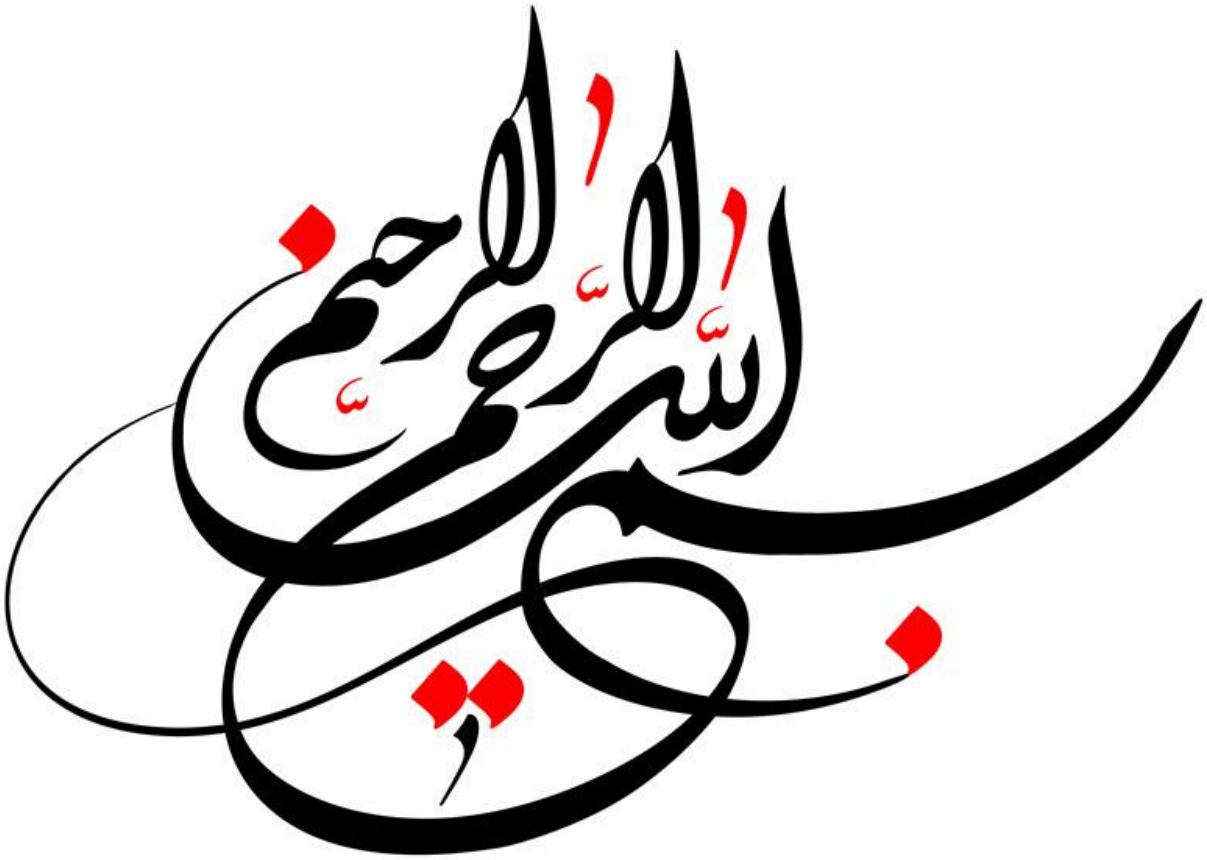


آموزش PHP صفر تا صد

Ůɔʒʞə p ɛǸq̄ɑ

PHP

مولف : میثم شادکام

Contents

PHPچیست و چرا به آن نیاز داریم؟ 8
برای شروع کار با PHP چه چیزهایی لازم دارید؟ 9
استفاده از PHP 19
ادغام کدهای HTML و PHP 28
انواع خطاها در PHP 31
کاراکترهای کنترلی 36
متغیر 38
انواع داده 40
تست نوع متغیر 43
تغییر نوع متغیر 44
ثابت ها 46
عبارات و عملگرها 48
عملگرهای ریاضی 49
عملگرهای تخصیصی 52
عملگرهای مقایسه ای 54
عملگرهای منطقی 56
عملگرهای بیتی 59
عملگر رشته 64
رشته ها 65
استفاده از Heredocs و Nowdocs 68
آرایه ها 70
دستورات شرطی 73
دستور if 74
دستور if...else 77
دستور if else if 78
عملگر سه تایی 79
دستور Switch 80
دستورات تکرار 82

