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

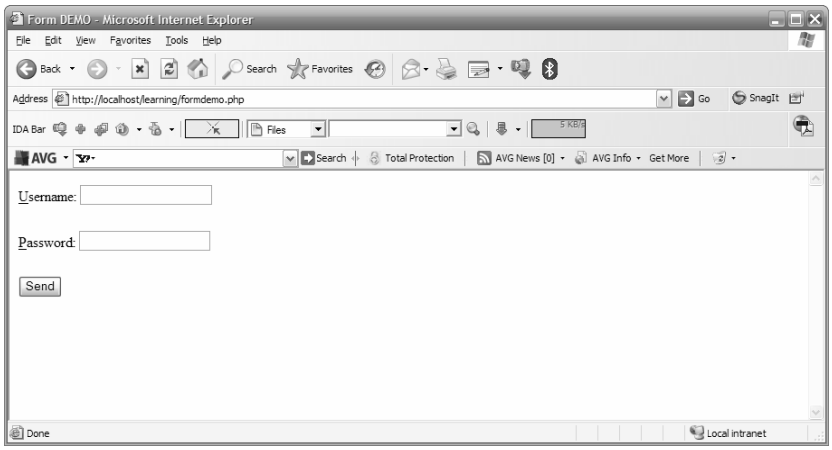
در مثال فوق، دقت‌کنید که برای انتخاب گزینه Married، می‌توانید هم برروی کادر انتخاب و هم برروی نوشته Married کلیک‌کنید. همچنین، امکان انتخاب این گزینه توسط کلیدهای Alt+M وجود دارد. البته کلید میانبر، به‌دلیل خاصیت accesskey تگ LABEL ایجاد شده‌است و برای اطلاع کاربر از این ویژگی، حرف M از کلمه Married،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

تفاوت متدهای GET و POST در فرم

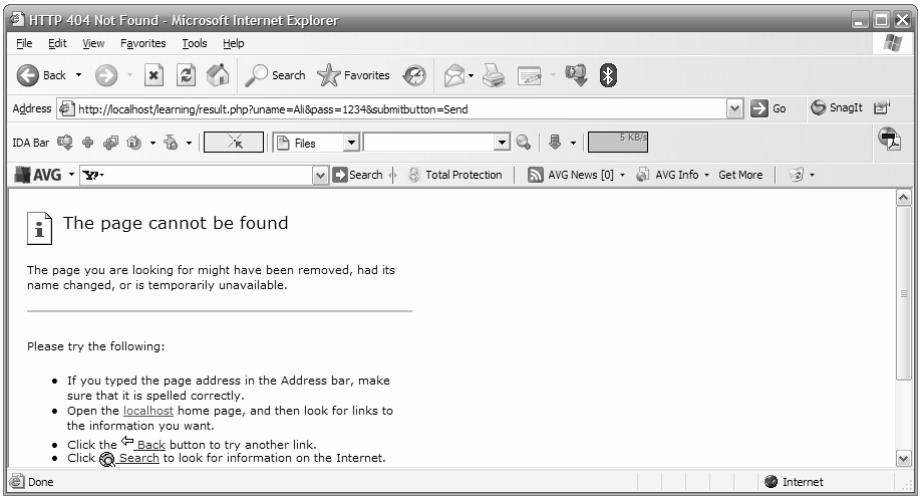
حال که به‌اندازه کافی برای ایجاد یک فرم، اطلاعات دراختیار داریم، اجازه‌دهید که تفاوت متدهای GET و POST را در عمل مشاهده‌کنیم. به فرم زیر دقت‌کنید:

```
<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:

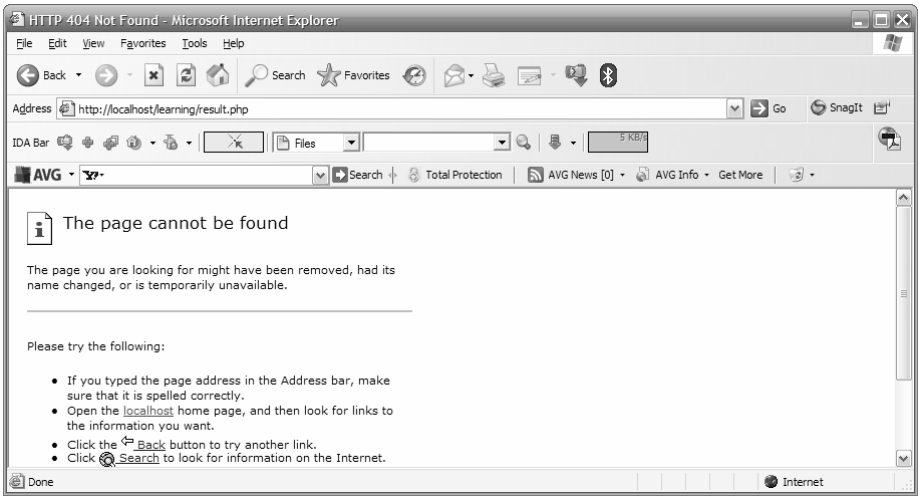
```
URL?category=computer
```

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");
```

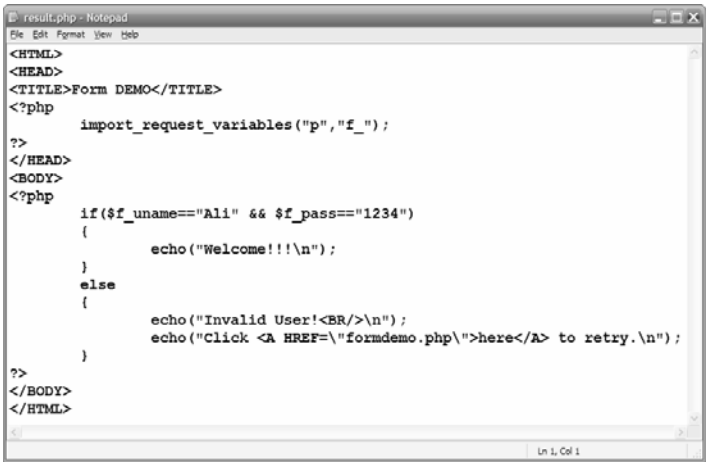
تنها ایراد این روش، طولانی‌شدن نام متغیرها برای دسترسی به پارامترها است ($_POST به ابتدای همه اسامی اضافه می‌شود).

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.