری و رمز عبور)، با یک پیغام خطا و لینکی که وی را به صفحه اصلی سایت باز می‌گرداند، مواجه خواهد شد.در خط 025 ارسال صحیح پارامتر action به‌روش Get مورد بررسی قرار می‌گیرد. در خط 026 یک شئ به‌نام $comments از کلاس Comments ایجاد می‌شود.در خط 027 توسط یک ساختار switch، مقدار پارامتر action (البته بعد از تبدیل به حروف کوچک) مورد ارزیابی قرار می‌گیرد. اگر این پارامتر برابر با new باشد، خطوط 029 تا 041 اجرا می‌شود که با روشی مشابه فایل article_result.php، اقدام ثبت نظر جدید درصورت صحت ارسال اطلاعات لازم می‌نماید. اگر پارامتر فوق برابر با edit باشد،خطوط 043 تا 055 اجرا می‌شود که بعد از بررسی صحت ارسال اطلاعات موردنیاز، اقدام به ویرایش نظر موجود می‌کند. درصورتی‌که پارامتر action برابر با confirmed باشد، درصورت تأیید صحت ارسال پارامتر id به روش Get، در خط 059 به‌کمک تابع Set از شئ $comments و ارسال پارامتر دوم برابر با confirmed، عمل تأیید نظر مربوطه انجام می‌شود. به‌روش مشابه، اگر پارامتر action برابر با waiting، private یا public باشد، به‌ترتیب در خطوط 066، 073 و 080 عمل عدم‌تأیید،خصوصی‌سازی یا عمومی‌سازی نظر مربوطه با فراخوانی تابع Set و ارسال پارامتر دوم برابر با عمل موردنظر، انجام می‌شود.

فایل خروج از بخش مدیریت management/logout.php

وظیفه این فایل، خارج‌کردن مدیر از سایت است. کد این فایل، بسیار ساده و به‌شرح زیر است:

```
001. <?php
002.     //Copyright محمدمصطفی‌شهرکی@ http://www.ncis.ir
003.     session_start();
004.     require_once '../config.php';
005. ?>
006. <!doctype html>
007. <html dir="rtl">
```

```
008. <head>
009. <meta charset="utf-8"/>
010. <title><?php echo TITLE; ?></title>
011. <link href="../style.css" rel="stylesheet" type="text/css"/>
012. </head>
013. <body style="background-image: url(../robot.jpg); background-repeat: no-repeat;">
014. <?php
015.     if(isset($_SESSION['manager'])) {
016.         unset($_SESSION['manager']);
017.     }
018.     header('location: '.URL);
019.     exit();
020. ?>
021. </body>
022. </html>
```

تغییرات کد نسبت به جلسه قبل:

- حذف دستورات ob_start و ob_end_flush از ابتدا و انتهای کد به‌دلیل عدم ضرورت
- تبدیل اسامی تگ‌ها به حروف کوچک
- مرتب‌شدن خاصیت‌های تگ‌ها به‌ترتیب الفبا
- حذف پرانتزهای دستور require_once
- اضافه‌شدن تگ doctype در خط 006
- کوتاه‌شدن تگ meta در خط 009
- جایگزینی متغیر $title با ثابت TITLE در خط 010
- تغییر روش تعیین تصویر پس‌زمینه به ساختار CSS در خط 013 جهت رعایت استانداردهای جدید HTML
- اضافه‌شدن شرط موجود در خط 015 جهت جلوگیری از نمایش پیغام خطای احتمالی درصورت عدم تعریف متغیر manager از نوع Session

توضیح کد:

این فایل ابتدا در خط 015 تعریف‌شدن متغیر manager از نوع Session بررسی می‌شود و درصورت مثبت‌بودن (ورود مدیر با نام کاربری و رمز عبور)،درخط 016 باحذف متغیر manager از نوع Session (درصورت ایجادشدن)، موجب خروج مدیر از سایت می‌شود و سپس، در خط 018 وی را به صفحه اصلی سایت هدایت می‌کند.

توضیح پایانی این جلسه

همان‌طور که مشخص‌است، این کد هنوز هم نیازمند بهینه‌سازی بیشتر بوده و اصول بنیادی برنامه‌نویسی وب‌سایت در آن به‌طور کامل رعایت نشده است؛ اما از آنجاکه‌قصد مادر این جلسات، آموزش مرحله به مرحله است، به‌مرور این نواقص را برطرف خواهیم کرد. یکی‌دیگر از ایراداتی که در این سایت دیده می‌شود (و در این جلسه اضافه شده‌است)، به‌هم‌ریختگی نسبی جداول است که علت آن، استفاده از تگ <!doctype html> است که برای معرفی استاندارد به‌کاررفته در اسناد به مرورگر کاربرد دارد و در اینجا، معرفی‌کننده HTML5 می‌باشد. از آنجا که در استاندارد جدید، وظیفه تعیین نحوه نمایش عناصر برعهده CSS است، طراحی مبتنی بر جدول تقریباً منسوخ شده‌است و درنتیجه، نمایش مناسبی با استفاده از جداول به‌دست نخواهد آمد که در جلسات آینده، نحوه نمایش عناصر را کاملاً با کمک CSS و طراحی Table-Less (بدون جدول) بازنویسی خواهیم‌کرد.

برنامه‌نویسی شئ‌گرا دربرابر برنامه‌نویسی رویه‌گرا

روش معمول در برنامه‌نویسی، رَویه‌گرا (**Procedural**) نام‌دارد. در این روش برنامه‌نویسی، برنامه از یک‌سری کدهای معمولی و کدهایی که درون تابع (یا رَویه) قرار دارند، تشکیل می‌شود. در زمان اجرا نیز کد برنامه از ابتدا تا انتها به‌ترتیب و به‌صورت خط‌به‌خط اجرا می‌شود و هرگاه یک تابع فراخوانی گردد، کنترل برنامه به آن‌قسمت منتقل می‌شود و بعد از اجرای آن تابع، مجدداً به بخش فراخواننده بازگشته و برنامه از قسمت بعد از فراخوانی تابع، ادامه می‌یابد. حال در مرحله فراخوانی، امکان ارسال یک‌سری مقادیر برای تابع فراخوانی‌شده (بعنوان پارامتر ورودی) نیز وجود دارد و آن تابع نیز درصورت لزوم می‌تواند یک مقدار را در پایان اجرا و زمان بازگشت به بخش فراخواننده، بعنوان نتیجه اجرا بازگرداند. این روش به‌خوبی تا چند سال قبل پاسخ‌گوی نیازهای برنامه‌نویسان بود؛ اما اخیراً با بزرگ‌شدن و پیچیده‌شدن برنامه‌ها، کنترل برنامه و درک عملکرد آن به‌صورت یکپارچه مشکل شد. کارشناسان با بررسی روند برنامه‌نویسی رایج، متوجه‌شدند که علت این مسئله در نوع طراحی روش رویه‌گرا نهفته است. در این روش، برنامه متشکل از یک‌سری داده‌ها و مجموعه‌ای از کدها بود که برروی داده‌ها کار می‌کردند. بخش‌های مختلف برنامه هم یا به داده‌های مختلف دسترسی نداشتند و یا سطح دسترسی آنها بطور کامل بود. بنابراین، تنها امکان برقراری امنیت داده‌ها پنهان‌سازی آنها از دید سایر بخش‌های برنامه بود. بعلاوه درک این سیستم کدنویسی در برنامه‌های بزرگ مشکل‌بود. با بررسی دقیق‌تر، کارشناسان دریافتند که علت این ابهام آن‌است که ما در زندگی روزمره با محیط پیرامون خود به‌صورت داده‌ها و عملیات مجزا از هم سروکار نداریم. در عوض ما در دنیای واقعی و در محیط پیرامون خود، با اشیاء مواجه هستیم. درنتیجه ایده مدل‌سازی دنیای واقعی در برنامه‌نویسی در ذهن طراحان زبان‌های برنامه‌نویسی شکل‌گرفت. در همین راستا اولین زبان‌های برنامه‌نویسی شئ‌گرا (**Object Oriented**) ازقبیل **Java** و **SmallTalk** و... خلق‌شدند. در این زبان‌ها می‌توانیم با کمک اشیاء، مفاهیم دنیای واقعی (مثل فرآیندها و ایده‌ها) را مدل‌سازی کنیم. در این مدل، برنامه از اشیائی تشکیل می‌شود که هرکدام نقش یکی از اشیاء را در دنیای واقعی بازی می‌کند. همچنین هر شئ می‌تواند کارهای خاصی را انجام‌دهد و سرویس‌های ویژه‌ای را به سایر اشیاء ارائه‌نماید. از آنجا که در هر لحظه از زندگی با این روش سروکار داریم، درک برنامه‌هایی که با این روش نوشته می‌شوند نیز ساده‌تر خواهد بود. در دنیای واقعی هر شئ دارای سه دسته کلی از ویژگی‌ها است (برای درک بهتر، شئ «مداد» بعنوان مثال ذکر شده است) :

۱- خصوصیات: وضعیت ظاهری و فعلی شئ را مشخص می‌کند (مثل ارتفاع مداد، قطر، رنگ نوک مداد، رنگ بدنه و...)

۲- رفتارها: واکنش شئ را درمقابل محرک‌های بیرونی مشخص می‌کند (مثلاً مداد براثر فشار زیاد می‌شکند، با کشیدن روی کاغذ، از خود اثری برجای می‌گذارد و...)

۳- روابط: رابطه شئ را با سایر اشیاء مشخص می‌کند که اغلب از نوع مالکیت است. مثلاً اگر «مداد» درون «کمد» باشد، رابطه مالکیت بین کمد و مداد برقرار است و کمد مالک یا والد مداد محسوب می‌شود. حال اگر کمد را بسوزانیم، مداد نیز از بین می‌رود، درحالی‌که اگر کمد دیگری داشته‌باشیم که مداد درون آن نباشد و کمد فاقد مداد را بسوزانیم، از آنجا که رابطه مالکیت بین کمد سوخته‌شده و مداد برقرار نیست و مداد درون کمد سالم قرار دارد، آسیبی به مداد نخواهد رسید.

حال ببینیم در محیط برنامه‌های کامپیوتری، این مفاهیم چگونه شبیه‌سازی شده‌اند. فرض‌کنید نرم‌افزارهای **Word** و **Photoshop** را اجرا کرده‌اید و می‌خواهیم ویژگی‌های فوق‌الذکر را درباره عناصر داخل این نرم‌افزارها بررسی کنیم:

۱- خصوصیات: هرکدام از عناصر برنامه‌ها مثل دکمه‌ها و... دارای خصوصیاتی ازقبیل پهنا، ارتفاع، متن، رنگ متن، قلم و... هستند.

۲- رفتارها: درصورت کلیک برروی هر دکمه، اتفاق خاصی می‌افتد؛ برخی از عناصر به دوبار کلیک (**Double Click**) حساس هستند و به آن واکنش نشان می‌دهند؛ برخی دیگر به حرکت ماوس یا فشردن کلیدهای صفحه‌کلید پاسخ می‌دهند و...

۳- روابط: اگر پنجره **Word** را ببندیم، نوار ابزار آن در صفحه باقی نمی‌ماند چون رابطه مالکیت بین این شئ و شئ والد یعنی برنامه **Word** برقرار است اما نوار ابزار برنامه **Photoshop** دست‌نخورده باقی خواهند ماند؛ زیرا رابطه مالکیت بین این نوار ابزار و برنامه بسته‌شده یعنی **Word** وجود ندارد.

PHP یکی از محبوب‌ترین زبان‌های اسکریپت‌نویسی در چندسال اخیر است. تقریباً ۶۰ درصد سرورهای وب از برروی نرم‌افزار وب‌سرور **Apache** همراه‌با **PHP** اجرا می‌شوند. **PHP** آنقدر محبوب است که هر ماه میلیون‌ها وب‌سایت و برنامه کاربردی تحت وب توسط آن ساخته می‌شوند. **PHP** سفر خود را بعنوان یک جایگزین ساده برای **Perl** شروع کرد و چند سال بعد، فوق‌العاده محبوب و قدرتمند شد. زبان **PHP** بسیار نزدیک به زبان **C** و شبیه آن است.

همان‌طور که تاکنون قطعاً متوجه شده‌اید، یکی از دلایل محبوبیت فوق‌العاده **PHP**، کوتاه‌بودن منحنی یادگیری آن است. فراگیری **PHP** حقیقتاً کار سختی نیست؛ به‌خصوص اگر با دستور زبان **Java** یا **C** آشنایی داشته باشید. از آنجاکه نوشتن اسکریپت‌های **PHP** ساده است، هر کسی می‌توان کد **PHP** را بدون پیروی از مفاهیم و لایه‌های برنامه‌نویسی استاندارد و منطق‌های برنامه‌نویسی تجاری بنویسد (که این مسئله یکی از دلایل وجود پروژه‌های مدیریت‌نشده زیادی است که پیرامون ما وجود دارند). به‌دلیل عدم وجود قوانین سخت‌گیرانه کدنویسی در **PHP**، یک پروژه در طی سال‌هایی که بزرگ می‌شود، می‌تواند به یک دیو غیرقابل کنترل تبدیل‌گردد!

OOP یا **Object Oriented Programming** (برنامه‌نویسی شئ‌گرا) تمرین خوبی برای ساخت پروژه‌هایی است که راحت‌تر مدیریت می‌شوند. برنامه‌نویسی رویه‌گرا (**Procedural Programming**) به‌معنای نوشتن کد بدون اشیاء است. برنامه‌نویسی رویه‌گرا شامل کدهایی شامل (یا فاقد) روال (**Routine**) است. این بدان‌معناست که برنامه شما شامل مجموعه‌ای از خطوط است که به‌ترتیب اجرا می‌شوند و حداکثر انعطاف‌پذیری در این روش، نوشتن برخی توابع است که آنها نیز درصورت فراخوانی، به‌ترتیب اجرا می‌شوند. **OOP** هر زبان برنامه‌نویسی را با نور کدنویسی بهتر برای بهترین کارآیی و نوشتن پروژه‌های بزرگ بدون نگرانی زیاد درباره مدیریت آنها، روشن می‌کند! **OOP** به شما ابزارهایی برای ساخت اشیاء با قابلیت استفاده مجدد می‌دهد تا شما یا سایر برنامه‌نویسان پروژه بتوانید از آنها در پروژه‌های مختلف بدون نیاز به ساخت و کدنویسی مجدد، بارها و بارها استفاده کنید. **OOP** سختی‌های نوشتن و مدیریت برنامه‌های بزرگ را از بین می‌برد. در این آموزش قصد داریم درباره اینکه چگونه می‌توانید به حداکثر مزایای **OOP** در **PHP** دست‌یابید، صحبت‌کنیم. البته مطابق روش معمول این آموزش، بصورت مرحله‌به‌مرحله پیش خواهیم‌رفت و از مثال‌هایی استفاده می‌کنیم که در پروژه‌های واقعی به‌کار خواهند رفت.

تاریخچه مختصری از برنامه‌نویسی شئ‌گرا در PHP

زمانی که **PHP** ساخته شد، ویژگی‌های برنامه‌نویسی شئ‌گرا را درون خود نداشت. بعد از نسخه **PHP/FI** و زمانی که **Zeev**، **Rasmus** و **Andy** هسته **PHP** را بازنویسی کردند و **PHP3** تولید شد، برخی از ویژگی‌های بسیار ابتدایی شئ‌گرایی درون آن معرفی شد. وقتی **PHP4** منتشر شد، ویژگی‌های شئ‌گرایی به بلوغ نسبی رسیدند و کارآیی آنها بهبود یافت. اما تیم **PHP** مجدداً هسته را بازنویسی کرد تا مدل‌های شئ کاملاً جدیدی را ارائه کند و **PHP5** را منتشر کرد. اکنون دو نسخه از **PHP** تولید می‌شود. در مقایسه نسخه‌های **PHP** با سایر زبان‌ها نباید دچار ابهام و سردرگمی شوید. **PHP5** آخرین نسخه **PHP** نیست. همان‌طور که قبلاً اشاره شد، **PHP4** و **PHP5** هر دو به‌صورت فعال منتشر می‌شوند (هرچند بعد از دسامبر ۲۰۰۷ نسخه‌ای برای **PHP4** منتشر نشده است). در بین این دو سری تولید، **PHP5** تقریباً تمامی ویژگی‌های شئ‌گرایی را پیاده‌سازی می‌کند، درحالی‌که **PHP4** فاقد این ویژگی‌ها است. در حال حاضر آخرین نسخه این دو نگارش، **PHP5.4** و **PHP4.4** است.

روش برنامه‌نویسی رویه‌گرا درمقابل روش برنامه‌نویسی شئ‌گرا در PHP

PHP به شما امکان کدنویسی در دو طعم مختلف را می‌دهد. یک روش، رویه‌گرا و دیگری شئ‌گرا نام‌دارد. شما حتی می‌توانید در **PHP5** کدهای خود را بصورت رویه‌گرا بنویسید و این کدها بدون هرگونه مشکلی اجرا خواهند شد. اگر هنوز درک روشنی از برنامه‌نویسی رویه‌گرا و شئ‌گرا ندارید، بهتر است برای هرکدام از روش‌های فوق یک نمونه ارائه‌دهیم. دو نمونه کد زیر را مدنظر قراردهید:

```php
<?php
    session_start();
    if(isset($_POST['username'], $_POST['password'])) {
        mysql_connect('localhost', 'root', '') or die('Connection error');
        mysql_select_db('dbname') or die('Database error');
        mysql_query('SET NAMES \'utf8\'');
        mysql_set_charset('utf8');
        $username = mysql_real_escape_string(strtolower($_POST['username']));
        $password = md5($_POST['password']);
        $info = mysql_query("SELECT * FROM `users` WHERE (LOWER(`username`)='{$username}' AND `password`='{$password}')");
        if($info && mysql_num_rows($info)) {
            $info = mysql_fetch_assoc($info);
            $_SESSION['uid'] = $info['id'];
        }
    }
    header('Location: index.php');
    exit();
?>
```

کد فوق، مثالی از برنامه‌نویسی رویه‌گرا است که در آن، بسیاری از پردازش‌ها بصورت درون‌خطی انجام‌شده و حداکثر انعطاف آن در صدازدن توابع یا رویه‌های دیگر خلاصه شده‌است. حال اگر بخواهیم همین کد را بصورت شئ‌گرا بنویسیم، مشابه کد زیر خواهد شد:

```php
<?php
    session_start();
    $filter = new Filter();
    // Check the valid username and password are sent via post method
    if($filter->ValidPost('username', 'password')) {
        $db = new DAL('mysql'); // Data access layer
        $db->Connect('localhost', 'root', '', 'dbname');
        $username = $db->Escape(strtolower($_POST['username']));
        $password = $db->MakeHash($_POST['password']);
        $info = $db->FetchOne("SELECT * FROM `users` WHERE (LOWER(`username`)='{$username}' AND `password`='{$password}')");
        if($db->ValidResult($info)) {
            $_SESSION['uid'] = $info['id'];
        }
    }
    header('Location: index.php');
    exit();
?>
```

اگر با دقت به دو کد فوق نگاه کنید، خواهید دید که کد دوم به‌مراتب خواناتر و مرتب‌تر است. البته با معرفی توابع بیشتر، می‌توان کد اول را نیز بهینه‌سازی نمود اما حقیقتاً آمادگی به‌خاطر سپردن چند تابع را دارید تا در زمان نیاز، آنها را فراخوانی کنید؟ کد دوم خوانایی بیشتری دارد زیرا شما می‌دانید که هر شئ کدام عملیات را برعهده دارد. اگر برنامه‌های بزرگ را به‌روش رویه‌گرا بنویسید، مدیریت آنها بعد از ارائه چند نسخه مختلف، تقریباً غیرممکن خواهد شد. البته شما می‌توانید قوانین سخت‌گیرانه و محکمی برای کدنویسی پیاده‌کنید تا کد استاندارد و بهینه‌ای داشته‌باشید اما میلیون‌ها برنامه‌نویس با این امر موافقند که درنهایت، روش رویه‌گرا به‌اندازه روش شئ‌گرا نمی‌تواند حداکثر قابلیت مدیریت و بهینه‌سازی را به شما عرضه‌کند. تقریباً تمامی پروژه‌های بزرگ با روش شئ‌گرا نوشته شده‌اند.

مزایای برنامه‌نویسی شئ‌گرا

OOP برای ساده‌کردن زندگی برنامه‌نویسان به‌وجود آمده است. با استفاده از **OOP** شما می‌توانید مسائل بزرگ را تبدیل به چند مسئله کوچک کنید که نسبت‌به مسئله اصلی، راحت‌تر حل می‌شوند. هدف اصلی **OOP** این است: هر کاری می‌خواهید انجام‌دهید باید ازطریق اشیاء انجام‌شود. اشیاء در اصل اجزای کوچکی از کد هستند که می‌توانند داده‌ها و رفتارها را بطور همزمان در بر داشته‌باشند. در یک برنامه، تمامی این اشیاء با هم در ارتباط هستند و داده‌ها را به‌اشتراک گذاشته و مسائل را حل می‌کنند.

OOP از جنبه‌های مختلفی می‌تواند روش برتر شناخته‌شود؛ علی‌الخصوص زمانی که زمان و هزینه نگهداری پروژه اهمیت داشته‌باشد. اهداف اصلی **OOP** را می‌توان به‌طور خلاصه چنین برشمرد:

۱- قابلیت استفاده مجدد: یک شئ، موجودیتی است که می‌تواند خصوصیات و رفتارها را در بر داشته و با اشیاء دیگر مرتبط باشد. یک شئ می‌تواند مستقل یا وابسته به سایر اشیاء باشد؛ اما درهرحال یک شئ اغلب برای رفع مجموعه مشخصی از مشکلات تولید می‌شود. درنتیجه هرگاه در سایر پروژه‌ها نیاز به رفع مشکلات مشابهی باشد، می‌توان از همان شئ در آن پروژه‌ها نیز استفاده‌نمود.

درحقیقت اشیاء از «اختراع مجدد چرخ» جلوگیری می‌کنند. در برنامه‌نویسی رویه‌گرا، استفاده مجدد از کدی که قبلاً نوشته‌شده، ممکن اما سخت و پیچیده است.

۲- اصلاح مجدد: اگر نیاز به اصلاح پروژه‌های خود داشته باشید، **OOP** به شما حداکثر سود را می‌رساند زیرا تمامی اشیاء موجود در برنامه، موجودیت‌های کوچکی هستند که شامل خصوصیات و رفتارهای خاص خود می‌باشند. بنابراین، تغییر و اصلاح آنها به‌مراتب ساده‌تر است.