دستور While 83
حلقه do while 85
حلقه for 87
حلقه foreach 89
خارج شدن از حلقه با استفاده از break و continue 91
تابع 93
مقدار برگشتی از یک تابع 95
پارامترها و آرگومان ها 98
پارامترهای اختیاری 100
ارسال آرگومان به روش ارجاع و مقدار 101
محدوده متغیر 103
بازگشت (Recursion) 107
سربارگذاری متدها 109
برنامه نویسی شیء گرا 111
کلاس 112
سازنده 115
مخرب 118
سطح دسترسی 119
کپسوله سازی 120
خواص 121
وراثت 124
سطح دسترسی Protect 126
trait 128
فضای نام 130
Overriding 133
کلاسهای انتزاعی 135
کلاس final و متد final 137
اعضای Static 139
ثابت های کلاس 141

عملگر instanceof 142
چند ریختی 144
ثابت های جادویی 146
متدهای جادویی (Magic Methods) 148
آرایه های فوق سراسری (super globals) 154
مدیریت استثناءها و خطایابی 157
استثناءهای اداره نشده 158
دستورات try و catch 159
دستورات include و require 160
تگ input 165
تگ Form 167
دکمه ارسال (submit) 172
جعبه متن (text) 173
دکمه رادیویی (Radio) 178
چک باکس (checkbox) 184
لیست کشویی (Select) 186
لیست کشویی (Select) 188
تابع date 190
مهر زمانی و توابع time() و mktime() 193
کار با فایل ها 195
به دست آوردن اطلاعات در مورد فایل 196
به دست آوردن نام فایل و پوشه 198
باز و بسته کردن یک فایل 199
نوشتن در فایل 201
خواندن از فایل 206
خواندن فایل CSV 210
ایجاد، حذف، کپی، برش و تغییر نام فایل ها 213
به درست آوردن موقعیت و انتقال اشاره گر به مکانی دیگر 216
آپلود فایل 218

کار با پوشه ها 224
پروتکل های ارسال ایمیل 225
ارسال ایمیل در PHP 227
کوکی (cookie) چیست؟ 233
Session (سشن) چیست 241
امنیت در اجزای فرم های HTML 244
تابع htmlspecialchars() 247
تابع strip_tags() 250
زبان نشانه گذاری توسعه پذیر (XML) 253
Document Object Model یا DOM چیست 256
SimpleXML چیست 266
متد addAttribute() 270
متد addChild() 271
متد Attributes() 272
متد children() 273
متد count() 274
متد getName() 275
MYSQL چیست؟ 276
مبانی MYSQL 277
ایجاد جدول و دیتابیس به روش کدنویسی 280
ایجاد جدول و دیتابیس با phpMyAdmin 289
ارتباط با سرور و بانک اطلاعاتی (MySQL) 294
اضافه کردن رکورد به بانک (MySQL) 302
انتخاب یا خواندن رکورد از بانک (MySQL) 304
ویرایش رکوردهای بانک (MySQL) 306
حذف رکورد از بانک (MySQL) 307

PHPچیست و چرا به آن نیاز داریم؟

PHP شاید عمومی ترین زبان اسکریپتی تحت وب باشد و استفاده از آن روز به روز بیشتر می شود. با استفاده از PHP می توان صفحات ورودی ایجاد کرده، جزییات ورود اطلاعات از طریق فرم ها را بررسی نمایید، فروم ،گالری تصاویر و بسیاری از چیزهای دیگر را ایجاد کنید.

PHP به عنوان یک زبان برنامه نویسی سمت سرور مشهور است. چون نمی تواند در داخل کامپیوتر شما اجرا شود. دیگر زبان های برنامه نویسی که احتمالا اسم آنها را شنیده اید عبارتند از : ASP ، Python، Perl . در باره این زبان ها لازم نیست بیشتر از اینها بدانید چون در این آموزش فرض بر این است که شما برنامه نویس حرفه ای نیستید!

عمومی ترین تعریف PHP این است که PHP مخفف کلمات Hypertext Pre-processor می باشد. شاید برایتان این سوال پیش بیاید که مخفف کلمات فوق HPP است. درست است، اما در نسخه های قبلی برنامه PHP را به عنوان مخفف کلمات Personal Home Page تعریف کرده اند. که مخفف آنها PHP می شود. در این آموزش به شما نحوه اجرای PHP را آموزش می دهیم و متوجه خواهید شد که یادگیری آن بسیار آسان است.

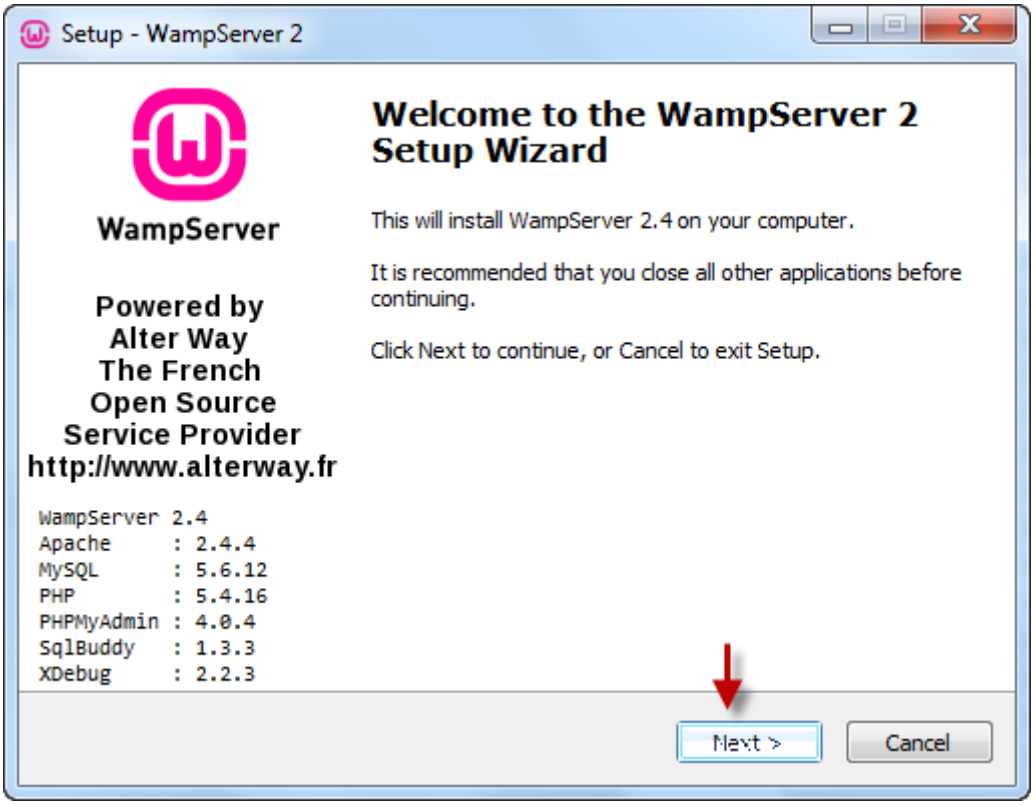
برای شروع کار با PHP چه چیزهایی لازم دارید؟