۳- قابلیت گسترش: درصورت نیاز به افزودن امکانات خاصی به پروژه، می‌توانید بهترین نتایج را با استفاده از **OOP** کسب‌کنید. یکی‌از ویژگی‌های بنیادی **OOP**، قابلیت گسترش آن است. می‌توانید اشیاء خود را بازنویسی‌کرده و ویژگی‌های خاصی را به آنها اضافه‌کنید. همزمان با این کار می‌توانید سازگاری با نسخه‌های قبلی را نیز حفظ‌نمایید. برای این کار، کد قبلی شما بصورت پایه تعریف و امکانات جدید به آن اضافه شده و آنرا گسترش می‌دهند. بدین‌ترتیب، اشیاء جدید تمامی خصوصیات و شئ والد را از آن به‌ارث می‌برند و سپس، ویژگی‌های جدید را به آن اضافه می‌نمایند. این عمل، اصطلاحاً «وراثت» نام دارد و یکی‌از ویژگی‌های بسیار مهم **OOP** به‌شمار می‌رود.

٤- قابلیت نگهداری: کد شئ‌گرا راحت‌تر نگهداری می‌شود زیرا از مفاهیم و اصول نسبتاً سخت‌گیرانه‌تری تبعیت می‌کند و در قالب یک ساختار واضح و روان نوشته شده‌است. برای مثال وقتی که یک برنامه‌نویس آنرا توسعه می‌دهد، بازنویسی و یا اشکال‌زدایی می‌کند، می‌توان به‌راحتی ساختار کدنویسی داخلی آنرا کشف‌کرد و کد را در هر زمان به‌روزرسانی نمود. بعلاوه، هرگاه یک محیط توسعه تیمی بر پروژه حاکم باشد، **OOP** بهترین راهکار ممکن خواهد بود؛ زیرا می‌توانید کدنویسی را با تقسیم وظایف، بین افراد گروه توزیع کنید. این بخش‌های کوچک هرکدام بصورت اشیاء جداگانه تعریف می‌شوند و لذا برنامه‌نویسان می‌توانند آنها را بطور تقریباً مستقل توسعه‌دهند. درنهایت، ادغام کد نیز به‌دلیل تبعیت کدهای اشیاء جداگانه از الگوی ساختاری مشخص، کار راحتی خواهد بود.

٤- کارآیی: مفهوم برنامه‌نویسی شئ‌گرا درحقیقت برای کارآیی بهتر و راحتی فرایند توسعه نرم‌افزار به‌وجود آمده‌است. الگوهای طراحی مختلفی برای تولید کد کاراتر و بهتر به‌وجود آمده‌اند. همچنین در **OOP** شما می‌توانید به پروژه خود با دید بهترین نسبت‌به برنامه‌نویسی رویه‌گرا نگاه‌کنید؛ زیرا ابتدا مسئله را به مجموعه‌ای از مسائل کوچک‌تر تبدیل می‌کنید و سپس، راه‌حل این مسائل کوچک را می‌یابد و طبیعتاً مسئله بزرگ و اصلی بطور خودکار حل خواهد شد.

مفهوم کلاس

کلاس (**Class**) چیست؟ قبلاً گفتیم که برنامه‌نویسی شئ‌گرا از محیط واقعی پیرامون ما الگوبرداری شده‌است. جهان اطراف ما سرشار از اشیاء مختلف است که ما برای راحتی در کار با اشیاء و به‌خاطر سپردن آنها، طبقه‌بندی‌های مختلفی برای آنها ایجاد کرده‌ایم. برای مثال، «اتومبیل» یک طبقه‌بندی است. وقتی ما کلمه اتومبیل را می‌شنویم، بطور کاملاً ناخودآگاه برخی خصوصیات و تصویر مبهمی از یک اتومبیل (بسته به سلیقه و حافظه ما) در ذهنمان نقش می‌بندد. مزیت اصلی طبقه‌بندی نیز همین مسئله است. برای مثال، اگر به ما بگویند **Bugatti Veyron** یک اتومبیل است، حتی اگر در طول زندگی خود، این اتومبیل را ندیده باشیم، تصویر مبهمی از آن را در ذهنمان تجسم خواهیم کرد (دست‌کم می‌دانیم ٤ چرخ دارد، از بدنه، شاسی، جعبه‌دنده، سیستم سوخت‌رسانی و... تشکیل شده‌است). حال طبقه‌بندی دیگری مثل «سیب» را درنظر بگیرید. قطعاً تأیید می‌کنید که بلافاصله تصویر یک سیب (برحسب سلیقه و ذائقه شما سبز، زرد یا قرمز) در ذهنتان مجسم شد. در برنامه‌نویسی، از واژه کلاس به‌جای طبقه‌بندی استفاده می‌کنیم.

تفاوت کلاس و شئ

حال که فهمیدیم کلاس چیست، باید بفهمیم تفاوت بین کلاس و شئ در چیست؟ برای درک بهتر، مجدداً اجازه‌دهید از دنیای واقعی مثال مطرح کنیم. کلاس «سیب» را که به‌خاطر دارید؟ فرض‌کنید کنفرانسی به‌مدت ۳ روز متوالی درباره خواص سیب، مزایای طبی و دارویی، عطر، طعم، مواد تشکیل‌دهنده آن و... برگزار کنیم. آیا چیزی به‌جز برخی اطلاعات دستگیر شما خواهد شد؟ تا زمانی که یک «سیب» بصورت فیزیکی در اختیار نداشته‌باشید و لذت گاززدن و خوردن آنرا نچشید، کنفرانس مزبور هیچ فایده‌ای برای شما

نخواهد داشت! تفاوت میان طبقه‌بندی «سیب» و «سیب» فیزیکی دقیقاً تفاوت میان «کلاس» و «شئ» است. یک شئ، نمونه‌ای از یک کلاس است. در برنامه‌نویسی، با تعریف کلاس، یک طبقه‌بندی ایجاد می‌کنیم. اما این کلاس به‌تنهایی هیچ قابلیت اجرایی ندارد. وقتی از کلاس شئ ایجاد می‌کنیم، آن‌وقت می‌توان از آن شئ در برنامه استفاده‌نمود و تمامی ویژگی‌هایی که در کلاس مربوطه تعریف شده‌است (اعم‌از خصوصیات، رفتارها و...) برروی شئ مربوطه قابل دسترسی خواهند بود. همان‌طور که در دنیای واقعی، هیچ شیئی بدون طبقه‌بندی وجود ندارد، در **OOP** نیز هیچ شیئی بدون کلاس وجود ندارد و به‌عبارت دیگر، اشیاء ازروی کلاس‌ها ساخته می‌شوند و تا زمانی که کلاسی وجود نداشته‌باشد، شیئی نیز وجود نخواهد داشت.

تعریف دقیق‌تر کلاس

اجازه‌دهید نگاهی عمیق‌تر به یک کلاس بیاندازیم. حقیقتاً یک کلاس چیست؟ بسیارخوب، کلاس چیزی به‌جز قطعه‌ای از کد با تعدادی خصوصیات (فیلد – **Field**) و رفتارها (متد – **Method**) نیست. بنابراین، آیا یک کلاس تاحدودی شبیه یک آرایه است (زیرا آرایه‌ها می‌توانند داده‌هایی را در خود نگهداری کنند که توسط اندیس (یا کلید) از هم تفکیک می‌شوند)؟ پاسخ این سؤال، **خیر** است. کلاس‌ها بسیار فراتر از آرایه‌ها هستند زیرا می‌توانند علاوه‌بر داده‌ها (فیلدها) شامل رفتارها (متدها) هم باشند. می‌توانند هرکدام از این فیلدها و متدها را آشکار یا مخفی کنند (که در آرایه ممکن نیست). کلاس تاحدودی قابل‌مقایسه با ساختار داده‌ها (**Data Structure**) است و می‌تواند حاوی اشیائی از کلاس‌های دیگر (یا حتی اشیائی از نوع خودش) باشد. در آینده درمورد ساخت کلاس‌های ساده و پیچیده بحث مفصلی خواهیم داشت.

یک کلاس نمونه

بهترین راه برای تفهیم توضیحات، ارائه مثال است. بنابراین اجازه‌دهید یک کلاس نمونه را بررسی کنیم:

```php
class Point {
    private $x;
    private $y;

    public function Point($x = 0, $y = 0) {
        $this->Set($x, $y);
    }

    public function Set($x = 0, $y = 0) {
        $this->SetX($x);
        $this->SetY($y);
    }

    public function SetX($x) {
        $this->x = ($x >= 0 && $x <= 639) ? $x : 0;
    }

    public function SetY($y) {
        $this->y = ($y >= 0 && $y <= 479) ? $y : 0;
    }

    public function Display() {
        echo '<p>X = ' . $this->x . ' , Y = ' . $this->y . '</p>' . PHP_EOL;
    }

    public function Distance($other) {
        echo '<p>Distance = ' . $this->calc($other) . '</p>' . PHP_EOL;
    }

    private function calculate($other) {
        $dX = $this->x - $other->x;
        $dY = $this->y - $other->y;
        return round(sqrt($dX * $dX + $dY * $dY), 2);
    }
}
```

فعلاً قصد نداریم وارد جزئیات کد شویم. بطور خلاصه در کد فوق، کلاسی به‌نام **Point** تعریف شده‌است که حاوی دو فیلد خصوصی (**Private**) به‌اسامی **$x** و **$y** است و پس‌از آنها، ٦ متد عمومی (**Public**) با اسامی **Point** و **Set** و **SetX** و **SetY** و **Display** و **Distance** معرفی شده‌اند. این متدها به‌ترتیب سازنده کلاس، ٣ متد دستیاب و دو متد عملیاتی هستند. درمورد سازنده و

دستیاب نیز در آینده صحبت خواهیم کرد. متدهای عملیاتی، همانند توابع معمولی هستند که در برنامه‌نویسی رویه‌گرا، در خارج از کلاس تعریف می‌شوند و عمل مشخصی را انجام می‌دهند که می‌تواند محاسبه یک فرمول، تولید یک نمودار یا چاپ نتایج دلخواه باشد. در نهایت نیز یک متد خصوصی به‌نام **calculate** معرفی شده‌است که در درون متد عمومی **Display** از کلاس به‌کار می‌رود. باید دقت کنید که این کلاس، فقط یک طبقه‌بندی (ساختار) به‌نام **Point** را معرفی می‌کند و به‌تنهایی هیچ‌گونه کاربرد عملی ندارد و برای استفاده از آن باید یک شئ از کلاس مذکور ایجاد کنیم.

یک شئ نمونه

حال اجازه‌دهید به کد نمونه‌ای که از کلاس **Point** استفاده می‌کند توجه‌کنیم. در این مثال، فرض بر آن است که کد کلاس **Point** را در فایل **class.point.php** ذخیره کرده‌ایم.

```
require_once 'class.point.php';
$p1 = new Point();
$p1->Display();
$p2 = new Point(320, 240);
$p2->Display();
$p1->Set(100, 200);
$p1->Display();
$p2->SetX(200);
$p2->SetY(100);
$p2->Display();
$p1->Distance($p2);
```

خروجی کد فوق به‌صورت زیر است:

```
X = 0 , Y = 0
X = 320 , Y = 240
X = 100 , Y = 200
X = 200 , Y = 100
Distance = 141.42
```

مجدداً تأکید می‌کنیم که فعلاً به کدها و ساختار آنها توجه نکنید. به‌طور خلاصه، در کد فوق ابتدا یک شئ به‌نام **$p1** از کلاس **Point** ایجاد می‌شود و بعد تابع **Display** آن شئ فراخوانی می‌گردد. سپس شئ **$p2** با مختصات دلخواه از روی کلاس **Point** ایجاد شده و تابع **Display** آن صدا زده می‌شود. همان‌طور که مشاهده می‌کنید، هر شئ مؤلفه‌های **X** و **Y** مخصوص به خود را داراست. سپس با کمک تابع **Set** مختصات شئ **$p1** تغییر کرده و مجدداً تابع **Display** آن فراخوانی می‌شود. در ادامه، با فراخوانی توابع **SetX** و **SetY** از شئ **$p2** مختصات این شئ نیز تغییر کرده و با فراخوانی تابع **Display** چاپ می‌شود. در نهایت نیز با فراخوانی تابع **Distance** از شئ **$p1** و ارسال شئ **$p2** بعنوان پارامتر برای تابع فوق، فاصله دو شئ (دو نقطه با مختصات مؤلفه‌های **X** و **Y** اشیاء **$p1** و **$p2**) محاسبه‌شده و به نمایش در می‌آید.

همان‌گونه که مشاهده کردید، در کد فوق که به روش شئ‌گرا نوشته شده است، متدهای کلاس یک‌بار نوشته شده‌اند و تمامی اشیائی که از کلاس فوق ایجاد می‌شوند، تمامی ساختار معرفی‌شده توسط کلاس را دارا می‌باشند.

برخی تعاریف پرکاربرد در برنامه‌نویسی شئ‌گرا

کلاس: یک کلاس (**Class**)، الگوی یک شئ است. کلاس شامل کدی است که مشخص می‌کند اشیاء ایجاد شده از آن چگونه عمل‌کرده و با اشیاء دیگر تعامل داشته‌باشند.

شئ: یک شئ (**Object**)، نمونه‌ای از یک کلاس است که قابلیت اجرا و کاربرد دارد و دارای خصوصیات و رفتارهای خاص خود است که ساختار آنها توسط کلاس مربوطه تعریف شده‌است.

خاصیت: یک خاصیت (**Property**) یا فیلد (**Field**)، درحقیقت متغیری از کلاس است که مستقیماً درون خود کلاس (و نه درون متدهای داخل کلاس) تعریف می‌شود. این عناصر توسط تمامی توابع (متدها) در تمامی بخش‌های کلاس قابل دسترسی هستند. برای مثال، یک متد می‌تواند خاصیتی را مقداردهی کرده و متد دیگر، آنرا خوانده و نمایش دهد.

متد: متدها (**Method**) توابعی هستند که درون یک کلاس تعریف می‌شوند و رفتارهای اشیاء ایجادشده از کلاس را تعریف می‌کنند.

کپسوله‌سازی: این اصطلاح (**Encapsulation**) به‌معنای مکانیزمی است که توسط آن، کد درون کلاس (متدها) و داده‌های موجود در آن (فیلدها) به هم متصل می‌شوند و هر دو مورد (فیلد و متد) از تداخل خارجی و استفاده نامناسب مصون می‌مانند. درحقیقت به عمل محافظت از داده‌ها و متدها ازطریق یک سیستم واحد (کلاس)، کپسوله‌سازی می‌گویند. مزیت اصلی کپسوله‌سازی تضمین صحت اطلاعات و انجام عملیات است.

وراثت: فرآیند مشتق‌شدن یک کلاس از کلاس دیگر و گسترش امکانات تعریف‌شده توسط آن، وراثت (**Inheritance**) نام دارد. وقتی یک کلاس را از دیگری مشتق می‌کنید، کلاس مشتق‌شده (فرعی) تمامی رفتارها و خصوصیات کلاس والد (اصلی) را به‌ارث می‌برد. سپس کلاس فرعی می‌تواند متدها و فیلدهای دلخواه خود را اضافه‌کرده یا تغییر دهد (که به این عمل بازنویسی یا **Override** می‌گویند).

چندریختی: اشیاء می‌توانند از نوع هر کلاسی باشند. اگر یک شئ، از کلاسی ایجاد شود که مشتق‌شده از کلاس دیگری نیست، به آن شئ خالص می‌گوییم. همان‌طور که گفتیم، اگر یک کلاس از کلاس دیگری مشتق شده‌باشد، می‌توانند متدهای آنرا بازنویسی کند. بدین‌ترتیب، اگر یک شئ از کلاس مشتق‌شده ایجاد شود، این امکان وجود دارد که با فراخوانی یک متد خاص، رفتاری متفاوت با زمانی که همان متد را از شیئی که از کلاس والد ایجاد شده‌است صدا می‌زنیم، اجرا گردد. به این عمل، چندریختی (**Polymorphism**) می‌گوییم که در آن، یک شئ بسته به آنکه از کلاس والد یا مشتق‌شده ایجاد شود، با فراخوانی یک متد خاص می‌تواند به‌اشکال مختلفی رفتار کند. توضیح کامل این مبحث را در آینده ارائه خواهیم کرد.

امتزاج: این اصطلاح (**Coupling**) بیانگر وابستگی کلاس‌ها به یکدیگر است. برای مثال، ممکن‌است یک کلاس حاوی یک یا چند فیلد باشد که این فیلدها درحقیقت اشیائی از کلاس‌های دیگر باشند. بدین‌ترتیب، اشیاء یک کلاس بدون وجود کلاس دیگر قادر به کارکردن نیستند. هرچقدر امتزاج یک کلاس کمتر باشد (کمتر از اشیاء کلاس‌های دیگر استفاده کرده باشد)، قابلیت استفاده مجدد کلاس مزبور در سایر پروژه‌ها افزایش خواهد یافت و بالعکس، کلاس‌هایی که امتزاج بالایی دارند، باید همراه با سایر کلاس‌های مرتبط، در سایر پروژه‌ها به‌کار گرفته‌شوند. بنابراین مشاهده می‌کنید که امتزاج یکی از مفاهیم مهم در طراحی اشیاء بهینه است.

نمونه: هر زمان که یک شئ از یک کلاس ایجاد می‌کنیم، به آن نمونه (**Instance**) می‌گوییم. به‌بیان ساده‌تر، اشیاء درحقیقت نمونه‌های یک کلاس هستند.

کلاس برتر (والد): در مفاهیم وراثت، کلاسی که سایر کلاس‌ها از آن مشتق می‌شوند و امکانات آنرا توسعه می‌دهند، کلاس برتر (**SuperClass**) یا والد (**Parent**) نام دارد.

کلاس فرعی (فرزند): به کلاسی که از یک کلاس دیگر مشتق شده‌باشد و خصوصیات ورفتارهای آنرا به‌ارث می‌برد، کلاس فرعی (**SubClass**) یا فرزند (**Child**) می‌گویند.

الگوی طراحی: این اصطلاح (**Design Pattern** یا به‌اختصار **DP**) اولین‌بار توسط گروه ٤ (**Gang of Four**) به‌کار رفت و به‌معنای روش‌هایی است که برنامه‌نویسی شئ‌گرا به‌شکل هوشمندانه برای حل مجموعه مشکلات مشابه، به‌کار می‌برد. با استفاده از **DP** می‌توان کارآیی کلی برنامه را با حداقل کد، افزایش چشمگیری داد. گاهی‌اوقات نوشتن کد بهینه بدون داشتن یک الگوی طراحی مناسب امکان‌پذیر نیست. درمقابل استفاده از **DP** در مواقع غیرضروری یا به‌شکل اشتباه نیز می‌تواند کارآیی را کاهش‌دهد. در آینده درخصوص الگوهای طراحی نیز بحث مفصلی خواهیم‌داشت.

خط‌مشی‌های کلی کدنویسی

برای حفظ یکپارچگی و رعایت استانداردها، ما از اصول استاندارد کدنویسی شئ‌گرا بهره می‌جوییم. این اصول به شما در نگهداری برنامه‌های بزرگ کمک شایانی خواهد نمود. همچنین قابلیت نگهداری و توسعه کد شما را افزایش خواهد داد. بعلاوه شما را در نوشتن کد کارآمد و بهینه (ازطریق پرهیز از تکرار کد و تولید کلاس‌های متداخل و هم‌پوشان) یاری خواهد کرد. درنهایت نیز کد شما خواناتر خواهد شد.

۱- در یک فایل **PHP** هرگز بیش از یک کلاس نمی‌نویسیم. خارج‌از محدوده آن کلاس نیز هیچ‌گونه کد رویه‌گرایی نوشته نخواهد شد.

۲- ما هر کلاس را با یک نام مناسب ذخیره می‌کنیم. برای مثال، کلاس **Point** را در فایلی به‌نام **class.point.php** ذخیره می‌کنیم. مزیت این روش نام‌گذاری در آن است که بدون بازکردن فایل فوق، می‌دانیم حاوی کد کلاس **Point** است. بعلاوه از آنجا که اسامی همه فایل‌های کلاس ما با **class.** شروع می‌شود، درنتیجه در سیستم فایل نیز تمامی فایل‌های کلاس‌های پروژه در کنار هم فهرست خواهند شد و عمل یافتن کلاس‌ها سهولت و سرعت بیشتری می‌یابد.

۳- هرگز در اسامی فایل‌ها از حروف بزرگ استفاده نمی‌کنیم. این مسئله موجب ایجاد ساختار مناسب در برنامه می‌شود. اسامی فایل‌ها را تماماً با حروف کوچک بنویسید. این مسئله در سرورهایی مثل لینوکس که نسبت‌به بزرگی و کوچکی حروف حساسند، اهمیت دوچندان پیدا می‌کند؛ زیرا وجود حتی یک‌حرف تفاوت در بزرگی و کوچکی حروف موجب می‌شود که با خطای «عدم یافتن فایل» مواجه‌شویم.

٤- مشابه کلاس‌ها، رابط‌ها را با اسامی شبیه **interface.name.php** (**name** معادل حروف کوچک نام رابط است) و کلاس‌های چکیده را با اسامی مشابه **abstract.name.php** و کلاس‌های نهایی را با نامی مشابه **final.name.php** ذخیره می‌کنیم. درباره مفاهیم رابط، کلاس چکیده و نهایی و... در آینده توضیح خواهیم داد.

۵- ما همیشه از ساختار نام‌گذاری **CamelCase** برای تعیین نام کلاس‌ها استفاده می‌کنیم. در این ساختار، برای نام‌گذاری فقط از حروف الفبا و اعداد استفاده می‌شود و اسامی با یک حرف الفبا شروع شده و حرف اول هر کلمه بصورت بزرگ و مابقی حروف بصورت کوچک تایپ می‌شود (مثل **SomeClass** یا **My1stClass** و...).

٦- برای فیلدها و متدهای عمومی (**Public**) از ساختار نام‌گذاری **CamelCase** و برای فیلدها و متدهای خصوصی (**Private**) از ساختار **PascalCase** استفاده می‌کنیم. این ساختار، مشابه **CamelCase** است؛ با این تفاوت که کلمه اول تماماً با حروف کوچک نوشته می‌شود (مثل **myVariable** یا **someUsefulMethod** و...).

جمع‌بندی

در این جلسه، یک آشنایی بسیار جزئی و اجمالی با برنامه‌نویسی شئ‌گرا و چگونگی ارتباط آن با **PHP** پیدا کردیم. همچنین آموختیم که مزایای عمده این روش برنامه‌نویسی نسبت‌به برنامه‌نویسی رویه‌گرا و تابع‌گرا چیست و البته وارد اعماق **OOP** نشدیم. در آینده بیشتر با اشیاء و متدها و فیلدهای آنها آشنا خواهیم‌شد و به‌ویژه برروی ساخت کلاس‌ها، توسعه قابلیت‌های آنها و کارکردن با آنها ازطریق اشیاء ایجادشده از کلاس‌ها تمرکز خواهیم‌کرد. بنابراین، از جلسه بعد سفر ما به اعماق **OOP** ازطریق **PHP** آغاز خواهد شد.

شروع برنامه‌نویسی شئ‌گرا در PHP

در این جلسه می‌خواهیم چگونگی تعریف کلاس‌ها، تعریف فیلدها و پیاده‌سازی متدهای آنها را آموزش‌دهیم. کلاس‌ها در **PHP** همیشه با استفاده از کلمه کلیدی **class** ایجاد می‌شوند. در این جلسه جزئیات کلاس‌ها را فرا خواهیم گرفت. همچنین می‌آموزیم که محدوده متدها چیست و تغییردهنده‌ها چه هستند و مزایای کاربرد رابط‌ها چیست. ضمناً این جلسه ویژگی‌های ابتدایی شئ‌گرایی را در **PHP** معرفی می‌کند. بطور کلی، این جلسه یکی از بهترین منابع شما برای شروع برنامه‌نویسی شئ‌گرا در **PHP** خواهد بود.

رفع یک ابهام مهم!

در طی سالهای مختلف، برنامه‌نویسان و اساتید دروس مرتبط با برنامه‌نویسی شئ‌گرا بارها درباره کلاس‌ها و اشیاء صحبت کرده‌اند و متأسفانه آنها را بصورت واژه‌های قابل جایگزینی به‌جای هم معرفی کرده‌اند. حقیقت این است که این دو مفهوم کاملاً متفاوت هستند (و در جلسه قبل به این نکته اشاره کردیم)؛ هرچند تفاوت آنها ممکن است کمی مبهم باشد. اجازه‌دهید قبل از شروع به کار، یک‌بار برای همیشه مفهوم این دو واژه را بطور کامل تفهیم‌کنیم.

یک کلاس، برای مثال شبیه نقشه یک خانه است. این نقشه، شکل خانه را برروی کاغذ تعریف می‌کند و ارتباط بین بخش‌های مختلف خانه را بطور واضح مشخص و طرح‌ریزی می‌نماید؛ هرچند خانه‌ای واقعاً وجود ندارد.

حال یک شئ، یک خانه واقعی است که برمبنای نقشه مذکور، ساخته می‌شود. داده‌های ذخیره‌شده در شئ مشابه چوب، سیم‌ها و سیمانی است که خانه را شکل می‌دهد: بدون چیدمان صحیح برمبنای نقشه، فقط توده‌ای از مواد اولیه خواهیم‌داشت؛ حال آنکه اگر آنها را برمبنای نقشه ترکیب کنیم، یک خانه ساخت‌یافته و قابل استفاده خواهیم‌داشت.

کلاس‌ها ساختار داده‌ها و اعمال را مشخص می‌کنند و با استفاده از این اطلاعات، اشیاء را می‌سازیم. می‌توانیم بیش از یک شئ از کلاس مشترک در زمان یکسان تعریف‌کنیم، درحالی‌که هرکدام مستقل از بقیه هستند. در مثال ساختمان نیز بطور مشابه، می‌توان یک مجتمع مسکونی کامل را صرفاً با یک نقشه واحد تولید کرد: ۱۵۰ ساختمان مختلف وجود دارند که همه مشابه به‌نظر می‌رسند، اما هرکدام دارای مصالح مختلف هستند و خانواده‌ها و دکوراسیون متفاوتی را در خود جای داده‌اند.

تعریف یک کلاس

همانطور که قبلاً گفتیم، می‌توانید یک کلاس را در **PHP** با استفاده از کلمه کلیدی **class** ایجاد کنید. یک کلاس می‌تواند حاوی خصوصیات و رفتارهای متعدد اعم از عمومی یا خصوصی باشد. برای مثال، به کلاس زیر دقت کنید:

```php
<?php
    class Emailer {
        private $sender;
        private $recipients;
        private $subject;
        private $body;

        public function __construct($sender) {
            $this->sender = $sender;
            $this->recipients = array();
        }

        public function AddRecipients($recipient) {
            array_push($this->recipients, $recipient);
        }

        public function SetSubject($subject) {
            $this->subject = $subject;
        }

        public function SetBody($body) {
            $this->body = $body;
        }

        public function SendEmail() {
            foreach ($this->recipients as $recipient) {
                $result = mail($recipient, $this->subject, $this->body, "From: {$this->sender}\r\n");
                if ($result) {
                    echo "Mail successfully sent to {$recipient}<br/>" . PHP_EOL;
                }
            }
        }
    }
?>
```

در این کد، ما کار را با عبارت **class Emailer** شروع کرده‌ایم که معنای آن، انتخاب اسم **Emailer** برای کلاسی است که قصد طراحی آنرا داریم. در زمان نام‌گذاری یک کلاس، باید از همان قواعد نام‌گذاری متغیرها و سایر شناسه‌ها در زبان **PHP** پیروی کنید (مثلاً نمی‌توانید نام را با یک عدد شروع کنید و...).

سپس خصوصیات این کلاس را تعریف کرده‌ایم که در اینجا ٤ خاصیت به اسامی **$sender** و **$recipient** و **$subject** و **$body** وجود دارد. دقت‌کنید که قبل از اسم هرکدام از آنها، کلمه کلیدی **private** را نوشته‌ایم که به‌معنای خصوصی است. یک خاصیت خصوصی فقط در داخل کلاس قابل دسترسی است. خاصیت‌ها هم چیز عجیبی نیستند! آنها فقط متغیرهایی هستند که درون یک کلاس تعریف می‌شوند و در اصول و تعاریف شئ‌گرایی به آنها فیلد (**Field**) هم می‌گویند. بنابراین اگر در کتاب یا مقاله‌ای با واژه فیلد برخورد کردید، دچار سردرگمی نشوید (منظور از فیلد، همان خاصیت است و خاصیت نیز متغیری است که درون یک کلاس تعریف می‌شود).

نوع عنصر بعدی که در یک کلاس می‌توان تعریف کرد، متد نام دارد. متد نیز چیز جدیدی نیست که قبلاً با آن کار نکرده باشید. متد فقط یک تابع است که درون کلاس تعریف می‌شود. بنابراین تا اینجا فهمیدیم که به متغیرهای درون کلاس، خاصیت یا فیلد و به توابع درون کلاس، متد می‌گویند. این کلاس حاوی ٥ متد است:

```
__construct()
AddRecipient()
SetSubject()
SetBody()
SendMail()
```

دقت کنید که تمامی متدهای کلاس فوق با کلمه کلیدی **public** (یعنی عمومی) معرفی شده‌اند. این بدان‌معنی است که هر بخشی از برنامه که یک شئ از این کلاس ایجاد کند، می‌تواند این متدها را ازطریق شئ مربوطه، فراخوانی نماید. به‌عبارت دیگر، اگر یک عنصر (اعم از فیلد یا متد) بصورت خصوصی تعریف شود، فقط در داخل کلاس قابل دسترسی است و ازطریق اشیاء ایجادشده از کلاس نمی‌توانیم به آن دسترسی پیدا کنیم.

در نام‌گذاری عناصر داخل کلاس، چه فیلد باشند و چه متد، بهتر است قراردادهای زیر را رعایت کنید:

- از کارکتر _ (**UnderScore**) در اسامی استفاده نکنید.
- عناصر **private** (خصوصی) را با روش **camelCase** نام‌گذاری کنید (کلمه اول تماماً با حروف کوچک و حرف اول بقیه کلمات بصورت بزرگ). مثال: **myFirstValue** یا **mailRecipients** و...
- عناصر **public** (عمومی) را با روش **PascalCase** نام‌گذاری کنید (حرف اول تمام کلمات بصورت بزرگ و بقیه حروف بصورت کوچک). مثال: **SendMail** یا **TheValueOfFuture** و...

البته همان‌طور که گفتیم، این موارد قرارداد هستند نه قانون ولی ایده بدی نیست که یک کدنویسی استاندارد داشته‌باشیم و چه‌بهتر که این استاندارد، بصورت جهانی پذیرفته باشد تا کدی که می‌نویسیم، مقبولیت بیشتری داشته و ضمناً با درک و تفهیم کدی که از سایر منابع (مثل اینترنت و پروژه‌های گروهی) بدست می‌آوریم و سازگارکردن آن با سایر کدهای خودمان، مشکلی نداشته‌باشیم.

متد **__constuct()** یک استثنا است که در همه کلاس‌ها وجود دارد و شکل نوشتن آن نیز به‌همین‌ترتیب باید باشد (ابتدا دوبار کارکتر _ (**UnderScore**) و سپس کلمه **contruct** با حروف کوچک). هرگاه بخواهیم یک شئ جدید از کلاس ایجاد کنیم، این متد بطور خودکار فراخوانی می‌شود. بنابراین اگر می‌خواهیم کارهایی را در زمان آماده‌سازی و ایجاد شئ انجام‌دهیم، دستورات مربوطه را در این متد قرار می‌دهیم که به آن، سازنده (**Constructor**) می‌گوییم. برای مثال، در کلاس فوق، سازنده یک پارامتر دریافت می‌کند (نام فرستنده پیام) و سپس، خاصیت **$sender** شئ مربوطه را با آن پارامتر مقداردهی کرده و گیرندگان پیام (خاصیت **recipients** شئ ایجادشده از کلاس) را نیز بصورت یک آرایه خالی تعریف می‌کند. البته سازنده کلاس را می‌توان به‌صورت متدی هم‌نام با کلاس (با رعایت بزرگی و کوچکی حروف) نیز تعریف‌کرد اما باید مراقب‌باشیم که درصورت تغییر نام کلاس، سازنده را نیز تغییر نام دهیم. درنتیجه پیشنهاد می‌کنیم برای سهولت بیشتر، از همان کلمه **__construct** استفاده‌کنید.

دسترسی به فیلدها و متدها در داخل کلاس

آیا از اینکه چطور یک متد می‌تواند به فیلدهای کلاس دسترسی پیدا کند تعجب کرده‌اید؟ برای مثال، کد زیر را ببینید:

```
public function SetBody($body) {
    $this->body = $body;
}
```

همان‌طور که می‌دانید، ما یک فیلد خصوصی (**private**) به‌نام **$body** در کلاس خود تعریف کرده‌ایم و اگر بخواهیم آنرا درون متدها به‌کار گیریم، باید توسط متغیر ویژه‌ای موسوم‌به **$this** به آن دسترسی پیدا کنیم. **$this** به‌معنای یک اشاره‌گر به شئ جاری است که از کلاس ایجاد شده‌است. درواقع **$this** در زمانی که متد **SetBody** (یا هر متد دیگری از کلاس) ازطریق شئ ایجادشده از کلاس فراخوانی شود، به همان شئ اشاره می‌کند. بنابراین **$this->body** یعنی فیلد **body** از شیئی که متد **SetBody** را برروی آن فراخوانی کرده‌ایم. بعد از نوشتن اسم شئ، برای دسترسی به عناصر داخلی آن (مثل فیلدها یا متدها) باید از علامت **->** (یعنی کارکترهای - یا خط فاصله (تفریق) و سپس **>** یا علامت بزرگتر) استفاده‌کنیم که در مجموع شکل یک فلش را می‌سازند. بنابراین اگر بخواهیم دستور موجود در متد **SetBody** را بخوانیم، باید اینگونه توضیح‌دهیم:

«فیلد **$body** از شئ جاری را با پارامتر **$body** که برای متد **SetBody** ارسال شده‌است، مقداردهی کن».

دقت‌کنید که برای دسترسی به فیلدها ازطریق شئ، نیازی نیست و نباید از کارکتر **$** بعد از **->** استفاده‌کنیم و مفسر **PHP** بطور خودکار متوجه می‌شود که این عنصر یک فیلد است یا یک متد (از کجا می‌فهمد؟ خیلی ساده است: بعد از اسم متدها پرانتزها می‌آیند!).

مشابه فیلدها، برای دسترسی به هر عنصری از کلاس در درون کلاس از این ساختار می‌توانیم استفاده‌کنیم. برای مثال، در متد **SetBody** می‌توانیم با کمک دستور **$this->SetSubject($this->body);** متد **SetSubject** شئ مربوطه را فراخوانی کنیم و فیلد **$body** همان شئ را بعنوان پارامتر برای آن بفرستیم.

نکته مهم: دقت‌کنید که کلمه کلیدی **$this** فقط درون متدهای یک کلاس قابل استفاده است. آن‌هم نه همه کلاس‌ها: فقط کلاس‌هایی که بصورت **static** (ایستا) تعریف نشده‌اند (یعنی همان کلاس‌های معمولی). ضمناً کلمه **$this** را خارج از بدنه کلاس نمی‌توانید به‌کار ببرید. درباره مفاهیم **private** و **public** و... در ادامه همین جلسه و در بخش اصلاح‌گرها توضیح خواهیم‌داد. مفهوم کلاس‌های **static** نیز در جلسات آینده مورد بررسی دقیق قرار خواهد گرفت.

استفاده از یک شئ

حال اجازه‌دهید به‌کمک یک شئ از کلاس **Emailer** خودمان، یک ایمیل ارسال کنیم. اما صبر کنید! یک کلاس که به‌تنهایی قابل‌استفاده نیست (مثال نقشه خانه را به‌یاد بیاورید: آیا می‌توانید روی نقشه خانه بنشینید و از تماشای تلویزیون لذت ببرید، درحالی‌که واقعاً خانه ندارید؟!). بنابراین ابتدا باید یک شئ از کلاسی که تعریف کرده‌ایم ایجاد کنیم (یک خانه مطابق نقشه موجود بسازیم). برای این‌کار از کلمه کلیدی **new** استفاده می‌کنیم:

```
<?php
    $emailerObject = new Emailer('mmshfe@gmail.com');
?>
```

بدین‌ترتیب، یک شئ به‌نام **$emailerObject** از کلاس **EMailer** ساخته می‌شود و بطور خودکار متد سازنده کلاس فراخوانی می‌شود. از آنجا که سازنده‌ای که برای کلاس **EMailer** نوشتیم، یک پارامتر ورودی دارد، در اینجا نیز موقع ایجاد یک شئ از کلاس، باید یک آرگومان برای آن ارسال‌کنیم. حال اجازه‌دهید با استفاده از این شئ، یک ایمیل بفرستیم:

```
<?php
    $emailerObject = new Emailer('mmshfe@gmail.com');
    $emailerObject->AddRecipient('info@xoogle.ir');
    $emailerObject->SetSubject('Test');
    $emailerObject->SetBody('Hi, I\'m a test email!');
    $emailerObject->SendEmail();
?>
```

ساده است، اینطور نیست؟ ابتدا با فراخوانی متد **AddRecipient** یک گیرنده به پیام موردنظرمان اضافه کردیم. سپس با فراخوانی متد **SetSubject** عنوان **Test** را برای آن برگزیدیم. در ادامه ازطریق فراخوانی متد **SetBody** متن **Hi, I'm a test email!** را بعنوان بدنه پیام قرارداده و نهایتاً با فراخوانی متد **SendEmail** پیام را ارسال کردیم.

امتیاز اصلی شئ‌گرایی، کد تمیزتر و مرتب‌تر است. اکنون در پروژه خودمان هرگاه نیاز به ارسال ایمیل داشته‌باشیم، به‌راحتی یک شئ از کلاس **EMailer** ایجادکرده و با تنظیم پارامترهای آن، اقدام به ارسال ایمیل می‌کنیم. هرگاه نیاز به تغییر الگوریتم ارسال ایمیل باشد، کافی‌است کد موجود در کلاس **EMailer** اصلاح شود و بلافاصله تمامی بخش‌هایی از کد که از این کلاس استفاده کرده‌اند، الگوریتم جدید را به‌کار خواهند گرفت.

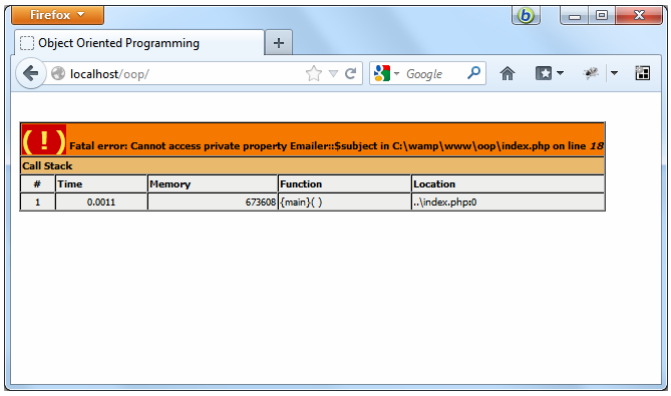
اصلاح‌گرها

قبلاً دیدیم که از برخی کلمات کلیدی مثل **private** یا **public** در کلاس استفاده می‌شود. واقعاً این عبارات چه‌هستند و چرا به آنها نیاز داریم؟ این کلمات کلیدی اصلاح‌گر نام‌دارند و در **PHP5** معرفی شده‌اند (بنابراین در **PHP4** نمی‌توانید از آنها استفاده‌کنید – قبلاً گفتیم که **PHP4** پشتیبانی قابل‌قبولی از شئ‌گرایی ارائه نمی‌دهد). این کلمات کلیدی به شما کمک می‌کنند تا سطح دسترسی استفاده‌کنندگان از این کلاس (بخش‌هایی از کد که یک شئ از این کلاس ایجاد می‌کنند) را مشخص‌کنید. اجازه‌دهید نحوه کار واقعی این اصلاح‌گرها را ببینیم:

۱- **private** (خصوصی) : فیلدها و متدهایی که بصورت خصوصی تعریف می‌شوند، اجازه دسترسی از بیرون کلاس را ندارند. البته هر متدی درون کلاس می‌تواند به آنها دسترسی داشته‌باشد و درواقع فقط از بیرون کلاس (ازطریق اشیاء ایجادشده از کلاس) قابل دسترسی نیستند. برای مثال:

```
<?php
    $emailerObject = new EMailer('mmshfe@gmail.com');
    $emailerObject->subject = 'Test';
?>
```

اجرای این کد موجب بروز خطای زیر می‌شود:



بله، همان‌طور که می‌بینید، نمی‌توانیم به فیلد خصوصی **$subject** دسترسی پیدا کنیم (اگر این‌طور بود، آیا می‌توانستیم به آن لقب خصوصی بدهیم؟!). اما این کد هیچ مشکلی ندارد:

```
<?php
    $emailerObject = new EMailer('mmshfe@gmail.com');
    $emailerObject->SetSubject('Test');
?>
```

زیرا متد **SetSubject** یک متد خصوصی نیست. درست است که درون بدنه متد **SetSubject** این کد نوشته شده‌است:

```
public function SetSubject($subject) {
    $this->subject = $subject;
}
```

و این متد به‌کمک کلمه کلیدی **$this** به فیلد **subject** شئ جاری (یعنی همان **$emailerObject**) دسترسی پیدا می‌کند، اما نباید فراموش‌کنیم که این متد، درون بدنه کلاس قرار دارد (بعد از آکولاد باز **{** ابتدای تعریف کلاس و قبل از آکولاد بسته **}** انتهای کلاس). درواقع محل دستیابی به فیلد خصوصی **$subject** داخل کلاس است نه بیرون از کلاس.

درواقع فیلدها و متدهای خصوصی، فقط درون کلاس قابل مشاهده و تغییر هستند و از بیرون کلاس، نه دیده می‌شوند و نه می‌توان آنها را تغییر داد و تنها راه مشاهده یا ویرایش آنها، وجود یک متد عمومی است که کار موردنظر را برای ما انجام دهد (مثل **SetSubject**).

۲- **public** (عمومی): هر عنصری از کلاس (اعم از فیلد یا متد) که بصورت عمومی تعریف‌شود، به‌راحتی توسط اشیاء ایجادشده از کلاس (از بیرون کلاس) قابل دسترسی است.

حال شاید بپرسید اصلاً چرا باید یک عنصر کلاس را بصورت خصوصی تعریف کنیم؟ جواب ساده است: امنیت! اگر عنصر مربوطه خصوصی نباشد، امکان تغییر آن به هر مقداری ازطریق اشیاء ایجادشده از کلاس وجود دارد. فرض‌کنید کلاسی می‌نویسید که فهرست برندگان قرعه‌کشی را از پایگاه داده‌ها استخراج‌کرده و به کاربر نشان می‌دهد. طبیعتاً این فهرست را در یک فیلد ذخیره می‌کنید. حال اگر این فیلد عمومی باشد، هر کسی می‌تواند با ایجاد یک شئ از کلاس و اضافه‌کردن اسم خود به آرایه موجود در آن فیلد، خود را جزو برندگان قرعه‌کشی اعلام‌کند؛ درحالی‌که اگر این فیلد خصوصی باشد، چنین امکانی وجود نخواهد داشت. اما یک سؤال در اینجا مطرح می‌شود: اگر این فیلد خصوصی باشد، استفاده‌کنندگان از کلاس، چگونه فهرست اسامی برندگان را مشاهده‌کنند؟ باز هم پاسخ ساده است: یک متد عمومی می‌نویسیم که به فیلد خصوصی دسترسی داشته و عناصر آنرا چاپ‌کرده یا بعنوان نتیجه با کلمه کلیدی **return** بازمی‌گرداند. بدین‌ترتیب، از آنجا که متد شما فقط محتوای فیلد را اعلام می‌کند و اجازه تغییر آنرا نمی‌دهد، هم توانسته‌اید فهرست اسامی برندگان را اعلام‌کنید و هم جلوی تغییر آنرا گرفته‌اید.

مثال دیگر، زمانی است که شما قصد دارید کلاسی تعریف‌کنید که مختصات یک نقطه را در صفحه‌نمایش نشان می‌دهد. حال اگر تراکم صفحه‌نمایش (**Resolution**) شما برروی **1024x768** تنظیم شده‌باشد، آیا مختصات نقطه‌ای بصورت **(-5, 900)** قابل‌قبول است؟ طبیعتاً پاسخ شما منفی است. اما اگر فیلدهای مربوطبه مؤلفه‌های **X** و **Y** را عمومی تعریف‌کنیم، امکان مقداردهی آن به هر عدد مجاز در **PHP** وجود دارد و اعداد **-5** و **900** هم اعدادی مجاز هستند. مسئله اینجاست که این اعداد در کلاس ما قابل پذیرش نیستند. برای رفع این مشکل نیز فیلدهای مذکور را خصوصی تعریف می‌کنیم و متدهای عمومی می‌نویسیم که یک پارامتر دریافت‌کرده و پس‌از اعتبارسنجی لازم و تأیید قرارگرفتن پارامتر در محدوده مجاز، فیلد خصوصی مربوطه را با مقدار پارامتر دریافتی مقداردهی می‌کنند و درصورت غیرمعتبربودن پارامتر، فیلد را تغییر نمی‌دهند. بدین‌ترتیب، یک مکانیزم امن برای اصلاح فیلدها درنظر گرفته‌ایم و مطمئن هستیم که همیشه از کلاس ما به‌شکل صحیح استفاده می‌شود.

۳- **protected** (محافظت‌شده) : این اصلاح‌گر معنای خاصی در **OOP** دارد. اگر یک عنصر کلاس بصورت محافظت‌شده تعریف‌شود، فقط می‌توانید از آن در خود کلاس استفاده‌کنید و ازطریق اشیاء (بیرون از کلاس) قابل‌استفاده نیست. صبر کنید! این تعریف، همان تعریف اصلاح‌گر خصوصی است! پس تفاوت در کجاست؟ تفاوت اصلاح‌گر محافظت‌شده با اصلاح‌گر خصوصی در آن است که کلاس‌های فرعی که از کلاس اصلی مشتق می‌شوند نیز توانایی دسترسی به عناصر محافظت‌شده آنرا خواهند داشت. البته باز هم فقط در بدنه کلاس‌های فرعی و نه در اشیاء ایجادشده از آنها. درمورد کلاس‌های اصلی (والد) و مشتق‌شده (فرزند) به‌تفصیل در مبحث وراثت توضیح خواهیم‌داد. فعلاً ذهنتان را درگیر این اصلاح‌گر نکنید تا به‌موقع در بخش مربوطه، توضیحات مفصل آنرا به شما ارائه‌دهیم.

سازنده‌ها و مخرب‌ها

ما قبلاً درباره سازنده در همین جلسه صحبت کردیم و گفتیم سازنده متد خاصی است که بطور خودکار در زمان ایجاد یک شئ از کلاس فراخوانی می‌شود. برای مثال، کلاس زیر را مشاهده‌کنید:

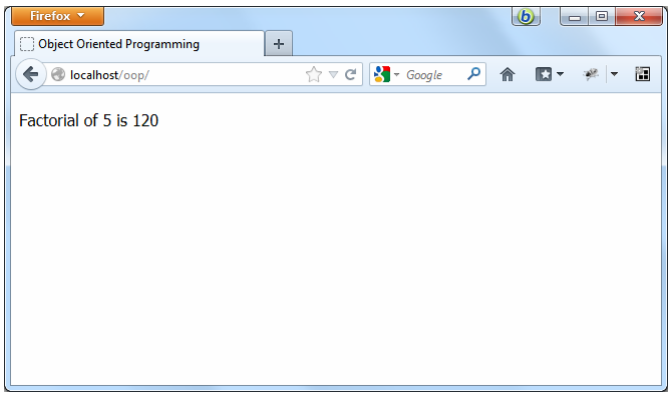
```php
<?php
    class Factorial {
        private $result;
        private $number;
        public function __construct($number) {
            $this->result = 1;
            $this->number = $number;
            for($i = 2; $i <= $number; $i++) {
                $this->result *= $i;
            }
        }
        public function ShowResult() {
            echo "<p>Factorial of {$this->number} is {$this->result}</p>" . PHP_EOL;
        }
    }
?>
```

همان‌طور که ملاحظه می‌کنید، کلاس فوق، وظیفه محاسبه فاکتوریل یک عدد را برعهده دارد (حاصل‌ضرب اعداد از ۱ تا خود عدد).

حال اجازه‌دهید نحوه استفاده از این کلاس را مشاهده‌کنیم:

```
<?php
    $fact = new Factorial(5);
    $fact->ShowResult();
?>
```

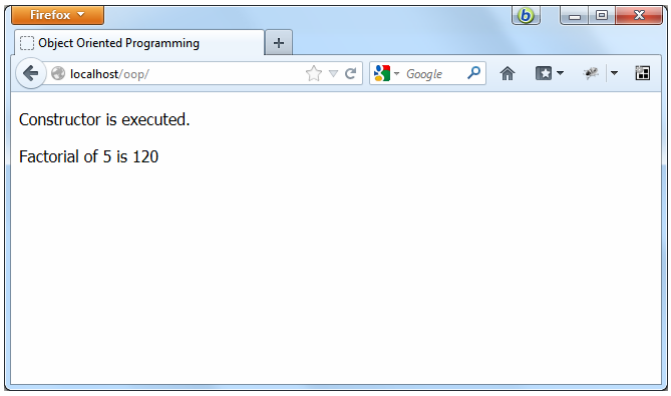
و خروجی بدست‌آمده بصورت زیر است:



گمان نمی‌کنیم مشکلی با درک نحوه کار کد فوق داشته‌باشید. حال اجازه‌دهید یک دستور به سازنده اضافه‌کنیم تا هربار سازنده فراخوانی می‌شود، پیغامی را چاپ‌کند:

```
public function __construct($number) {
    $this->result = 1;
    $this->number = $number;
    for($i = 2; $i <= $number; $i++) {
        $this->result *= $i;
    }
    echo '<p>Constructor is executed.</p>' . PHP_EOL;
}
```

حال اگر مجدداً کد استفاده از کلاس را اجرا کنیم، خروجی زیر را خواهیم‌داشت:

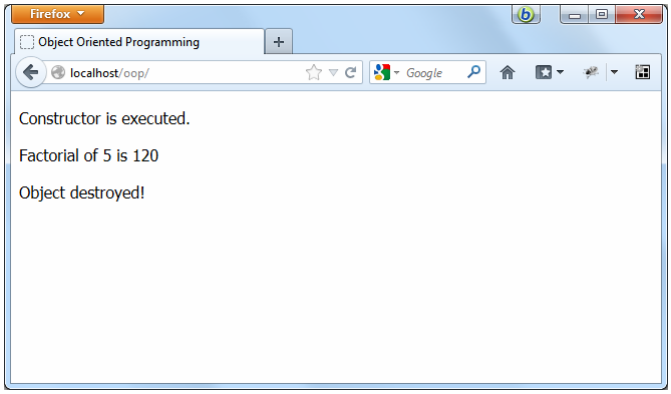


همان‌طور که مشاهده می‌کنید، هم‌زمان با ایجاد شئ از کلاس، سازنده فراخوانی شده‌است.

درست مشابه سازنده، متد دیگری به‌نام مخرب (**Destructor**) وجود دارد که بطور خودکار در زمان ازبین‌رفتن یک شئ از کلاس اجرا می‌شود. برای ایجاد یک متد مخرب باید از نام **__destruct** (دوبار کارکتر _ (**UnderScore**) و سپس کلمه (**destruct**) استفاده‌کنید. این متد هیچ پارامتر ورودی ندارد و برخلاف سازنده که می‌توانستیم بجای کلمه **__construct** از نام کلاس استفاده‌کنیم، تنها نام مجاز آن، **__destruct** است. این متد همان‌گونه که ذکر شد، در زمان تخریب یا ازبین‌رفتن یک شئ (مثلاً با پایان‌یافتن اسکریپت یا حذف شئ توسط تابع **unset**) اجرا می‌شود. برای مثال، این کد را به کلاس **Factorial** اضافه‌کنید:

```
public function __destruct() {
    echo '<p>Object destroyed!</p>' . PHP_EOL;
}
```

و مجدداً کد را اجرا کنید. خروجی به‌دست آمده به‌صورت زیر خواهد بود:



مشاهده می‌کنید که در پایان اجرای اسکریپت (یا هر زمان که خودمان شئ را با کمک تابع **unset** حذف‌کنیم)، متد مخرب بطور خودکار فراخوانی می‌شود. دقت‌کنید که وجود سازنده اجباری و وجود مخرب اختیاری است. اما واقعاً چه‌زمانی باید از مخرب استفاده‌کرد؟ این مسئله بستگی به شما بعنوان برنامه‌نویس دارد اما بطور کلی هرگاه منابعی توسط کلاس در اختیار گرفته شده‌باشد که بطور خودکار آزاد نشود (مثل فایل‌ها و... که درصورت آزادنشدن، تا پایان اجرای اسکریپت آزاد نمی‌شوند)، بهترین‌مکان جهت آزادسازی آنها، متد مخرب کلاس است. بدین‌ترتیب، هرگاه شئ ایجادشده از کلاس ازبین برود، تمام منابعی را که در اختیار گرفته‌بود، بطور خودکار آزاد خواهد کرد.

جمع‌بندی

در این جلسه، با نحوه تعریف یک کلاس و ایجاد شئ از آن آشنا شدیم و همچنین روش تعیین سطح دسترسی به عناصر کلاس را نیز ازطریق اصلاح‌گرها آموختیم. انتظار می‌رود که در پایان این جلسه، بتوانید کلاس‌های ساده دلخواه خود را ایجاد کنید و پروژه‌های ساده‌ای را به‌روش شئ‌گرا بنویسید. در جلسه آینده درخصوص مسائل پیشرفته‌تر شئ‌گرایی مثل تعریف ثابت‌ها در کلاس و عناصر ایستا (**static**) در کلاس و مباحث ابتدایی وراثت توضیحاتی را ارائه خواهیم کرد.

فیلدها و متدهای ایستا (Static)

کلمه کلیدی **static** در برنامه‌نویسی شئ‌گرا اهمیت بسیار زیادی دارد. فیلدها و متدهای ایستا نقش اساسی در الگوی طراحی برنامه ایفا می‌کنند. حال ببینیم فیلدها و متدهای ایستا چه هستند؟ تاکنون مشاهده‌کردید که برای دسترسی به هر فیلد یا متد یک کلاس باید یک شئ از آن کلاس ایجاد کنیم (با کمک کلمه کلیدی **new**). اما فیلدها و متدهای ایستا از این جهت، با فیلدها و متدهای معمولی متفاوتند. شما می‌توانید به یک فیلد یا متد ایستا بصورت مستقیم و بدون ایجاد هرگونه شئ از کلاس مربوطه، دسترسی پیدا کنید. عناصر ایستا مشابه یک عضو سراسری برای کلاس عمل می‌کنند و تمام اشیاء ایجادشده از آن کلاس می‌توانند به آنها دسترسی پیداکنند. بعلاوه، فیلدهای ایستا همیشه آخرین وضعیت (مقدار) خود را حفظ می‌کنند. برای مثال، یک شئ از کلاس می‌تواند یک فیلد ایستا را مقداردهی کند و شئ دیگر، مقدار آنرا بخواند که این ویژگی، در موارد خاصی بسیار سودمند است.

شاید بپرسید چرا باید از یک متد ایستا استفاده کنیم؟ واقعیت آن‌است که اغلب متدهای ایستا مشابه متدهای خصوصی عمل می‌کنند و به بقیه متدها خدمت‌رسانی می‌کنند. متدهای ایستا یک وظیفه بسیار خاص و پیچیده را انجام می‌دهند یا یک مقدار یا شئ خاص را بعنوان نتیجه باز می‌گردانند. بنابراین، ایجاد یک شئ از کلاس مربوطه در هربار نیاز به اینگونه متدها، منابع زیادی از سرور را مصرف خواهد نمود. درمورد کاربردهای خاص این متدها در الگوهای طراحی، در آینده نزدیک مثال‌های عملی مطرح خواهیم‌کرد. فعلاً اجازه‌دهید مثالی ساده از کاربرد متدهای ایستا را مشاهده‌کنیم:

```
class Math {
    public static function Abs($number) {
        if($number < 0) {
            return ($number * -1);
        }
        return $number;
    }

    public static function Power($x, $y) {
        $result = 1;
        for($i = 0; $i < $y; $i++) {
            $result *= $x;
        }
        return $result;
    }

    public static function Square($number) {
        return ($number * $number);
    }
}
```

همان‌طور که ملاحظه می‌کنید، کلاس فوق حاوی سه متد **static** (ایستا) است که به‌ترتیب قدرمطلق یک عدد، حاصل یک عدد به‌توان عدد دیگر و مربع یک عدد را باز می‌گردانند. حال ببینیم چگونه از متدهای این کلاس استفاده می‌شود:

```
echo Math::Abs(-5);     // output: 5
echo Math::Power(2, 3); // output: 8
echo Math::Square(100); // output: 10000
```

مشاهده می‌کنید که هیچ شئ خاصی از کلاس **Math** ایجاد نکردیم. درواقع مزیت اصلی متدهای ایستا نیز همین مسئله است. برای دسترسی به آنها، از نام کلاس و علامت **::** استفاده می‌کنیم و از ساختار نام شئ و **->** و سپس نام متد بهره نمی‌جوییم. بدین‌ترتیب، در مصرف حافظه و منابع سرور صرفه‌جویی به‌عمل می‌آید. البته این ساختار فقط در مواقعی که متد موردنظر یک کار خاص را انجام‌داده و تمام می‌شود، سودمند است و طبیعتاً اگر بخواهیم نتیجه را در یک فیلد ذخیره‌کنیم و در بخش‌های دیگر کد به‌کار ببریم، متدهای ایستا نمی‌توانند به ما کمک‌کنند. از آنجا که متدهای ایستا با استفاده از کلاس فراخوانی می‌شوند و نه شئ، طبیعتاً در متدهای ایستا فقط می‌توان به فیلدهای ایستا دسترسی پیدا نمود و فیلدهای معمولی قابل دسترسی نیستند. این امر کاملاً منطقی است زیرا هدف متدهای ایستا، فراخوانی بدون شئ است و وقتی شئ وجود ندارد، خاصیت (فیلد) هم وجود نخواهد داشت. نکته مهم بعدی آن‌است که متغیر **$this** درون متدهای ایستا قابل دسترسی نیست. این مسئله نیز به‌دلیل عدم وجود شئ از کلاس است. البته می‌توان به‌راحتی در هر بخش از کلاس، با کمک کلمه کلیدی **self** و **::** (**self::**) به هر عنصر دلخواه ایستا (**static**) دسترسی پیدا نمود.

اجازه‌دهید مثالی از نحوه کاربرد فیلدهای ایستا نیز مشاهده‌کنیم:

```
ObjectCounter::StaticGetObjectCount();  // output: Current object count: 0
$obj1 = new ObjectCounter();
$obj1->GetObjectCount();                // output: Current object count: 1
ObjectCounter::StaticGetObjectCount();  // output: Current object count: 1
$obj2 = new ObjectCounter();
$obj1->GetObjectCount();                // output: Current object count: 2
ObjectCounter::StaticGetObjectCount();  // output: Current object count: 2
$obj2->GetObjectCount();                // output: Current object count: 2
$obj3 = new ObjectCounter();
ObjectCounter::StaticGetObjectCount();  // output: Current object count: 3
```

مشاهده می‌کنید که هم متدهای ایستا و هم متدهای معمولی می‌توانند به فیلدهای ایستا دسترسی پیدا کنند. ازطرفی، فیلدهای ایستا مقدار مشترکی در تمامی اشیاء دارند و می‌توان از آنها برای به‌اشتراک‌گذاری مقادیر بین اشیاء یک کلاس استفاده‌نمود.

نکته مهم: عناصر ایستا برنامه‌نویسی شئ گرا را بسیار شبیه برنامه‌نویسی سنتی و رویه‌گرا می‌کنند: بدون ساخت اشیاء می‌توان مستقیماً هر تابعی را صدا زده و به هر فیلدی دسترسی پیدا نمود. به همین علت، باید از عناصر ایستا با دقت استفاده‌شود. زیاده‌روی در عناصر ایستا اصلاً توصیه نمی‌شود. از این عناصر فقط در زمانی که منظور خاص و توجیه قابل قبولی وجود دارد، استفاده کنید.

ثابت‌ها در کلاس

خوشبختانه تا اینجا می‌دانید که می‌توان با استفاده از دستور **define** در اسکریپت‌های **PHP** یک ثابت تعریف نمود. درمورد کلاس‌ها کمی اوضاع متفاوت است. برای تعریف یک ثابت در داخل کلاس، از کلمه کلیدی **const** (مخفف **Constant**) استفاده می‌کنیم. ثابت‌های تعریف‌شده درون کلاس با ثابت‌های عمومی تفاوت دارند و از ثابت‌های تعریف‌شده درون کلاس، فقط می‌توان در داخل کلاس یا ازطریق اشیاء ایجادشده از کلاس استفاده‌نمود. البته ثابت‌های کلاس نیز مشابه ثابت‌های عمومی، غیرقابل تغییر هستند. برای درک بهتر نحوه استفاده از ثابت‌ها در کلاس، به مثال زیر دقت‌کنید:

```
class WordCounter {
    // You should not use $ sign before constants
    const ASC = 1;
    const DESC = 2;
    private $words;

    public function __construct($filename) {
        $file_content = file_get_contents($filename);
        $this->words = (array_count_values(str_word_count(strtolower($file_content), 1)));
    }

    public function Count($order) {
        if($order == self::ASC) {
            asort($this->words);
        }
        elseif($order == self::DESC) {
            arsort($this->words);
        }
        foreach($this->words as $word => $count) {
            echo $word . ' = ' . $count . '<br/>' . PHP_EOL;
        }
    }
}
```

این کلاس **WordCounter** تعداد تکرار کلمات را درون یک فایل مشخص می‌شمارد. در اینجا ما دو ثابت به اسامی **ASC** و **DESC** به‌ترتیب با مقادیر ۱ و ۲ تعریف کرده‌ایم. برای دسترسی به این ثابت‌ها از درون کلاس، باید به آنها با کمک کلمه کلیدی **self** دسترسی پیدا کنیم. دقت‌کنید که به آنها توسط **::** دستیابی پیدا می‌کنیم (نه **->**) زیرا این ثابت‌ها در عمل مشابه یک عنصر ایستا عمل‌می‌کنند. سازنده کلاس فوق، ابتدا محتوای فایل اعلام‌شده توسط پارامتر دریافتی را توسط تابع **file_get_contents** دریافت‌کرده و در متغیر **$file_content** قرار می‌دهد. سپس ابتدا تمام محتوا را توسط تابع **strtolower** به حروف کوچک تبدیل‌نموده و در ادامه، آنرا به‌کمک تابع **str_word_count** کلمه به کلمه بصورت یک آرایه در اختیار تابع **array_count_values** قرار می‌دهد تا این تابع نیز تعداد تکرار هرکدام از عناصر آرایه را بدست‌آورد. بدین‌ترتیب، پس از اجرای سازنده کلاس فوق، فیلد خصوصی **$words** حاوی آرایه‌ای انجمنی خواهد بود که در آن، اندیس هر خانه از آرایه، یکی از کلمات فایل اعلام‌شده و مقدار آن، تعداد تکرار کلمه مربوطه در درون فایل خواهد بود. متد عمومی **Count** نیز ابتدا برحسب اینکه پارامتر دریافتی برابر با ثابت **ASC**

(مقدار ۱) یا **DESC** (مقدار ۲) است، به‌ترتیب آرایه موجود در فیلد خصوصی **$words** را بصورت صعودی (تابع **asort**) و یا نزولی (تابع **arsort**) مرتب‌کرده و به‌کمک حلقه **foreach** آنرا نمایش می‌دهد.

حال اجازه‌دهید نحوه استفاده از کلاس فوق را نیز بررسی کنیم:

```
require_once 'class.wordcounter.php';
if(file_exists('words.txt')) {
    $wCounter = new WordCounter('words.txt');
    $wCounter->Count(WordCounter::DESC);
}
```

مشاهده می‌کنید که با ایجاد یک شئ از کلاس **WordCounter** و ارسال نام فایل **words.txt** (با فرض اینکه چنین فایلی وجود داشته‌باشد)، محتوای فایل مذکور خوانده‌شده و درون فیلد خصوصی **$words** شئ مربوطه (**$wCounter**) قرار می‌گیرد. سپس با فراخوانی متد عمومی **Count** برروی شئ فوق‌الذکر و ارسال ثابت **DESC** از کلاس **WordCounter** (یعنی **WordCounter::DESC**)، خروجی موردنظر یعنی نمایش تعداد تکرار هر کلمه در فایل **words.txt** را بصورت نزولی خواهیم‌داشت.

دقت‌کنید که برای دسترسی به ثابت‌های کلاس در بیرون از کلاس باید از نام کلاس و عملگر **::** استفاده‌کنیم و از ساختار نام شئ و سپس عملگر **->** خبری نخواهد بود. اگر دقت‌کنید، این ساختار بسیار شبیه فیلدهای ایستا است، با این تفاوت که ثابت‌ها فقط‌خواندنی هستند. حال اجازه‌دهید کلاس فوق را در عمل تست کنیم. یک فایل با نام **words.txt** با محتوای زیر ایجاد کنید:

```
Wordpress is an open source blogging engine. If you are not familiar with blogging, it is something like keeping a diary on the web. A blog stands for web log. Wordpress is totally free and released under the GPL.
```

حال با اجرای کد فوق، نتیجه زیر به‌دست خواهد آمد:

```
is = 3
the = 2
web = 2
blogging = 2
wordpress = 2
a = 2
stands = 1
blog = 1
for = 1
on = 1
totally = 1
under = 1
gpl = 1
released = 1
and = 1
diary = 1
free = 1
log = 1
like = 1
engine = 1
if = 1
source = 1
open = 1
an = 1
you = 1
are = 1
it = 1
something = 1
with = 1
familiar = 1
not = 1
keeping = 1
```

یک ابزار سودمند؛ نظر شما چیست؟

متدهای دستیاب

متدهای دستیاب را می‌توان به‌سادگی متدهایی تعریف‌کرد که وظیفه خواندن یا مقداردهی هرکدام از فیلدهای کلاس را برعهده دارند. عادت خوبی است که به‌جای دسترسی مستقیم به فیلدها ازطریق عمومی‌سازی آنها، فیلدهای کلاس را بصورت خصوصی تعریف‌کرده و برای کلاس، متدهای دستیاب را بصورت عمومی تعریف‌کنیم.

بطور کلی دو گروه دستیاب وجود دارد: یکی برای خواندن (موسوم به **Getter**) و دیگری برای نوشتن (معروف به **Setter**). در

نتیجه، برای هر فیلد باید هر دو متد نوشته‌شود. برای مثال، به کلاس زیر دقت‌کنید:

```php
class Student {
    private $name;
    private $grade;

    public function __construct() {
        $name = '';
        $grade = 0;
    }

    public function GetGrade() {
        return $this->grade;
    }

    public function GetName() {
        return $this->name;
    }

    public function SetGrade($grade) {
        if($grade >= 0 && $grade <= 20) {
            $this->grade = $grade;
        }
    }

    public function SetName($name) {
        if(mb_strlen($name, 'utf-8') < 20) {
            $this->name = $name;
        }
    }
}

// Usage:
$st = new Student();
$st->SetName('Alireza');
$st->SetGrade(17.5);
echo 'Name: ' . $st->GetName() . '<br/>' . PHP_EOL;
echo 'Grade: ' . $st->GetGrade() . '<br/>' . PHP_EOL;
/* Output:
Name: Alireza
Grade: 17.5
*/
```

در مثال فوق، دو متد **Getter** و دو متد **Setter** وجود دارد. در نامگذاری متدهای دستیاب، اغلب از کلمه **Get** در ابتدای **Getter** و از کلمه **Set** در ابتدای **Setter** استفاده می‌شود.

حال شاید بپرسید چرا باید اینقدر کارهای اضافه انجام‌دهیم، درحالی‌که می‌توانیم به‌راحتی فیلدهای موردنظر را عمومی‌کنیم؟ درحقیقت هدف از کاربرد متدهای دستیاب در بیشتر موارد، دستیاب **Setter** است؛ زیرا مقداردهی یک فیلد ازطریق یک متد، به ما امکان بررسی اعتبار مقدار موردنظر را می‌دهد. اما شاید بپرسید آیا واقعاً با این روش، اگر کلاس ما ۵۰ فیلد داشته‌باشد، نیاز به نوشتن ۱۰۰ متد دستیاب داریم؟ سؤال خوبی مطرح کردید! خوشبختانه **PHP** به‌اندازه کافی مهربان هست که شما را از این کار خسته‌کننده نجات‌دهد. چطور؟ با استفاده از متدهای جادویی! این متدها استرس و فشار کاری شما را تا ۹۰ درصد کاهش می‌دهند. باور نمی‌کنید؟ خواهیم‌دید.

استفاده از متدهای جادویی برای دستیابی به فیلدهای کلاس

برای جلوگیری از انجام کارهای تکراری و خسته‌کننده جهت نوشتن متدهای دستیاب **Getter** و **Setter** برای تک‌تک فیلدهای کلاس، **PHP** به شما امکان سربارگذاری (**OverLoad**) دستیاب‌ها را می‌دهد. سربارگذاری به‌طور ساده به عملی می‌گویند که در آن، یک متد براساس پارامترهای مختلفی که دریافت می‌کند، کارهای متفاوتی انجام‌دهد. متدهای جادویی سربارگذاری دستیاب‌های **Getter** و **Setter** به‌ترتیب **__Get** و **__Set** (دوبار کارکتر **Underscore** (_) و سپس کلمه **Get** یا **Set**) نام‌دارند. برای استفاده از این متدها، باید ابتدا تمامی فیلدهای کلاس را درون یک آرایه تعریف‌کنیم. درحقیقت کلاس فقط یک فیلد از نوع آرایه خواهد داشت که هر کدام از خانه‌های آرایه فوق، یکی از فیلدهای کلاس می‌باشند و اسم هر فیلد، اندیس خانه مربوطه در آرایه مذکور را تشکیل می‌دهد. برای درک بهتر، به مثال زیر که نمونه بازنویسی‌شده کلاس قبلی **Student** با این روش است، دقت‌کنید:

```php
class Student {
    private $fields;

    public function __construct() {
        $this->fields = array(
            'name' => '',
            'grade' => 0
        );
    }

    public function __Get($fieldName) {
        if(array_key_exists($fieldName, $this->fields)) {
            return $this->fields[$fieldName];
        }
        return '';
    }

    public function __Set($fieldName, $value) {
        if(array_key_exists($fieldName, $this->fields)) {
            switch(strtolower($fieldName)) {
            case 'name':
                if(mb_strlen($value, 'utf-8') < 20) {
                    $this->fields['name'] = $value;
                }
                break;
            case 'grade':
                if($value >= 0 && $value <= 20) {
                    $this->fields['grade'] = $value;
                }
                break;
            }
        }
    }
}

// Usage:
$st = new Student();
$st->name = 'Alireza';
$st->grade = 17.5;
echo 'Name: ' . $st->name . '<br/>' . PHP_EOL;
echo 'Grade: ' . $st->grade . '<br/>' . PHP_EOL;
/* Output:
Name: Alireza
Grade: 17.5
*/
```

همان‌طور که ملاحظه می‌کنید، نحوه استفاده از این کلاس کاملاً تغییر کرد و مشابه یک فیلد عمومی از عناصر **name** و **grade** استفاده می‌شود. نکته در اینجاست که با فراخوانی **$st->name** در وضعیت نوشتن (مقداردهی با کلمه **Alireza**)، دستیاب **__Set** بطور خودکار فراخوانی می‌شود و عبارت بعد از **->** یعنی **name** بعنوان پارامتر اول و عبارت بعد از **=** یعنی کلمه **Alireza** بعنوان پارامتر دوم برای آن ارسال می‌گردد. متد مذکور نیز بعد از بررسی وجود اندیس **name** در آرایه فیلدهای کلاس و اعتبارسنجی مقدار واردشده، خانه مربوطه را از آرایه **$fields** مقداردهی می‌کند. به‌همین‌ترتیب مقداردهی اندیس **grade** نیز انجام خواهد شد. در ادامه، زمانی‌که **$st->name** در وضعیت خواندن فراخوانی می‌شود (دستور **echo**)، بطور خودکار دستیاب **__Get** فراخوانی‌شده و عبارت بعد از **->** یعنی **name** برای آن ارسال می‌شود و این متد نیز درصورت وجود اندیس مذکور، مقدار آن و درغیر اینصورت، یک رشته خالی را بازمی‌گرداند. به روش مشابه، خواندن اندیس **grade** نیز انجام می‌شود.

البته کد فوق را می‌توان به‌شکل ساده‌تری نیز نوشت:

```php
class Student {
    private $fields;

    public function __construct() {
        $this->fields = array();
    }

    public function __Get($fieldName) {
        return $this->fields[$fieldName];
    }

    public function __Set($fieldName, $value) {
        $this->fields[$fieldName] = $value;
    }
}
```

که البته در اکثر کتاب‌ها نیز به‌همین‌روش عمل شده‌است؛ اما شدیداً توصیه می‌کنیم که به روش اول یعنی روش کامل و همراه‌با اعتبارسنجی عمل‌کنید زیرا در روش دوم، هرگاه کدی مشابه **$st->family = 'Hoseini';** اجرا شود، اندیس مذکور به آرایه موجود در فیلد **$fields** اضافه خواهد شد درحالی‌که طراحی کلاس ما فاقد این عنصر بوده‌است. بنابراین امکان اضافه‌کردن مقادیر غیرمجاز در کلاس فراهم می‌شود؛ لذا مجدداً توصیه می‌شود تنها درصورتی‌که عمداً قصد استفاده از روش دوم (آزادگذاشتن درج عناصر جدید) مدنظر شما باشد، روش دوم را به‌کار گیرید و در سایر موارد از روش اول که روشی امن‌تر و حرفه‌ای‌تر است، استفاده‌نمایید. با استفاده از این دستیاب‌های جادویی، همان‌طور که ملاحظه می‌کنید، هم امکان کنترل صحت و اعتبار مقادیر فیلدها را خواهید داشت و هم اشیاء ایجادشده از کلاس، با مقادیر موردنظر خود مشابه فیلدهای عمومی معمولی برخورد می‌کنند و از پرانتزها و... در خارج از کلاس خبری نخواهد بود.

متدهای جادویی برای سربارگذاری متدهای کلاس

از سربارگذاری خوشتان آمده‌است؟ قصد داریم کاربردهای بیشتر آنرا نیز برای شما معرفی‌کنیم. درحقیقت مشابه دستیاب‌ها، هر متدی را در کلاس می‌توان سربارگذاری کرد. سربارگذاری متدها در **PHP** گام را فراتر از سربارگذاری در سایر زبان‌های برنامه‌نویسی مثل **C#** و... می‌نهد و به مرزی می‌رسد که می‌توانید حتی به متدهایی که وجود ندارند نیز دسترسی پیدا کنید! جالب است، نه؟ اجازه‌دهید نگاهی نزدیک‌تر به مسئله داشته‌باشیم.

یک متد جادویی وجود دارد که به شما امکان سربارگذاری هر متدی را در یک کلاس تحت **PHP5** می‌دهد. نام این متد جادویی، **__Call** است. این متد به شما قابلیت انجام عملیات دلخواه یا بازگشت مقادیر موردنظرتان را در زمان فراخوانی متدهایی که صراحتاً تعریف نشده‌اند، می‌دهد. همچنین علاوه‌بر قابلیت سربارگذاری متدها، امکان ارائه قابلیت مدیریت خطاهای احتمالی را نیز به شما می‌دهد. هر زمان که یک متد توسط یک شئ فراخوانی می‌شود که مستقیماً در بدنه کلاس تعریف نشده است، متد **__Call** (درصورت وجود) فراخوانی خواهد شد. این متد دو آرگومان دریافت می‌کند: اولی نام متد و دومی، آرایه‌ای از آرگومان‌هایی که باید به آن متد تعریف‌نشده، ارسال شوند. برای مثال، کد زیر را مشاهده‌کنید:

```php
class BlackHole {
    function __Call($method, $arguments) {
        echo "You called a method '{$method}' with the following arguments:<br/>" . PHP_EOL;
        echo '<pre>' . PHP_EOL;
        print_r($arguments);
        echo '</pre>' . PHP_EOL;
    }
}

// Usage:
$bh = new BlackHole();
$bh->Access(2, 3, 4);
$bh->NotAnyMethod('Hello World');
/* Output:
You called a method 'Access' with the following arguments:
Array
(
    [0] => 2
    [1] => 3
    [2] => 4
)
You called a method 'NotAnyMethod' with the following arguments:
Array
(
    [0] => Hello World
)
*/
```

اسم سیاهچاله برای کلاس فوق کاملاً مناسب است زیرا هیچ متدی با هیچ آرگومانی از دست آن نمی‌تواند فرار کند. درحقیقت اشیاء کلاس فوق برای هر متدی با هر نوع آرگومانی، یک پاسخ مناسب خواهند داشت. این بدان‌معنی است که شما همه آرگومان‌ها را بصورت یک آرایه در درون متد **__Call** در اختیار خواهید داشت. همچنین می‌توانید با بررسی آرگومان اول، نام تابع فراخوانی‌شده را بدست‌آورده و براساس ترکیب نام تابع و نوع آرگومان‌هایی که برای آن ارسال شده‌است، تصمیم بگیرید که چه کدی اجرا شود. البته

در **PHP** توابع جادویی زیادی وجود دارد که به‌مرور و با دنبال‌کردن این آموزش‌ها با آنها آشنا شده و وارد دنیای جادویی **PHP** خواهید شد. بی‌دلیل نیست که برخی از توسعه‌دهندگان وب به **PHP** لقب مرلین (**Merlin**) دنیای برنامه‌نویسی وب را داده‌اند.

جمع‌بندی

در این جلسه آموختیم که چگونه اشیاء دلخواه خود را بسازیم و با آنها کار کنیم. **PHP5** رشد بسیار زیادی در انواع مدل‌های اشیاء در مقایسه با **PHP4** داشته‌است. موتور شماره ۲ **Zend** که در هسته **PHP5** کار می‌کند، در مدیریت این ویژگی‌ها بسیار کارآمد عمل می‌کند و بهینه‌سازی بسیار خوبی در مصرف حافظه و پردازشگر سیستم‌های سرور به‌عمل می‌آورد.

در جلسه بعد بیشتر به اعمال **OOP** فرو رفته و با مباحثی همچون وراثت، کلاس‌های چکیده، رابط‌ها و... آشنا خواهیم‌شد؛ اما قبل‌از پیشروی، لطفاً هرآنچه را که تاکنون در این آموزش‌ها درباره شئ‌گرایی آموخته‌اید، تا حد ممکن تمرین‌کنید تا به تسلط کافی در آنها دست‌یابید. درغیر اینصورت در جلسات آینده ممکن‌است بخش‌هایی از مطالب را ابهام‌آمیز بدانید. بنابراین دست از تمرین بر ندارید. بعنوان یک تمرین خوب، سعی‌کنید همه کدهای قبلی خودتان را بصورت شئ‌گرا درآورید. هرچه بیشتر تمرین‌کنید، کارآیی و راندمان بالاتری کسب خواهید کرد.

توسعه یک کلاس (وراثت)

یکی‌از بهترین ویژگی‌های برنامه‌نویسی شئ‌گرا آن‌است که می‌توانید یک کلاس را توسعه داده و کلاسی کاملاً جدید بسازید. کلاس مشتق‌شده (فرزند) می‌تواند تمامی قابلیت‌های کلاس پایه (والد) را داشته‌باشد یا درصورت نیاز، برخی از آنها را بازنویسی (**Override**) نماید. همچنین می‌تواند قابلیت‌های کاملاً جدیدی را معرفی‌کند که در کلاس پایه وجود ندارد. اجازه‌دهید کلاس **Emailer** که قبلاً نوشته‌ایم را توسعه‌دهیم و کلاسی جدید بسازیم که قابلیت ارسال ایمیل‌هایی با ساختار **HTML** را نیز داشته‌باشد. کلاس **Emailer** اصلی به‌صورت زیر است:

```
class Emailer {
    protected $sender;
    protected $recipients;
    protected $subject;
    protected $body;

    public function __construct($sender) {
        $this->sender = $sender;
        $this->recipients = array();
    }

    public function AddRecipients($recipient) {
        array_push($this->recipients, $recipient);
    }

    public function SetSubject($subject) {
        $this->subject = $subject;
    }

    public function SetBody($body) {
        $this->body = $body;
    }

    public function SendEmail() {
        foreach ($this->recipients as $recipient) {
            $result = mail($recipient, $this->subject, $this->body, "From: {$this->sender}\r\n");
            if ($result) {
                echo "Mail successfully sent to {$recipient}<br/>" . PHP_EOL;
            }
        }
    }
}
```

اگر به‌خوبی به کد فوق دقت‌کرده و آنرا با کد موجود در جلسات قبل مقایسه‌کنید، متوجه می‌شوید که به‌جای کلمه کلیدی **private** از کلمه کلیدی **protected** استفاده شده‌است. عناصری که با کمک این کلمه کلیدی تعریف می‌شوند، از دید اشیاء، مشابه عناصر **private** هستند (در بیرون از کلاس قابل دسترسی نمی‌باشند)؛ اما اگر کلاسی از کلاس فوق مشتق‌شود، این عناصر را به‌ارث خواهد برد (مشابه عناصر **public**). عناصر **public** نیز توسط کلاس فرزند به‌ارث برده می‌شوند و در کلاس مشتق‌شده، قابل استفاده خواهند بود اما عناصر **private** کاملاً اختصاصی هستند و نه‌تنها در اختیار اشیاء ایجادشده از کلاس قرار نمی‌گیرند، بلکه کلاس‌های مشتق‌شده از کلاس پایه نیز به عناصر **private** آن دسترسی نخواهند داشت. حال به کد کلاس جدید دقت کنید:

```
class HTMLEmailer extends Emailer {
    public function SendHTMLEmail() {
        foreach ($this->recipients as $recipient) {
            $headers = '';
            $headers .= 'MIME Version: 1.0' . "\r\n";
            $headers .= 'Content-Type: text/html; charset=utf-8' . "\r\n";
            $headers .= 'From: ' . $this->sender . "\r\n";
            $headers .= 'Reply-To: ' . $this->sender . "\r\n";
            $headers .= 'X-Mailer: PHP/' . phpversion() . "\r\n";
            $body = '';
            $body .= '<!doctype html>' . "\r\n";
            $body .= '<html>' . "\r\n";
            $body .= '<head>' . "\r\n";
            $body .= '<title>' . $this->subject . '</title>' . "\r\n";
            $body .= '<meta charset="utf-8"/>' . "\r\n";
            $body .= '<style type="text/css">' . "\r\n";
            $body .= '    * {' . "\r\n";
            $body .= '        font-family: Tahoma;' . "\r\n";
            $body .= '    }' . "\r\n";
            $body .= '</style>' . "\r\n";
            $body .= '</head>' . "\r\n";
```

```
            $body .= '<body>' . "\r\n";
            $body .= nl2br($this->body);
            $body .= '</body>' . "\r\n";
            $body .= '</html>' . "\r\n";
            $result = mail($recipient, $this->subject, $body, $headers);
            if ($result) {
                echo "HTML mail successfully sent to {$recipient}<br/>" . PHP_EOL;
            }
        }
    }
}
```

در ابتدای کد و در قسمت تعریف کلاس **HTMLEmailer** با کمک دستور **extends Emailer** اعلام می‌کنیم که این کلاس قصد توسعه و گسترش کلاس **Emailer** را دارد. بدین‌ترتیب، کلاس **Emailer** کلاس پایه (والد) و کلاس **HTMLEmailer** کلاس مشتق‌شده (فرزند) نامیده می‌شود. بدین‌ترتیب، تمامی عناصر غیر **private** کلاس **Emailer** در کلاس **HTMLEmailer** نیز وجود دارد. گویی همان کدها را دقیقاً در این کلاس تایپ کرده‌ایم. در ادامه، متد **SendHTMLEmail** تعریف می‌شود که در کلاس والد وجود نداشته‌است و فقط در کلاس فرزند تعریف شده‌است و بدین‌ترتیب، اشیاء ایجاد شده از کلاس فرزند دارای ویژگی خاصی هستند که اشیاء ایجاد شده از کلاس والد فاقد آن هستند و آن، ارسال ایمیل با ساختار **HTML** است. حال اجازه‌دهید ببینیم از این کلاس چگونه استفاده می‌شود:

```
$emailerObject = new HTMLEmailer('mmshfe@gmail.com');
$emailerObject->SetSubject('Test');
$emailerObject->SetBody('This is a simple HTML email.');
$emailerObject->SendEmail();
$emailerObject->SendHTMLEmail();
```

همان‌طور که ملاحظه می‌کنید، کاملاً مشابه یک کلاس مستقل، از این کلاس نیز می‌توانیم شئ بسازیم و متدهای موجود در آنرا فراخوانی کنیم. با دقت در کد فوق در می‌یابیم که اشیاء این کلاس جدید، هم متد **SendEmail** و هم متد **SendHTMLEmail** را در اختیار دارند ولی اشیاء ایجادشده از کلاس والد (**Emailer**) فقط دارای متد **SendEmail** هستند و فراخوانی متد **SendHTMLEmail** برروی آنها موجب بروز خطا خواهد شد.

اگر برای کلاس مشتق‌شده صراحتاً متد سازنده تعریف نشود، به‌طور خودکار در زمان ایجاد شئ از کلاس مشتق‌شده، سازنده کلاس پایه فراخوانی خواهد شد. البته بدیهی است که اگر در متد سازنده کلاس پایه، دستوری شبیه **$this->body = 'ok';** اجرا شود، فیلد **body** از شئ کلاس فرزند مقداردهی خواهد شد، نه کلاس پایه. درحقیقت فقط کد به‌ارث برده می‌شود و همیشه منظور از **$this** دقیقاً شیئی خواهد بود که کد برروی آن فراخوانی می‌شود (در اینجا شئ ایجاد شده از کلاس فرزند). یک کلاس براساس اصول مبنای برنامه‌نویسی شئ‌گرا، فقط می‌تواند یک کلاس پایه داشته‌باشد و وراثت چندگانه در **PHP** پشتیبانی نمی‌شود و درعوض برای این‌کار باید از رابطها استفاده نماییم (مبحث رابطها بعداً مورد بررسی قرار خواهد گرفت).

بازنویسی متدها (چندریختی یا Polymorphism)

در یک کلاس مشتق‌شده، می‌توانیم هر متدی را که در کلاس پایه بصورت **public** یا **protected** تعریف شده‌باشد، بازنویسی کنیم و تغییرات دلخواه را برروی آن انجام‌دهیم. برای انجام این‌کار، کافی‌است متدی با همان نام در کلاس مشتق‌شده تعریف‌کنید. برای مثال، اگر متدی به‌نام **SendEmail** در کلاس **HTMLEmailer** ایجاد کنیم، متد **SendEmail** موجود در کلاس **Emailer** را بازنویسی خواهدکرد و درنتیجه، با فراخوانی متد **SendEmail** برروی اشیاء ایجادشده از کلاس مشتق‌شده، دستورات موجود در متد جدید اجرا خواهد شد؛ هرچند اگر متد **SendEmail** را برروی اشیاء ایجادشده از کلاس پایه اجرا کنیم، کماکان متد قبلی (موجود در کلاس پایه) فراخوانی می‌شود. به این مسئله، چندریختی می‌گویند یعنی فراخوانی یک متد برروی اشیاء مختلف، اجراهای متفاوتی را در پی خواهد داشت. برای درک بهتر، به کلاس زیر دقت‌کنید:

```
class Point2D {
    protected $x;
    protected $y;

    public function __construct($x = 0, $y = 0) {
        $this->x = 0;
        $this->y = 0;
```

```
        $this->SetX($x);
        $this->SetY($y);
    }

    public function SetX($x) {
        if(is_numeric($x) && $x >= 0 && $x <= 639) {
            $this->x = $x;
        }
    }

    public function SetY($y) {
        if(is_numeric($y) && $y >= 0 && $y <= 479) {
            $this->y = $y;
        }
    }

    public function Set($x, $y) {
        $this->SetX($x);
        $this->SetY($y);
    }

    public function Display() {
        echo 'Point is at [' . $this->x . ',' . $this->y .']<br/>' . PHP_EOL;
    }

    public function Distance($point) {
        $diffX = $this->x - $point->x;
        $diffY = $this->y - $point->y;
        return round(sqrt($diffX * $diffX + $diffY * $diffY), 2);
    }
}
```

همان‌طور که ملاحظه می‌کنید، کلاس فوق برای نگهداری مختصات یک نقطه در صفحه نمایش با تراکم **640x480** طراحی شده‌است و متدهایی برای تغییر مؤلفه‌های **X** و **Y** و نمایش موقعیت و همچنین محاسبه فاصله نقطه تا مبدأ مختصات می‌باشد. حال اجازه‌دهید یک کلاس دیگر از این کلاس مشتق‌کنیم و برخی از متدهای آنرا بازنویسی نماییم:

```
class Point3D extends Point2D {
    private $z;

    public function __construct($x = 0, $y = 0, $z = 0) {
        parent::__construct($x, $y);
        $this->z = 0;
        $this->SetZ($z);
    }

    public function SetZ($z) {
        if(is_numeric($z) && $z >= 0 && $z <= 1023) {
            $this->z = $z;
        }
    }

    public function Set($x, $y, $z) {
        parent::Set($x, $y);
        $this->SetY($z);
    }

    public function Display() {
        echo 'Point is at [' . $this->x . ',' . $this->y . ',' . $this->z .']<br/>' . PHP_EOL;
    }

    public function Distance($point) {
        $diffX = $this->x - $point->x;
        $diffY = $this->y - $point->y;
        $diffZ = $this->z - $point->z;
        return round(sqrt($diffX * $diffX + $diffY * $diffY + $diffZ * $diffZ), 2);
    }
}
```

این کلاس، برای نگهداری مختصات یک نقطه در فضای سه‌بعدی کاربرد دارد و علاوه‌بر مؤلفه‌های **X** و **Y** حاوی مؤلفه **Z** نیز می‌باشد که در بازه **0** تا متغیر خواهد بود. مشاهده می‌کنید که متدهای سازنده، **Set**، **Display** و **Distance** در این کلاس بازنویسی شده‌اند. عبارت **parent::** به ما اجازه فراخوانی متدهای کلاس والد را می‌دهد. مثلاً در متد سازنده، بجای نوشتن دستورات زیر:

```
$this->x = 0;
$this->y = 0;
$this->SetX($x);
$this->SetY($y);
```

فقط دستور **parent::__construct($x, $y);** را می‌نویسیم یعنی متد سازنده کلاس پایه را فراخوانی می‌کنیم و **$x** و **$y** را که بعنوان آرگومان برای سازنده کلاس فرزند ارسال شده‌اند، برای آن می‌فرستیم؛ زیرا سازنده کلاس پایه، دقیقاً همین کد را اجرا می‌کند و درنتیجه، نیاز به نوشتن و تایپ مجدد آن نخواهیم‌داشت. بطور کلی هر زمان در کلاس مشتق‌شده، نیاز به فراخوانی یکی از متدهای کلاس پایه داشته‌باشیم، با کمک **parent::** می‌توانیم به آن دسترسی پیدا کنیم؛ اما باید دقت کنید که این دستورات در درون کلاس مشتق‌شده اجرا می‌شوند نه کلاس پایه. این بدان‌معناست که دستور **$this->x = 0;** فیلد **$x** کلاس مشتق‌شده را مقداردهی می‌کند، نه کلاس پایه را، زیرا محل اجرای دستور، در درون کلاس مشتق‌شده است و طبیعتاً **$this** به شئ ایجاد شده از کلاس فرزند اشاره می‌کند نه کلاس والد. بطور خلاصه، ما فقط کد را به‌ارث می‌بریم و محل اجرای کد، مشخص‌کننده مرجع **$this** خواهد بود.

جلوگیری از بازنویسی

شاید گاهی اوقات بخواهید یک یا چند متد را به‌نحوی بنویسید که امکان بازنویسی آنها در کلاس‌های مشتق‌شده وجود نداشته‌باشد و به‌بیان دیگر، تمامی کلاس‌های مشتق‌شده از کلاس پایه نیز آنرا به‌همان‌شکل مورد استفاده قرار دهند. برای این‌کار، کافی‌است قبل‌از کلمه کلیدی **function** از کلمه کلیدی **final** استفاده‌کنید. مثال:

```
class ParentClass {
    public final function SomeMethod() {
        echo 'This one can\'t be overriden<br/>' . PHP_EOL;
    }
}

class ChildClass extends ParentClass {
    public function SomeMethod() {
        echo 'This one will not execute<br/>' . PHP_EOL;
    }
}

// Results in Fatal error: Cannot override final method ParentClass::SomeMethod()
```

اگر کد فوق را اجرا کنید، با یک پیغام خطا مواجه می‌شوید که به شما اعلام می‌کند نمی‌توانید متد **SomeMethod** موجود در **ParentClass** را بازنویسی کنید زیرا بصورت **final** تعریف شده‌است و این، دقیقاً معنای **final** است: تعریف **final** آخرین تعریفی است که از یک تابع در سرتاسر کد وجود خواهد داشت.

جلوگیری از توسعه یک کلاس

درصورت نیاز، مشابه متدهای **final** می‌توانید یک کلاس **final** داشته‌باشید. کافی‌است کلمه کلیدی **final** را قبل از کلمه کلیدی **class** اضافه‌کنید تا جلوی گسترش کلاس توسط وراثت گرفته‌شود. بدین‌ترتیب، هیچ کلاسی اجازه نخواهد داشت از کلاس شما مشتق‌شود. مثال:

```
final class ParentClass {
   public function Test() {
       echo 'BaseClass::test() called<br/>' . PHP_EOL;
   }

   // Here it doesn't matter if you specify the function as final or not
   public final function MoreTesting() {
       echo 'BaseClass::moreTesting() called<br/>' . PHPE_EOL;
   }
}

class ChildClass extends ParentClass {
}

// Results in Fatal error: Class ChildClass may not inherit from final class (ParentClass)
```

با اجرای کد فوق، خواهید دید که پیغام خطایی مبنی‌بر عدم امکان ارث‌بردن **ChildClass** از کلاس **ParentClass** که به‌صورت **final** تعریف شده‌است صادر خواهد شد و بقیه کد اجرا نمی‌شود. بعلاوه در یک کلاس **final**، اهمیتی ندارد که متدها را به‌صورت **final** تعریف‌کنید یا نه؛ زیرا درهرحال کلاسی وجود نخواهد داشت که از کلاس شما مشتق‌شود تا بخواهد متدهای آنرا بازنویسی‌کند.

بررسی وراثت یا نوع کلاس یک شئ

گاهی‌اوقات در پروژه‌های بزرگ که به‌صورت گروهی انجام می‌شود و مشاهده و بررسی تمامی خطوط کد مقدور یا حداقل مقرون‌به‌صرفه نیست، نیاز است بررسی کنیم که آیا کلاسی که شئ خودمان را از آن ایجاد می‌کنیم، از کلاس خاص دیگری ارث برده‌است یا نه تا درصورت لزوم بتوانیم متدهای آن کلاس پایه را نیز برروی شئ موردنظر فراخوانی کنیم. برای این‌منظور، PHP کلمه کلیدی سودمند **instanceof** را ارائه می‌دهد که در شرط‌ها می‌توانیم از آن استفاده‌کنیم. سمت چپ این کلمه کلیدی نام شئ و سمت راست، نام کلاس موردنظر را می‌نویسیم و خروجی آن، درصورتی که شئ ایجادشده، از نوع کلاس مشخص‌شده باشد یا کلاس مربوطه‌به شئ، فرزند کلاس اعلام‌شده باشد، **true** و درغیر اینصورت **false** خواهد بود. مثال:

```php
$emailer = new Emailer('mmshfe@gmail.com');
$htmlEmailer = new HTMLEmailer('mmshfe@gmail.com');
$p2d = new Point2D();
$p3d = new Point3D();
if($emailer instanceof Emailer) {
    echo '$emailer is instance of Emailer class.<br/>' . PHP_EOL;
}
if($htmlEmailer instanceof HTMLEmailer) {
    echo '$htmlEmailer is instance of HTMLEmailer class.<br/>' . PHP_EOL;
}
if($emailer instanceof HTMLEmailer) {
    echo '$emailer is instance of a class which is derived from HTMLEmailer class.<br/>' . PHP_EOL;
}
if($htmlEmailer instanceof Emailer) {
    echo '$htmlEmailer is instance of a class which is derived from Emailer class.<br/>' . PHP_EOL;
}
if($p2d instanceof Point2D) {
    echo '$p2d is instance of Point2D class.<br/>' . PHP_EOL;
}
if($p3d instanceof Point3D) {
    echo '$p3d is instance of Point3D class.<br/>' . PHP_EOL;
}
if($p2d instanceof Point3D) {
    echo '$p2d is instance of a class which is derived from Point3D class.<br/>' . PHP_EOL;
}
if($p3d instanceof Point2D) {
    echo '$p3d is instance of a class which is derived from Point2D class.<br/>' . PHP_EOL;
}
/* Output:
$htmlEmailer is instance of HTMLEmailer class.
$emailer is instance of a class which is derived from HTMLEmailer class.
$htmlEmailer is instance of a class which is derived from Emailer class.
$p2d is instance of Point2D class.
$p3d is instance of Point3D class.
$p3d is instance of a class which is derived from Point2D class.
*/
```

همان‌طور که مشاهده می‌کنید، فقط در مواردی که شئ مشخص‌شده از نوع کلاس سمت راست کلمه کلیدی **instanceof** بوده‌است یا از نوع کلاسی بوده‌است که فرزند کلاس سمت راست کلمه کلیدی **instanceof** می‌باشد، دستور موجود در بلاک **if** مربوطه اجرا شده‌است.

کلاس چکیده (Abstract)

گاهی اوقات ممکن‌است چند کلاس داشته‌باشید که کارهای مشابهی را انجام می‌دهند. برای مثال، فرض‌کنید کلاس‌هایی برای مدیریت اشکال هندسی مثل دایره، مکعب، مثلث، مربع و... نوشته‌اید و قصد دارید همه این کلاس‌ها حاوی متدهایی برای نمایش محیط و مساحت باشند اما چگونگی محاسبه مقادیر مربوط به هر شئ، به خود آن کلاس واگذار شود. در این حالت، کلاس‌های چکیده به‌کمک می‌آیند. می‌توانید با اضافه‌کردن کلمه کلیدی **abstract** به قبل‌از کلمه کلیدی **class**، یک کلاس چکیده تعریف‌کنید. امکان تعریف شئ از کلاس‌هایی که به‌صورت **abstract** تعریف می‌شوند، وجود ندارد. در عوض، این کلاس‌ها متدهایی را تعریف می‌کنند که در تمامی کلاس‌های مشتق‌شده از آنها وجود دارد و درصورت نیاز، می‌توانید آنها را بازنویسی کنید. امکان تعریف هرکدام از متدهای موجود در این کلاس‌ها نیز بصورت **abstract** وجود دارد. برای این کار، قبل از کلمه کلیدی **function** در قسمت تعریف تابع باید کلمه کلیدی **abstract** ذکر شود. نکته مهم آن‌است که اگر حداقل یک متد **abstract** در یک کلاس وجود داشته‌باشد، باید

آن کلاس بصورت **abstract** تعریف شود و دیگر نمی‌توان برای آن، فیلد و... تعریف‌نمود. ازطرفی متدهایی که صراحتاً بصورت **abstract** تعریف می‌شوند (کلمه **abstract** در قسمت اعلان آنها، علاوه‌بر اعلان کلاس و بصورت مستقیم ذکر می‌شود)، **باید** در کلاس‌های مشتق‌شده بازنویسی شوند و در کلاس پایه (**abstract**) **نباید** برای آنها بدنه تعریف‌شود و بلافاصله بعد از پرانتز بسته قسمت اعلان تابع، کارکتر سمی‌کالن (;) ذکر می‌شود.

زمانی که کلاسی از یک کلاس **abstract** مشتق می‌شود، تمامی متدهایی که در کلاس والد بصورت **abstract** تعریف شده‌اند باید با همان الگو (ازنظر تعداد پارامترهای ورودی و نام و...) بازنویسی شوند. بعلاوه سطح دسترسی نیز باید مشابه یا آزادانه‌تر باشد. برای مثال، اگر متد **abstract** بصورت **protected** تعریف شده‌است، در زمان بازنویسی باید بصورت **protected** یا **public** تعریف شود و نمی‌توان آنرا بصورت **private** تعریف‌نمود. همچنین ازنظر تعداد پارامترها باید همخوانی وجود داشته باشد. برای مثال، اگر کلاس مشتق‌شده یک آرگومان اختیاری تعریف کند و متد **abstract** آنرا بصورت اجباری تعریف کرده باشد، مشکلی وجود ندارد زیرا درصورت عدم مقداردهی آرگومان اختیاری توسط کاربر، مقدار پیشفرض جایگزین آن خواهد شد. ضمناً از **PHP** نسخه **5.4** به‌بعد، اگر متد سازنده (**__construct**) در کلاس **abstract** مشخص شده باشد، باید الگوی آن در کلاس‌های مشتق‌شده نیز به‌همان‌شکل تعیین و رعایت شود؛ درحالی‌که در نسخه‌های قبلی، سازنده‌ها از این قاعده مستثنی بودند.

برای درک بهتر، اجازه‌دهید از یک مثال کمک بگیریم:

```
abstract class GenerateReport {
    public function GenerateReport($resultArray) {
        // Write the code to process the multi-dimensional array
        // And generate the HTML report
    }
    public abstract function CustomReport($resultArray);
}
```

مشاهده می‌کنید که کلاس **GenerateReport** بصورت **abstract** تعریف شده‌است. بنابراین، امکان ایجاد شئ از این کلاس وجود ندارد. در این کلاس، دو متد **GenerateReport** و **CustomReport** تعریف شده‌اند که اولی معمولی و دومی **abstract** است (به‌دلیل وجود یک متد **abstract**، نمی‌توانیم کلاس فوق را بدون کلمه کلیدی **abstract** تعریف‌کنیم). به‌دلیل تعریف متد اول بصورت معمولی، می‌توانیم برای آن بدنه بنویسیم و دستورات دلخواه خودمان را در آن پیاده‌سازی کنیم. همچنین امکان استفاده از کلمه کلیدی **$this** در این متد وجود دارد! شاید بپرسید وقتی شیئی وجود ندارد، **$this** به چه معناست؟ نکته اصلی باز هم در محل فراخوانی تابع **GenerateReport** است: قرار است کلاس‌های دیگری از این کلاس مشتق‌شوند و آن کلاس‌ها قطعاً دارای شئ خواهند بود و با فراخوانی این متد برروی اشیاء آن کلاس‌ها، **$this** به شئ مربوط به آن کلاس‌ها اشاره خواهد کرد. اما متد دوم بصورت **abstract** تعریف شده‌است و درنتیجه نمی‌توانیم برای آن بدنه بنویسیم و این متد باید در کلاس‌های مشتق‌شده، بازنویسی شود (البته با همان ساختار و سطح دسترسی مشخص‌شده در این کلاس). تفاوت اصلی میان متد اول و دوم در آن است که متد اول را **می‌توان** در کلاس‌های مشتق‌شده بازنویسی کرد، اما متد دوم **باید** بازنویسی شود. شاید بپرسید چرا این متدها را درون یک کلاس **abstract** قرار داده‌ایم. علت این مسئله آن‌است که تولید یک گزارش، یک ویژگی رایج در تمامی کلاس‌های مرتبط با پایگاه داده‌ها است و بدین‌ترتیب، یک متد هماهنگ با تمام کلاس‌ها وجود خواهد داشت و از آنجا که این متد، یک آرایه چندبعدی معمولی **PHP** بعنوان پارامتر دریافت می‌کند، نحوه تولید گزارش آن، ارتباطی به کلاس مرتبط با پایگاه داده‌ها ندارد و یک کد ثابت خواهد بود. ازطرفی اگر کلاسی نیاز به گزارش ویژه خود داشته باشد، می‌تواند همین متد را بازنویسی کند. ضمناً متد **CustomReport** نیز درنظر گرفته شده‌است که هر کلاس باید آنرا برحسب نیاز خود و متناسب با شرایط کاری خودش، بازنویسی کند و بدنه دلخواه خود را برای آن بنویسد. نکته جالب اینجاست که به‌سادگی می‌توانیم با کمک کلمه کلیدی **instanceof** بررسی کنیم که آیا یک شئ از نوع کلاس **GenerateReport** یا یکی از فرزندان آن است یا نه و اگر بود، با خیال راحت می‌توانیم متدهای **GenerateReport** و **CustomReport** را برروی آن فراخوانی کنیم. یک نکته مهم که باید به‌خاطر بسپارید، آن‌است که نمی‌توان هم‌زمان از کلمات کلیدی **abstract** و **final** در قسمت اعلان تابع یا کلاس استفاده‌نمود؛ زیرا **final** به‌معنای عدم امکان بازنویسی/وراثت و **abstract** به‌معنای اجبار بازنویسی/وراثت است و کاربرد هم‌زمان این‌دو نه‌تنها معنا ندارد، بلکه **PHP** نیز به شما اجازه آنرا نمی‌دهد و شما را با یک پیغام خطا متوقف می‌کند.

اگر به‌خوبی کاربرد کلاس‌ها و متدهای **abstract** را درک‌کنید، آنها را بسیار سودمند خواهید یافت. تنها نکته منفی درخصوص این کلاس‌ها، بحث عدم امکان وراثت چندگانه است که به‌موجب آن، اگر یک کلاس از یک کلاس پایه (چه **abstract** و چه عادی) مشتق‌شود، دیگر نمی‌تواند کلاس والد دیگری داشته‌باشد و هم‌زمان امکان توسعه چند کلاس پایه برای یک کلاس وجود ندارد. برای رفع این مشکل، رابط‌ها به‌کمک شما می‌آیند.

رابط (Interface)

رابط عملاً مشابه یک کلاس **abstract** است که تمامی متدهای آن صراحتاً بصورت **abstract** تعریف شده‌باشند و طبیعتاً فقط تعریف متدها در آن وجود دارد (قسمت اعلان متد) و حاوی هیچ فیلد یا بدنه متدی نیست. درنتیجه یک رابط به‌تنهایی هیچ کاربردی ندارد. کاربرد اصلی یک رابط زمانی است که یک کلاس قصد پیاده‌سازی متدهای تعریف‌شده در آنرا داشته‌باشد. برای تعریف یک رابط، از کلمه کلیدی **interface** و سپس، نام رابط استفاده می‌شود. مثال:

```
interface iShape {
    public function Area();
    public function Pyramid();
}
```

حال اگر کلاسی بخواهد این رابط را پیاده‌سازی کند، **باید** تمامی متدهای مشخص‌شده در آنرا با همین ساختار، بازنویسی نماید. دقت‌کنید که در تعریف رابط از کلمه **abstract** در تعریف متدها یا خود رابط استفاده نمی‌شود. ضمناً برای اینکه مشخص‌کنیم یک کلاس، قرار است یک رابط را پیاده‌سازی نماید، از کلمه کلیدی **implements** بعد از اسم کلاس (یا کلاس والد – درصورت وجود) و سپس نام رابط استفاده می‌کنیم. به کلاس‌های نمونه زیر دقت‌کنید:

```
class Circle implements iShape {
    private $radius;
    const PI = 3.1415926535;
    public function __construct($radius) {
        $this->radius = abs($radius);
    }
    public function Area() {
        echo $this->PI * $this->radius * $this->radius;
    }
    public function Pyramid() {
        echo 2 * $this->PI * $this->radius;
    }
}
class Square implements iShape {
    private $edge;
    public function __construct($edge) {
        $this->edge = abs($edge);
    }
    public function Area() {
        echo 4 * $this->edge;
    }
    public function Pyramid() {
        echo $this->edge * $this->edge;
    }
}
```

همان‌طور که مشاهده می‌کنید، هر کلاس، پیاده‌سازی خاص خودش را برای متدهای **Area** و **Pyramid** که در رابط **iShape** تعریف شده‌اند، ارائه کرده‌است اما درهرحال، اجباراً باید اقدام به بازنویسی متدها آن‌هم با ساختاری که رابط **iShape** تعیین کرده‌است، بنماید. حال به‌سادگی با کمک **instanceof** می‌توانیم پیاده‌سازی‌شدن یک رابط را توسط یک شئ (کلاس مربوط به شئ) بررسی کنیم و درصورت تأیید، متدهای معرفی‌شده در رابط (و پیاده‌سازی‌شده در کلاس شئ) را برروی شئ فراخوانی نماییم. ازطرفی می‌توان بیش‌از یک رابط را توسط یک کلاس، پیاده‌سازی نمود و کافی‌است اسامی آنها را بعد از کلمه کلیدی **implements** نوشته و با کاما (,) از هم جدا کنیم. البته نمی‌توان دو رابط را که دارای متدهای هم‌نام هستند، به‌طور هم‌زمان پیاده‌سازی نمود؛ زیرا ایجاد ابهام می‌کند و **PHP** این موضوع را با پیغام خطا به شما گوشزد خواهد نمود. همچنین یک رابط می‌تواند یک رابط دیگر را به‌کمک کلمه کلیدی **extends** گسترش‌دهد. بدین‌ترتیب، کلاسی که رابط فرزند را پیاده‌سازی می‌کند، باید همه متدهای تعریف‌شده در رابط پایه و رابط فرزند را پیاده‌سازی و بازنویسی نماید.

اطلاعات بیشتر درباره OOP

در جلسات قبل اطلاعات لازم را برای شروع کار با برنامه‌نویسی شئ‌گرا در **PHP** به شما ارائه کردیم. این جلسه به برخی از ویژگی‌های پیشرفته شئ‌گرایی همراه با جزئیات بیشتر، اختصاص دارد. برای مثال، درمورد توابع استخراج اطلاعات کلاس و نحوه کاربرد آنها برای بررسی جزئیات هر کلاسی خواهیم آموخت. سپس برخی از توابع مفید اطلاعات شئ‌گرایی را توضیح خواهیم داد و همچنین یکی از قابلیت‌های بسیار سودمند و جدید **PHP5** یعنی مدیریت خطا را بررسی خواهیم کرد.

بعلاوه، این جلسه به شما چگونگی تعریف پیمایشگر (**Iterator**) را جهت دسترسی راحت‌تر به آرایه‌ها خواهد آموخت. برای ذخیره هر شئ جهت استفاده در آینده، ما باید از یک ویژگی خاص **OOP** به‌نام سریال‌کردن (**Serialize**) استفاده کنیم که آنرا نیز در این جلسه آموزش خواهیم‌داد. بطور کلی، این جلسه پایه شما را در **OOP** محکم می‌کند.

توابع مربوط به اطلاعات کلاس

در پروژه‌های گروهی، بارها اتفاق می‌افتد که بخشی از پروژه را همکار شما درقالب یک کلاس، بنویسد و در اختیار شما قرار دهد. طبیعتاً بررسی ساختار کد همکارتان با توجه به تفاوت‌های موجود در سبک و روش کدنویسی، کار راحتی نخواهد بود. اما چاره چیست؟ به‌هرحال باید اطلاعات کلی درباره کلاسی که در اختیار گرفته‌اید، داشته باشید. حال اگر می‌خواهید هر کلاسی را بررسی کنید و اطلاعات بیشتری درباره آن کسب نمایید، توابع مربوط به اطلاعات کلاس، چراغ شما در تاریکی هستند. این توابع تقریباً می‌توانند هرگونه اطلاعات موردنظر شما را درباره یک کلاس، استخراج و درقالب یک‌سری اطلاعات استاندارد به شما عرضه کنند. خوشبختانه در **PHP5** یک نسخه توسعه‌یافته از این کلاس‌ها بصورت یک مجموعه **API** (**Application Programming Interface** یا رابط برنامه‌نویسی کاربردی) کاملاً جدید معرفی و تحت عنوان **Reflection** (انعکاس) نام‌گذاری شده‌است. درباره **Reflection** نیز اطلاعات خوبی در جلسه بعد خواهید یافت.

بررسی موجود بودن یک کلاس

یکی از عادت‌های خوب در برنامه‌نویسی شئ‌گرا که مانع از بروز مشکلات جدی در کدنویسی می‌شود، بررسی وجود یک کلاس قبل‌از ایجاد یک شئ از آن است. برای این‌کار **PHP** تابع **class_exists** را معرفی می‌کند. به مثال زیر دقت‌کنید:

```
include_once 'emailer.class.php';
if(class_exists('Emailer')) {
    $emailer = new Emailer('mmshfe@gmail.com');
    $emailer->AddRecipient('admin@ncis.ir');
    $emailer->AddRecipient('info@ncis.ir');
    $emailer->SetSubject('Test Email');
    $emailer->SetBody('This is a simple email.');
    $emailer->SendEmail();
}
else {
    echo 'Mail functionality is temporarily disabled.<br/>' . PHP_EOL;
}
```

در کد فوق، ابتدا با کمک تابع **include_once** فایل **email.class.php** را ضمیمه کرده‌ایم. اما همان‌طور که می‌دانید، ممکن‌است بنابه برخی دلایل (مثل آدرس دهی اشتباه، حذف ناخواسته فایل توسط مدیر سایت و...)، این فایل ضمیمه نشود. طبیعتاً در این شرایط، قابلیت ارسال ایمیل سایت به‌دلیل عدم بارگذاری کلاس **Emailer** با مشکل مواجه می‌شود و دیگر نمی‌توانیم از کلاس **Emailer** شئ بسازیم (چون اصلاً تعریف نشده است). بنابراین، ابتدا با کمک تابع **class_exists** وجود کلاس **Emailer** را بررسی می‌کنیم و درصورت وجود، از آن شئ می‌سازیم و سایر کارها را برمبنای شئ ساخته‌شده انجام می‌دهیم. شاید بگویید می‌توانیم این مشکل را با **require_once** حل کنیم؛ اما باید دقت‌کنید که استفاده از **require_once** تنها باعث می‌شود که درصورت عدم امکان ضمیمه‌کردن فایل موردنظر (در اینجا **email.class.php**)، اجرای اسکریپت خاتمه یابد و چنین راه‌حلی در زمانی که ویژگی معرفی‌شده در فایل ضمیمه‌شده (مثل ارسال ایمیل در مثال فوق)، یک ویژگی جانبی برای سایت است و عدم‌کارکرد آن، به کارکرد بقیه بخش‌های سایت صدمه‌ای وارد نمی‌کند، راه‌حل مناسبی نیست. بجای این‌کار، به‌سادگی از **include_once** استفاده کرده‌ایم و برای جلوگیری از بروز خطا، بعد از دستور فوق، بررسی می‌کنیم که آیا کلاس موردنظر ما بارگذاری شد یا خیر و فقط درصورت بارگذاری (و درنتیجه

تعریف‌شدن کلاس **Emailer**)، کارهای مربوط به ارسال ایمیل را انجام می‌دهیم. حتی می‌توانیم مطابق مثال فوق، درصورت عدم‌امکان ارسال ایمیل، به کاربر پیغام خطای مناسب را نیز نشان دهیم.

یافتن تمام کلاس‌های بارگذاری‌شده

در برخی از موارد، نیاز خواهید داشت که بدانید دقیقاً چه کلاس‌هایی در حال حاضر بارگذاری شده‌اند و می‌توانید از آنها شئ بسازید. در این شرایط، بررسی تک‌تک کلاس‌ها با استفاده از **class_exists** ایده مناسبی نخواهد بود. راه‌حل مناسب در این وضعیت، استفاده از تابع **get_declared_classes** است که تمامی کلاس‌های تعریف‌شده را بصورت یک آرایه به شما اعلام می‌کند. مثال:

```php
require_once 'emailer.class.php';
require_once 'htmlemailer.class.php';
$classes = get_declared_classes();
print_r($classes);
/* Output:
Array
(
    [0] => stdClass
    [1] => Exception
    [2] => ErrorException
    [3] => Closure
    [4] => COMPersistHelper
    [5] => com_exception
    [6] => com_safearray_proxy
    [7] => variant
    [8] => com
    [9] => dotnet
    [10] => DateTime
    [11] => DateTimeZone
    [12] => DateInterval
    [13] => DatePeriod
    [14] => LogicException
    [15] => BadFunctionCallException
    [16] => BadMethodCallException
    [17] => DomainException
    [18] => InvalidArgumentException
    [19] => LengthException
    [20] => OutOfRangeException
    [21] => RuntimeException
    [22] => OutOfBoundsException
    [23] => OverflowException
    [24] => RangeException
    [25] => UnderflowException
    [26] => UnexpectedValueException
    [27] => RecursiveIteratorIterator
    [28] => IteratorIterator
    [29] => FilterIterator
    [30] => RecursiveFilterIterator
    [31] => ParentIterator
    [32] => LimitIterator
    [33] => CachingIterator
    [34] => RecursiveCachingIterator
    [35] => NoRewindIterator
    [36] => AppendIterator
    [37] => InfiniteIterator
    [38] => RegexIterator
    [39] => RecursiveRegexIterator
    [40] => EmptyIterator
    [41] => RecursiveTreeIterator
    [42] => ArrayObject
    [43] => ArrayIterator
    [44] => RecursiveArrayIterator
    [45] => SplFileInfo
    [46] => DirectoryIterator
    [47] => FilesystemIterator
    [48] => RecursiveDirectoryIterator
    [49] => GlobIterator
    [50] => SplFileObject
    [51] => SplTempFileObject
    [52] => SplDoublyLinkedList
    [53] => SplQueue
    [54] => SplStack
    [55] => SplHeap
    [56] => SplMinHeap
    [57] => SplMaxHeap
    [58] => SplPriorityQueue
    [59] => SplFixedArray
    [60] => SplObjectStorage
    [61] => MultipleIterator
    [62] => ReflectionException
    [63] => Reflection
    [64] => ReflectionFunctionAbstract
    [65] => ReflectionFunction
    [66] => ReflectionParameter
    [67] => ReflectionMethod
    [68] => ReflectionClass
    [69] => ReflectionObject
    [70] => ReflectionProperty
    [71] => ReflectionExtension
    [72] => __PHP_Incomplete_Class
    [73] => php_user_filter
    [74] => Directory
    [75] => ZipArchive
    [76] => LibXMLError
    [77] => DOMException
    [78] => DOMStringList
    [79] => DOMNameList
    [80] => DOMImplementationList
    [81] => DOMImplementationSource
    [82] => DOMImplementation
    [83] => DOMNode
    [84] => DOMNameSpaceNode
    [85] => DOMDocumentFragment
    [86] => DOMDocument
    [87] => DOMNodeList
    [88] => DOMNamedNodeMap
    [89] => DOMCharacterData
    [90] => DOMAttr
    [91] => DOMElement
    [92] => DOMText
    [93] => DOMComment
    [94] => DOMTypeinfo
    [95] => DOMUserDataHandler
    [96] => DOMDomError
    [97] => DOMErrorHandler
    [98] => DOMLocator
    [99] => DOMConfiguration
    [100] => DOMCdataSection
    [101] => DOMDocumentType
    [102] => DOMNotation
    [103] => DOMEntity
    [104] => DOMEntityReference
    [105] => DOMProcessingInstruction
    [106] => DOMStringExtend
    [107] => DOMXPath
    [108] => PDOException
    [109] => PDO
    [110] => PDOStatement
    [111] => PDORow
    [112] => SimpleXMLElement
    [113] => SimpleXMLIterator
    [114] => XMLReader
    [115] => XMLWriter
    [116] => mysqli_sql_exception
    [117] => mysqli_driver
    [118] => mysqli
    [119] => mysqli_warning
    [120] => mysqli_result
    [121] => mysqli_stmt
    [122] => PharException
    [123] => Phar
    [124] => PharData
    [125] => PharFileInfo
    [126] => SoapClient
    [127] => SoapVar
    [128] => SoapServer
    [129] => SoapFault
    [130] => SoapParam
    [131] => SoapHeader
    [132] => Emailer
    [133] => HTMLEmailer
)
*/
```

همان‌طور که مشاهده می‌کنید، فهرست تمامی کلاس‌های موجود جهت استفاده (شامل کلاس‌های تعریف‌شده توسط **PHP**) ارائه می‌شود. در مثال فوق، فقط کلاس‌های ۱۳۲ و ۱۳۳ توسط برنامه‌نویس تعریف شده‌اند.

بررسی وجود یک متد در یک کلاس

برای بررسی وجود یک متد در یک کلاس و درنتیجه قابلیت فراخوانی آن برروی شئ، دو تابع وجود دارد. تابع اول، **method_exists** است که دو پارامتر دریافت می‌کند که اولی، متغیر شئ یا نام کلاس مربوط به شئ است و دومی، نام متد موردنظر و درصورت وجود متد ذکرشده (اعم از **private** یا **public** یا **protected**)، نتیجه **true** و درصورت تعریف‌نشدن متد موردنظر در شئ (کلاس مربوط به شئ)، نتیجه **false** خواهد بود. ضمناً این تابع به بزرگی و کوچکی حروف نام تابع و کلاس حساس نیست (البته اگر پارامتر اول شئ باشد، باید بزرگی و کوچکی حروف را درمورد آن رعایت کنید). مثال:

```
class Test {
    private function Test1() {
        echo 'Test1';
    }

    public function Test2() {
        echo 'Test2';
    }
}
$t = new Test();
echo (method_exists($t, 'test1') ? 'true' : 'false');     // Output: true
echo (method_exists($t, 'Test2') ? 'true' : 'false');     // Output: true
echo (method_exists('Test', 'Test1') ? 'true' : 'false'); // Output: true
echo (method_exists('Test', 'test1') ? 'true' : 'false'); // Output: true
```

تابع بعدی که دقت بیشتری دارد، **get_class_methods** نام دارد. این تابع نیز درخصوص نام کلاس به بزرگی و کوچکی حروف حساس نیست. مثالی از نحوه کاربرد این تابع را مشاهده می‌کنید:

```
class Test {
    private function Test1() {
        echo 'Test1';
    }

    public function Test2() {
        echo 'Test2';
    }
}
print_r(get_class_methods('Test'));
/* Output:
Array (
    [0] => Test2
)
*/
echo (in_array('Test1', get_class_methods('Test')) ? 'true' : 'false'); // Output: false
echo (in_array('Test2', get_class_methods('test')) ? 'true' : 'false'); // Output: true
```

همان‌طور که ملاحظه می‌کنید، تابع **get_class_methods** فقط متدهای **public** (با قابلیت فراخوانی برروی اشیاء کلاس) را استخراج می‌کند و بصورت یک آرایه باز می‌گرداند که با کمک تابع **in_array** می‌توان وجود یک تابع موردنظر را در این آرایه، بررسی‌نمود.

بررسی وجود یک فیلد (خاصیت) در یک کلاس

مشابه روش بررسی وجود یک متد، می‌توان وجود یک فیلد را نیز بررسی نمود. برای این‌کار نیز دو تابع **property_exists** و **get_class_vars** وجود دارند. ابتدا به مثالی از نحوه کاربرد آنها دقت کنید:

```
class Test {
    public $field1 = 25;
    private $fields2 = 5;
}
$t = new Test();
echo (property_exists($t, 'field1') ? 'true' : 'false');     // Output: true
echo (property_exists($t, 'field2') ? 'true' : 'false');     // Output: false
echo (property_exists('Test', 'field1') ? 'true' : 'false'); // Output: true
echo (property_exists('Test', 'field1') ? 'true' : 'false'); // Output: true
print_r(get_class_vars('Test'));
/* Output:
Array (
    ['field1'] => 25
)
*/
echo (in_array('field1', array_keys(get_class_vars('Test'))) ? 'true' : 'false'); // Output: true
echo (in_array('field2', array_keys(get_class_vars('test'))) ? 'true' : 'false'); // Output: false
```

درمورد توابع فوق، توضیحات زیر را به‌خاطر بسپارید:

۱- اگر پارامتر اول تابع **property_exists** یک شئ باشد (نَه نام کلاس)، فقط فیلدهای **public** (که از بیرون قابل مشاهده هستند) استخراج می‌شود و درنتیجه اگر فیلد مشخص‌شده در پارامتر دوم، **public** نباشد، نتیجه **false** خواهد بود.

۲- تابع **property_exists** درخصوص نام فیلد، نسبت به بزرگی و کوچکی حروف حساس است.

۳- تابع **get_class_vars** فقط فیلدهای عمومی (**public**) کلاس را استخراج می‌کند.

٤- خروجی تابع **get_class_vars** یک آرایه انجمنی (با اندیس رشته‌ای) است که نام فیلد، اندیس و مقدار آن، مقدار خانه مربوطه را در آرایه خروجی تشکیل می‌دهد. درنتیجه همان‌طور که در مثال فوق مشخص است، برای بررسی وجود یک فیلد در فهرست فیلدهای یک کلاس، باید ابتدا از تابع **array_keys** برای استخراج اندیس‌های آرایه خروجی تابع فوق‌الذکر استفاده‌کنیم و سپس، نام فیلد را در آرایه خروجی تابع **array_keys** جستجو نماییم.

بررسی نوع کلاس یک شئ

تابع خاصی به‌نام **is_a** وجود دارد که نوع کلاس یک شئ را بررسی می‌کنید. این تابع دو پارامتر دریافت می‌کند که اولی، شئ موردنظر جهت بررسی و دومی، نام کلاسی است که می‌خواهیم ببینیم شئ مربوطه، از نوع آن کلاس است یا نه؟ به مثال زیر دقت‌کنید:

```
class ParentClass {
}
class ChildClass extends ParentClass {
}

$obj = new ChildClass();
if(is_a($obj, 'ChildClass')) {
    echo '$obj is a ChildClass type object.<br/>' . PHP_EOL;
}
if(is_a($obj, 'ParentClass')) {
    echo '$obj is also a ParentClass type object.<br/>' . PHP_EOL;
}
/* Output:
$obj is a ChildClass type object.
$obj is also a ParentClass type object.
*/

if($obj instanceof ChildClass) {
    echo '$obj is a ChildClass type object.<br/>' . PHP_EOL;
}
if($obj instanceof ParentClass) {
    echo '$obj is also a ParentClass type object.<br/>' . PHP_EOL;
}
/* Output:
$obj is a ChildClass type object.
$obj is also a ParentClass type object.
*/
```

همان‌طور که مشاهده می‌کنید، می‌توانید از تابع **instanceof** که قبلاً آموختید نیز برای این منظور استفاده‌کنید. شاید بپرسید چرا برای یک هدف، دو روش وجود دارد. برای رفع ابهام فوق، به توضیحات زیر دقت‌کنید:

۱- **instanceof** یک تابع نیست بلکه یک کلمه کلیدی است و سرعت اجرای بیشتری دارد.

۲- **is_a** به‌لطف وجود **instanceof** در نسخه **5.0.0** زبان **PHP** منسوخ اعلام شده بود اما در نسخه **5.3.0** مجدداً از حالت منسوخ خارج شد و این نشان از قصد تیم توسعه **PHP** جهت بازنویسی و افزودن امکانات این تابع داشت و در نسخه **5.3.9** ویژگی جدید آن معرفی شد: این تابع اکنون یک پارامتر سوم هم دارد که اختیاری است و یک مقدار منطقی (**true** یا **false**) بعنوان پارامتر سوم دریافت می‌کند که مشخص می‌کند اگر کلاس وجود داشته باشد ولی بارگذاری نشده باشد، **PHP** تابع تعریف‌شده برای بارگذاری خودکار را فراخوانی کند یا خیر. مقدار پیش‌فرض آن، **false** (بارگذاری خودکار غیرفعال) است. البته ویژگی فوق (پارامتر سوم) فقط درصورتی کاربرد دارد که در پارامتر اول، نام کلاس (بصورت یک رشته) اعلام شده‌باشد، نه شئ ایجاد شده است کلاس. درخصوص بارگذاری خودکار کلاس‌ها در ادامه مطالب سودمندی را خواهید آموخت.

یافتن نام کلاس

در مثال قبل، نوع کلاس یک شئ را با یک مقدار خاص مقایسه کردیم تا ببینیم آیا شئ موردنظر ما از نوع آن کلاس است یا نه؟ حال اگر نخواهیم مقایسه‌ای انجام دهیم و صرفاً بخواهیم بدانیم که یک شئ که در کد مشاهده می‌کنیم، از نوع چه کلاسی است (در کدهای پیچیده و گروهی چنین وضعیتی به‌دفعات پیش خواهد آمد)، چه کاری باید انجام دهیم؟ نگران نباشید، تابع **get_class** در اینجا به‌کمک ما می‌آید. مثال:

```
class ParentClass {
}
class ChildClass extends ParentClass {
}
$obj = new ChildClass();
echo get_class($obj); // Output: ChildClass
```

ساده است؟ تابع **get_class** یک پارامتر اختیاری دریافت می‌کند که همان شئ موردنظر جهت بررسی است. اما صبر کنید، گفتیم اختیاری؟! مگر قرار نیست نام کلاس یک شئ را بدست بیاوریم؟ پس اگر نام شئ اختیاری است، درصورت حذف آن، چه‌چیزی بررسی خواهد شد؟ اجازه‌دهید این مثال را بررسی کنیم:

```
class ParentClass {
    public function GetClass() {
        echo get_class();
    }
    public function GetRealClass() {
        echo get_class($this);
    }
}
class ChildClass extends ParentClass {
}
$obj = new ChildClass;
$obj->GetClass();     // Output: ParentClass
$obj->GetRealClass(); // Output: ChildClass
```

در مثال فوق، یک کلاس والد (**ParentClass**) و یک کلاس فرزند (**ChildClass**) وجود دارد که کلاس والد، دو متد **GetClass** و **GetRealClass** را تعریف کرده‌است و کلاس فرزند، نه متدی مستقیماً تعریف کرده‌است و نه این متدها را بازنویسی می‌کند. بنابراین، اشیاء ایجادشده از کلاس فرزند، فقط همین دو متد را دارا خواهند بود. حال به متد **GetClass** دقت کنید که چگونه تابع **get_class** زبان **PHP** را بدون پارامتر به‌کار برده است. فقط در بدنه یک کلاس می‌توانیم این تابع را بدون پارامتر فراخوانی کنیم و در این‌صورت، منظور همان شئ ایجادشده از کلاس خواهد بود. منتها همان‌طور که ملاحظه می‌کنید، با فراخوانی متد **GetClass** برروی شئ **$obj** عبارت **ParentClass** چاپ می‌شود. آیا این یک خطا است؟ خیر، بار دیگر به کلاس‌های والد و فرزند دقت‌کنید. از آنجا که **ChildClass** متد فوق را بازنویسی نکرده‌است، محل فراخوانی (**Scope**) متد فوق، **ParentClass** خواهد بود و درنتیجه تابع **get_class** نام کلاس والد را استخراج می‌کند. اکنون به متد **GetRealClass** دقت‌کنید. این متد نیز در کلاس **ChildClass** بازنویسی نشده است، اما **$this** (شئ جاری) را بعنوان پارامتر به تابع **get_class** ارسال می‌کند و درنتیجه، محل فراخوانی (**Scope**) متد مذکور به **ChildClass** تغییر می‌یابد (زیرا **$this** یک شئ ایجادشده از کلاس فرزند است، نَه کلاس والد). بنابراین، تابع **get_class** نیز نام کلاس فرزند را بعنوان نتیجه استخراج می‌کند.

مدیریت خطاها

یکی‌از پیشرفته‌ترین امکانات **PHP5** مدیریت خطاها در قالب استثناهایی است که رخ می‌دهند (مشابه سایر زبان‌های **OOP**). **PHP5** استثناها را برای سادگی مدیریت خطا توسط شما، معرفی کرده‌است. برای مشاهده نمونه‌ای از نحوه مدیریت خطا در **PHP** به مثال زیر دقت‌کنید که سعی در اتصال به **MySQL** دارد:

```
class DB {
    public function Connect() {
        mysql_connect('somehost', 'someuser', 'somepass');
    }
}
$db = new DB();
$db->Connect();
```

با اجرای کد فوق، پیغام خطایی مشابه متن زیر دریافت خواهید نمود:

```
Warning: mysql_connect() [<a href="function.mysql-connect">function.mysql-connect</a>]:
php_network_getaddresses: getaddrinfo failed: The requested name is valid, but no data of the
requested type was found. in C:\wamp\www\phpbook\exception.php on line <i>4</i>
```

چه راه‌حلی برای مدیریت این‌گونه خطاها به‌ذهن شما می‌رسد؟ اگر دید شما به مسئله مشابه روندی که **PHP4** در اختیار شما قرار می‌دهد باشد، راه‌حل شما مشابه کد زیر خواهد بود:

```php
error_reporting(0);
class DB {
    public function Connect() {
        return mysql_connect('somehost', 'someuser', 'somepass');
    }
}
$db = new DB();
if(!$db->Connect()) {
    die('Faild to connect to MySQL server');
}
```

حال ببینیم مدیریت خطا در **PHP5** چگونه است؟ به مثال زیر دقت کنید:

```php
error_reporting(0);
class DB {
    public function Connect() {
        if(!mysql_connect('somehost', 'someuser', 'somepass')) {
            throw new Exception('Failed to connect to MySQL server');
        }
    }
}
$db = new DB();
try {
    $db->Connect();
}
catch(Exception $e) {
    echo '<pre>' . htmlentities(print_r($e, true)) . '</pre>';
}
/* Output:
Exception Object
(
    [message:protected] => Failed to connect to MySQL server
    [string:Exception:private] =>
    [code:protected] => 0
    [file:protected] => C:\wamp\www\phpbook\exception.php
    [line:protected] => 6
    [trace:Exception:private] => Array
        (
            [0] => Array
                (
                    [file] => C:\wamp\www\phpbook\exception.php
                    [line] => 12
                    [function] => Connect
                    [class] => DB
                    [type] => ->
                    [args] => Array
                        (
                        )
                )
        )
    [previous:Exception:private] =>
    [message] => 'Exception: Failed to connect to MySQL server in C:\wamp\www\phpbook\exception.php on line 6'
)
*/
```

همان‌طور که ملاحظه می‌کنید، اطلاعات بسیار زیادی در شئ **$e** از نوع کلاس **Exception** نهفته است. می‌توانید تمامی خطاها را توسط بلاک **try...catch** مدیریت کنید. در این ساختار، کدی که احتمال بروز خطا در آن وجود دارد، در یک بلاک **try** قرار می‌گیرد و سپس، بلاک **catch** به‌دنبال آن نوشته می‌شود که وظیفه مدیریت خطای به‌وجود آمده در بلاک **try** را دارد. هر زمان که یک خطا در بلاک **try** رخ‌دهد، بقیه دستورات بلاک **try** اجرا نمی‌شوند و کنترل برنامه به‌طور خودکار به بلاک **catch** منتقل می‌شود تا خطای تولیدشده را که بصورت پارامتر برای بلاک **try** ارسال شده‌است، مدیریت کند. همچنین می‌توان از یک بلاک **try...catch** درون بلاک **try...catch** دیگر استفاده کرد. به مثال زیر دقت‌کنید که چگونه با دو شئ از کلاس **Exception** ساختار مناسب‌تری از مدیریت خطا تولید کرده‌ایم. ابتدا دو کلاس خطا تولید می‌کنیم که کلاس **Exception** را گسترش می‌دهند تا بتوانیم استثنا (**Exception**) سفارشی خودمان را با متن و کد خطای دلخواه بسازیم.

```php
error_reporting(0);
class MySQLConnectionException extends Exception {
    public function __construct() {
        $message = 'Sorry, could not connect to MySQL server.';
        parent::__construct($message, 1111);
    }
}
```

```php
class MySQLDatabaseException extends Exception {
    public function __construct($connection) {
        $message = 'Sorry, could not select the defined database.';
        parent::__construct(mysql_error($connection), 2222);
    }
}
```

اگر به کد کلاس‌های فوق توجه‌کنید، می‌بینید که بعد از تولید پیام خطای مناسب، سازنده کلاس والد (یعنی **Exception**) را برای نمایش پیغام خطای تولیدشده همراه‌با کد موردنظر برای خطای مربوطه، فراخوانی می‌کنند. حال به ادامه کد دقت کنید که چگونه از کلاس‌های فوق برای مدیریت خطا استفاده می‌کند:

```php
class DB {
    public function Connect($host, $user, $pass) {
        $con = mysql_connect($host, $user, $pass);
        if(!$con) {
            throw new MySQLConnectionException();
        }
        return $con;
    }
    public function SelectDB($db, $con) {
        if(!mysql_select_db($db, $con)) {
            throw new MySQLDatabaseException();
        }
    }
}
$db = new DB();
try {
    $con = $db->Connect('localhost', 'root', '');
}
catch(Exception $e) {
    echo '<pre>' . htmlentities(print_r($e, true)) . '</pre>';
}
/* Output:
MySQLConnectionException Object
(
    [message:protected] => Sorry, could not connect to MySQL server.
    [string:Exception:private] =>
    [code:protected] => 1111
    [file:protected] => C:\wamp\www\phpbook\exception.php
    [line:protected] => 19
    [trace:Exception:private] => Array
        (
            [0] => Array
                (
                    [file] => C:\wamp\www\phpbook\exception.php
                    [line] => 31
                    [function] => Connect
                    [class] => DB
                    [type] => ->
                    [args] => Array
                        (
                            [0] => localhost
                            [1] => root
                            [2] =>
                        )

                )

        )

    [previous:Exception:private] =>
    [xdebug_message] =>
MySQLConnectionException: Sorry, could not connect to MySQL server. in
C:\wamp\www\phpbook\exception.php on line 19

Call Stack:
    0.0010     350272   1. {main}() C:\wamp\www\phpbook\exception.php:0
    0.0010     350696   2. DB->Connect() C:\wamp\www\phpbook\exception.php:31

)
*/
```

در کد فوق، به‌دلیل عدم امکان اتصال با نام سرور، نام کاربری و رمز عبور ارائه‌شده، پیغام خطا از نوع **MySQLConnectionExcption** تولید می‌شود و متن موردنظر برای پیغام خطا ظاهر می‌گردد. حال به یک مثال دیگر از فراخوانی دقت‌کنید که در آن، اتصال به‌درستی

برقرار می‌شود ولی نام بانک اطلاعاتی اشتباه نوشته شده‌است:

```
try {
    $con = $db->Connect('localhost', 'root', 'ncis');
    $db->SelectDB('old_ncis', $con);
    echo 'Everything is OK';
}
catch(Exception $e) {
    echo '<pre>' . htmlentities(print_r($e, true)) . '</pre>';
}
/* Output:
MySQLDatabaseException Object
(
    [message:protected] =>
    [string:Exception:private] =>
    [code:protected] => 2222
    [file:protected] => C:\wamp\www\phpbook\exception.php
    [line:protected] => 25
    [trace:Exception:private] => Array
        (
            [0] => Array
                (
                    [file] => C:\wamp\www\phpbook\exception.php
                    [line] => 32
                    [function] => SelectDB
                    [class] => DB
                    [type] => ->
                    [args] => Array
                        (
                            [0] => old_ncis
                            [1] => Resource id #2
                        )
                )
        )
    [previous:Exception:private] =>
    [xdebug_message] =>
MySQLDatabaseException:  in C:\wamp\www\phpbook\exception.php on line 25
Call Stack:
    0.0010     350272   1. {main}() C:\wamp\www\phpbook\exception.php:0
    0.0099     356528   2. DB->SelectDB() C:\wamp\www\phpbook\exception.php:32
)
*/
```

در اینجا خطای تولیدشده از نوع **MySQLDatabaseException** است و این یعنی اتصال با موفقیت انجام‌شده ولی نام پایگاه داده‌ها صحیح نبوده است. حال به مثال سوم که همه‌چیز در آن به‌درستی انجام می‌شود دقت‌کنید:

```
try {
    $con = $db->Connect('localhost', 'root', 'ncis');
    $db->SelectDB('ncis', $con);
    echo 'Everything is OK';
}
catch(Exception $e) {
    echo '<pre>' . htmlentities(print_r($e, true)) . '</pre>';
}
// Output: Everything is OK
```

دریافت تمامی خطاهای PHP بصورت استثنا (Exception)

اگر بخواهید تمامی خطاهای **PHP** را (بجز خطاهای مرگبار یا **FATAL errors** – مثل عدم وجود فایل در **require_once** و...) بصورت یکپارچه و تحت عنوان **Exception** مدیریت کنید، می‌توانید از تابع **set_error_handler** برای این‌منظور استفاده نمایید. این تابع، نام یک تابع را که شما با ساختاری که در ادامه ذکر می‌کنیم، نوشته‌اید دریافت کرده و آنرا بعنوان سیستم مدیریت خطای **PHP** تا پایان اجرای اسکریپت شما، معرفی می‌کند. یکی‌از کاربردهای این مسئله، زمانی است که می‌خواهید تمامی خطاهای رخ‌داده‌شده در سایتی که طراحی کرده‌اید، برای شما توسط پست الکترونیکی (ایمیل) ارسال شود. حال اجازه‌دهید ساختار و نحوه این‌کار را توضیح دهیم. ابتدا باید یک تابع با پارامترهای زیر تعریف کنید:

```
/**
 * My Error Handler
 * @param int $eNumber The error number
 * @param string $eMessage The error message
 * @param string $eFile The file name that the error was raised in
 * @param int $eLine The line number that the error was raised at
 * @param array $eContext The array of all variables those existed in the scope the error was triggered in
 * @return boolean Stop normal error handling (true: stop, false: normal error handling works after function)
*/
function MyErrorHandler($eNumber, $eMessage, $eFile = NULL, $eLine = NULL, $eContext = NULL) {
    // Your error handling code goes here
    // return true/false;
```

```
}
```

اسم این تابع و همچنین نام پارامترهای آنرا می‌توانید به‌دلخواه خود تغییر دهید. پارامترهای این تابع به‌طور کلی توسط **PHP** در زمان تولید خطا مقداردهی می‌شوند و به‌ترتیب به‌شرح زیر هستند:

۱- کد خطای تولیدشده

۲- متن پیغام خطای تولیدشده

۳- (اختیاری) نام فایلی که خطا در آن رخ داده است

٤- (اختیاری) شماره خطی از فایل که خطا در آن رخ داده است

٥- (اختیاری) آرایه‌ای شامل تمامی متغیرهای تعریف‌شده در محدوده و زمان رخ دادن خطا (تابع شما نباید این آرایه و متغیرهای آنرا تغییر دهد)

خروجی این تابع، یک مقدار منطقی (**true** یا **false**) است که اگر **true** باشد، فقط تابع ما کار مدیریت خطا را انجام می‌دهد و درصورت **false** بودن، بعد از پایان کار تابع ما، سیستم مدیریت خطای پیشفرض **PHP** نیز کار عادی خود را انجام خواهد داد. برای مثال، به کد زیر دقت‌کنید که چگونه خطاهای تولیدشده را به ایمیل **support@mysite.com** ارسال می‌کند:

```
function MyErrorHandler($eNumber, $eMessage, $eFile = NULL, $eLine = NULL, $eContext) {
    $to = 'support@mysite.com';
    $subject = 'An error occured in my site';
    $context = nl2br(print_r($eContext, true));
    date_default_timezone_set('Asia/Tehran');
    $dt = date('Y/m/d - H:i:s');
    $message  = <<<EOT
<!doctype html>
<html>
<head>
<title>{$subject}</title>
<meta charset="utf-8"/>
</head>
<body>
<b>Date - Time:</b> {$dt}<br/>
<b>Error Number:</b> {$eNumber}<br/>
<b>Error Message:</b> {$eMessage}<br/>
<b>File:</b> {$eFile}<br/>
<b>Line:</b> {$eLine}<br/>
<b>Context:</b> {$context}<br/>
</body>
</html>

EOT;
    $headers  = 'MIME-Version: 1.0' . "\r\n";
    $headers .= 'Content-type: text/html; charset=utf-8' . "\r\n";
    $headers .= 'From: admin@mysite.com' . "\r\n";
    $headers .= 'Reply-To: admin@mysite.com' . "\r\n";
    $headers .= 'X-Mailer: PHP/' . phpversion()  . "\r\n";
    @mail($to, $subject, $message, $headers);
    return true;
}
```

حال ببینیم چگونه باید این تابع را بعنوان سیستم مدیریت خطا در **PHP** معرفی کنیم؟ همان‌طور که اشاره کردیم، برای این‌کار باید از تابع **set_error_handler** استفاده نماییم. این تابع دو پارامتر دریافت می‌کند که اولی، نام تابع موردنظر برای مدیریت خطاها و اجباری است و دومی، یک پارامتر اختیاری است که مشخص می‌کند چه خطاهایی باید توسط این تابع مدیریت شوند و از همان ساختار تابع **error_reporting** استفاده می‌کند. مثال:

```
set_error_handler('MyErrorHandler', E_ALL);             // All errors
set_error_handler('MyErrorHandler', E_ALL - E_NOTICE); // All errors except notices
```

مقدار پیش‌فرض درصورت ذکرنکردن پارامتر دوم، **E_ALL | E_STRICT** است (همه خطاها شامل پیشنهادات **PHP** برای اصلاح کد شما جهت بیشترین سازگاری با تغییرات آینده و بهترین کارآیی فعلی).

پیمایشگرها

پیمایشگر (**Iterator**)، دستور جدیدی است که در **PHP5** برای کمک به پیمایش اشیاء بصورت یک آرایه اضافه شده‌است. اجازه‌دهید

این‌بار نیز با ذکر مثال، تفهیم مطالب را ساده‌تر نماییم. در **PHP4** شما می‌توانید هر آرایه‌ای را با کمک حلقه **foreach** پیمایش‌کنید:

```
foreach($anyArray as $key => $value) {
    // Do something
}
```

همچنین می‌توانید از **foreach** برای پیمایش یک شئ نیز استفاده‌کنید. به مثال زیر دقت‌کنید:

```
class Test {
    public  $field1 = 25;
    public  $field2 = 5;
    private $field3 = 7;
}
$t = new Test();
foreach($t as $key => $value) {
    echo $key . ' => ' . $value . '<br/>' . PHP_EOL;
}
/* Output:
field1 => 25
field2 => 5
*/
```

دقت‌کنید که با این روش، فقط می‌توانید فیلدهای عمومی را پیمایش‌کنید. حال فرض کنید یکی از فیلدهای کلاس ما، نمرات دانشجویان باشد و ما بخواهیم فقط نمراتی را که بیشتر از ۱۰ هستند، پیدا کرده و پیمایش‌کنیم. چنین کاری در **PHP5** به‌راحتی با پیاده‌سازی رابط **Iterator** قابل انجام است. به مثال زیر دقت کنید:

```
class Grades implements Iterator {
    private $key;
    private $keyN;
    private $keys;
    private $result;
    public function __construct() {
        $this->result = array();
        $this->keyN = -1;
        $this->keys = array();
    }
    public function Add($student, $grade) {
        if($grade >= 0 && $grade <= 20) {
            $this->result[$student] = $grade;
            $this->keys[] = $student;
        }
    }
    public function Rewind() {
        $this->keyN = 0;
        $this->key = $this->keys[$this->keyN];
    }
    public function Current() {
        return $this->result[$this->key];
    }
    public function Key() {
        return $this->key;
    }
    public function Next() {
        $this->keyN++;
        if($this->keyN < count($this->keys)) {
            $this->key = $this->keys[$this->keyN];
        }
    }
    public function Valid() {
        return isset($this->keys[$this->keyN]);
    }
}
$grd = new Grades();
$grd->Add('Ali', 5);
$grd->Add('Mohsen', 15);
$grd->Add('Hamid', 9);
$grd->Add('Reza', 11);
$grd->Add('Ahmad', 17);
foreach($grd as $student => $grade) {
    if($grade >= 10) {
        echo $student . ' : ' . $grade . '<br/>' . PHP_EOL;
    }
}
```

این رابط، ۵ متد را معرفی کرده‌است که باید پیاده‌سازی شوند. هیچ‌کدام از این متدها پارامتر ورودی ندارند ولی نوع خروجی آنها متفاوت است:

۱- **Current** : این متد، مقدار جاری را در هربار اجرای **foreach** باز می‌گرداند. همان‌طور که مشاهده می‌کنید، با کمک فیلد **$key** موقعیت جاری را در آرایه نمرات یافته و همان عنصر را از آرایه موجود در فیلد **$result** باز گردانده‌ایم. نوع خروجی این متد، هر چیزی می‌تواند باشد. درواقع، هر چیزی که می‌خواهیم در هر بار تکرار **foreach**، در بدنه حلقه داشته باشم. برای مثال، در اینجا نمرات دانشجویان و اسامی آنها را لازم داریم، پس نوع خروجی، آرایه است.

۲- **Key** : اندیس جاری را در هربار اجرای **foreach** مشخص می‌کند. این اندیس برای پیمایش آرایه لازم است و درصورتی که بعد از کلمه کلیدی **as** در حلقه **foreach** از ساختار **$key => $value** استفاده شود، در متغیر سمت چپ قرار خواهد گرفت. خروجی این متد باید یک نوع شمارشی (مثل **int** یا **enum**) و یا رشته‌ای باشد و نمی‌توان از انواع غیرشمارشی مثل **float** و... استفاده‌نمود.

۳- **Next** : این متد چیزی باز نمی‌گرداند ولی باید کلید را به‌نحوی افزایش دهد که با فراخوانی بعدی **current**، اندیس بعدی آرایه بازگردانده‌شود. به‌عبارت دیگر، وظیفه این متد افزایش کلید یا اندیس آرایه است.

٤- **Rewind** : این متد نیز وظیفه دارد اندیس آرایه را به اولین عنصر منتقل کند و چیزی باز نمی‌گرداند.

٥- **Valid** : این متد یک مقدار منطقی (**true** یا **false**) باز می‌گرداند و وظیفه آن، کنترل رسیدن به انتهای آرایه است. درواقع حلقه **foreach** تا زمانی که این عنصر مقدار **true** باز می‌گرداند، به تکرار ادامه می‌دهد.

برای درک بهتر، اجازه‌دهید ببینیم حلقه **foreach** چگونه کار می‌کند؟ وقتی که از حلقه **foreach** برای پیمایش یک کلاس که رابط **Iterator** را پیاده‌سازی کرده‌است استفاده می‌کنیم، مراحل زیر به‌ترتیب اتفاق می‌افتد:

۱- ابتدا متد **Rewind** فراخوانی می‌شود تا اندیس آرایه را به اولین عنصر منتقل کند.

۲- متد **Valid** فراخوانی می‌شود تا بررسی شود که اصلاً مقداری در آرایه برای پیمایش وجود دارد یا خیر (در اولین اندیس، مقدار وجود دارد یا نه)؟

۲- سپس متد **Current** فراخوانی می‌شود تا مقدار اندیس جاری را استخراج نماید.

۳- متد **Key** برای استخراج کلید (اندیس) مقدار جاری فراخوانی می‌شود (در حالت **foreach($array as $key => $value)** کاربرد دارد).

مراحل فوق در اولین اجرای **foreach** اجرا می‌شود و در تکرارهای بعدی، مراحل زیر اجرا خواهد شد:

٤- متد **Next** برای جلوبردن اندیس آرایه جهت اشاره به خانه بعدی فراخوانی می‌شود.

٥- متد **Valid** برای بررسی معتبربودن اندیس جدید (وجود مقدار در خانه بعدی آرایه) فراخوانی می‌شود.

٦- متد **Current** برای استخراج مقدار آرایه فراخوانی می‌شود.

۷- متد **Key** برای استخراج کلید (اندیس) مقدار آرایه فراخوانی می‌شود.

مراحل فوق تا زمانی که **Valid** مقدار **false** بازگرداند، تکرار می‌شوند. البته مثال فوق به‌دلیل آنکه می‌خواستیم اندیس رشته‌ای داشته‌باشیم، کمی پیچیده به‌نظر می‌رسد. حال همان مثال را با اندیس عددی مشاهده کنید:

```php
class Grades implements Iterator {
    private $key;
    private $result;
    public function __construct() {
        $this->result = array();
        $this->key = -1;
    }
    public function Add($student, $grade) {
        if($grade >= 0 && $grade <= 20) {
            $this->result[] = array('student' => $student, 'grade' => $grade);
        }
    }
    public function Rewind() {
        $this->key = 0;
    }
    public function Current() {
        return $this->result[$this->key];
    }
```

```php
    public function Key() {
        return $this->key;
    }
    public function Next() {
        $this->key++;
    }
    public function Valid() {
        return isset($this->result[$this->key]);
    }
}
$grd = new Grades();
$grd->Add('Ali', 5);
$grd->Add('Mohsen', 15);
$grd->Add('Hamid', 9);
$grd->Add('Reza', 11);
$grd->Add('Ahmad', 17);
foreach($grd as $g) {
    if($g['grade'] >= 10) {
        echo $g['student'] . ' : ' . $g['grade'] . '<br/>' . PHP_EOL;
    }
}
```

برای تمرین، درک تفاوت این کد با کد قبل را به شما واگذار می‌کنیم.

سریال‌کردن (Serialization)

تا اینجا آموختیم که چگونه اشیاء را بسازیم و آنها را تغییر دهیم. حال اگر بخواهیم وضعیت یک شئ را ذخیره کنیم (مثلاً در پایگاه داده‌ها یا فایل) و بعداً آنرا به همان‌صورتی که وجود داشته است بازیابی کنیم، چه‌کاری باید انجام دهیم؟ در **PHP** می‌توانید این‌کار را با سریال‌کردن انجام‌دهید.

سریال‌کردن، فرآیند ثبت وظعیت یک شئ در هر مکانی، اعم از فایل فیزیکی، متغیر، پایگاه داده‌ها و... است. برای بازیابی وضعیت شئ سریال‌شده، فرآیند دیگری موسوم به «خارج‌کردن از سریال» (**Unserialization**) لازم‌است. برای سریال‌کردن هر شئ از تابع **serialize** استفاده می‌شود. همچنین خارج‌کردن هر شئ از سریال به‌کمک تابع **unserialize** انجام می‌شود. به مثال زیر دقت‌کنید:

```php
class Sample {
    public $var1;
    private $var2;
    protected $var3;
    static $var4;
    public function __construct() {
        $this->var1 = 'Value one';
        $this->var2 = 'Value two';
        $this->var3 = 'Value three';
        Sample::$var4 = 'Value four';
    }
}
$obj = new Sample();
echo '<pre>' . print_r($obj, true) . '</pre>' . PHP_EOL;
echo Sample::$var4 . '<br/>' . PHP_EOL;
$serialObj = serialize($obj);
unset($obj);
echo '<pre>' . $serialObj . '</pre>' . PHP_EOL;
$unserialObj = unserialize($serialObj);
echo '<pre>' . print_r($unserialObj, true) . '</pre>' . PHP_EOL;
echo Sample::$var4 . '<br/>' . PHP_EOL;
/* Output:
Sample Object
(
    [var1] => Value one
    [var2:Sample:private] => Value two
    [var3:protected] => Value three
)

Value four

O:6:"Sample":3:{s:4:"var1";s:9:"Value one";s:12:"Samplevar2";s:9:"Value two";s:7:"*var3";s:11:"Value three";}

Sample Object
(
    [var1] => Value one
    [var2:Sample:private] => Value two
    [var3:protected] => Value three
)

Value four
*/
```