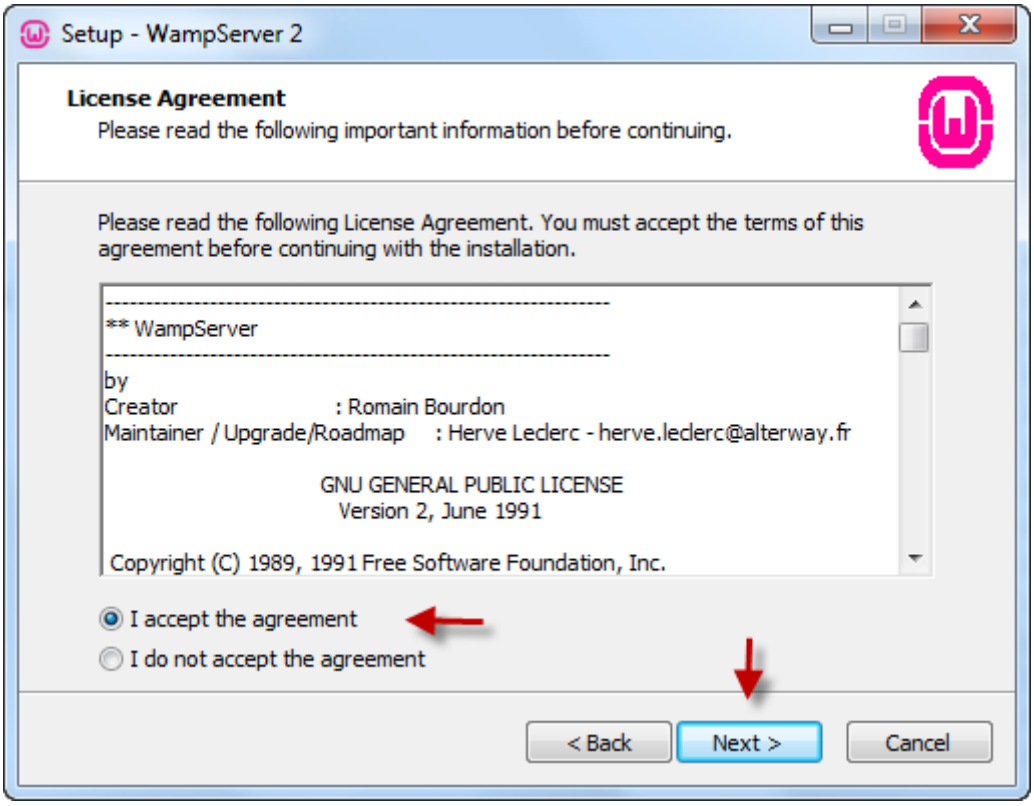
برای نوشتن و تست کدهای **PHP** ابتدا به یک سرور نیاز دارید .خوشبختانه، لازم نیست که یک سرور خریداری کنید و نیاز به خرج پول نیست .درست است که **PHP** یک زبان برنامه نویسی عمومی است اما چون یک زبان اسکریپتی سمت سرور است باید یک هاست (مقداری از فضای وب) که **PHP** را پشتیبانی کند خریداری کنید و یا کاری کنید که کامپیوترتان به عنوان یک سرور عمل کند. چون **PHP** نمی تواند بر روی کامپیوتر اجرا شود. این زبان بر روی سرور **(server)** اجرا شده و نتایج را به کامپیوتر سرویس گیرنده **(client)** برگشت می دهد .نگران راه اندازی و تست کدها بر روی کامپیوترتان نباشید. یک راه ساده برای اجرای کدها بر روی کامپیوتر استفاده از نرم افزاری به نام **Wampserver** می باشد.این نرم افزار تمام موارد لازم برای اجرای کدها را نصب می کند. نحوه نصب و استفاده از این نرم افز ار را برای شما توضیح می دهیم.برای دانلود این نرم افزار بر روی لینک زیر کلیک کنید:

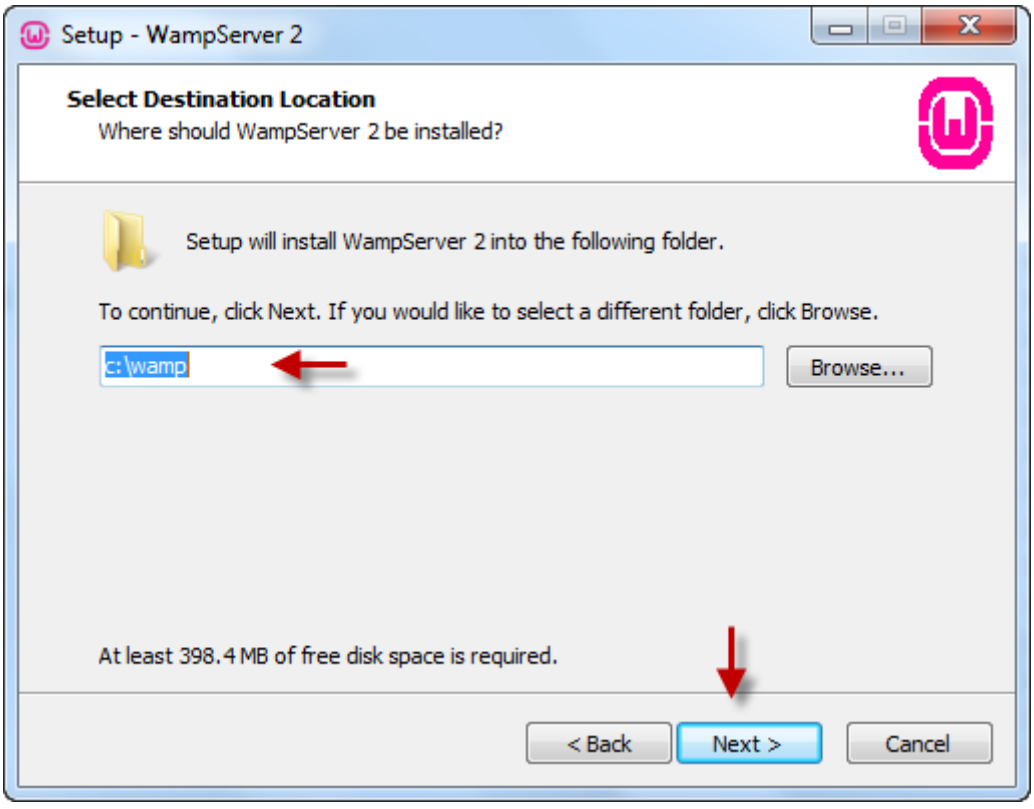
Download Wampserver

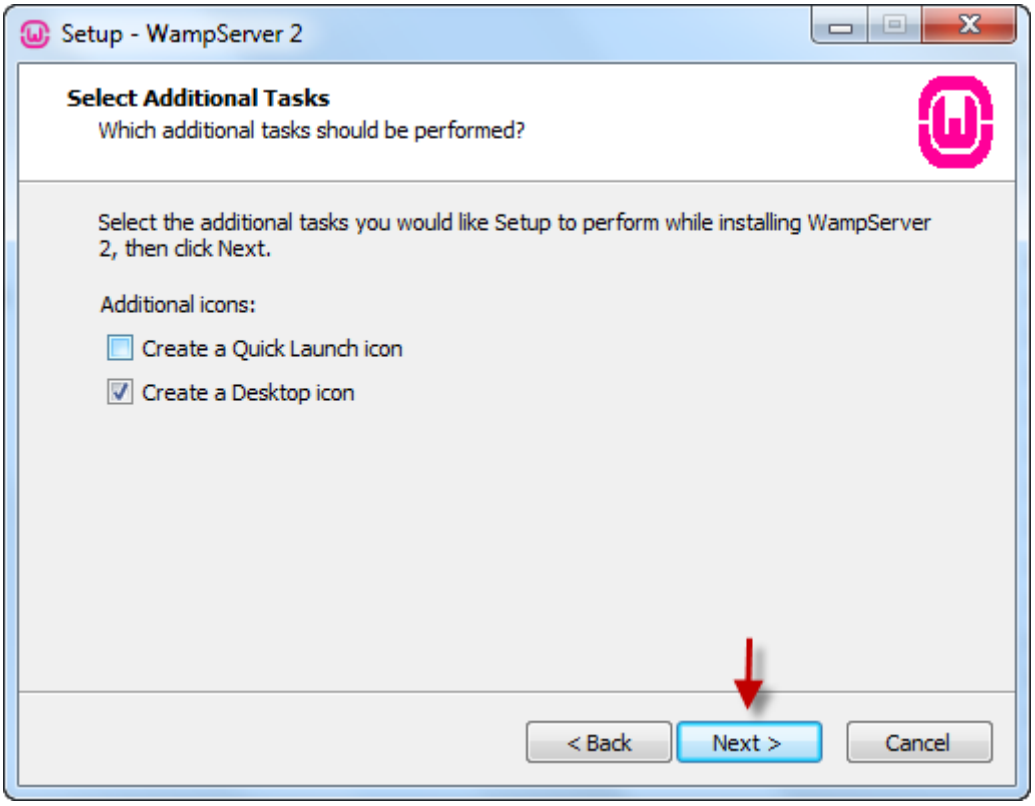
نصب و تست Wampserver

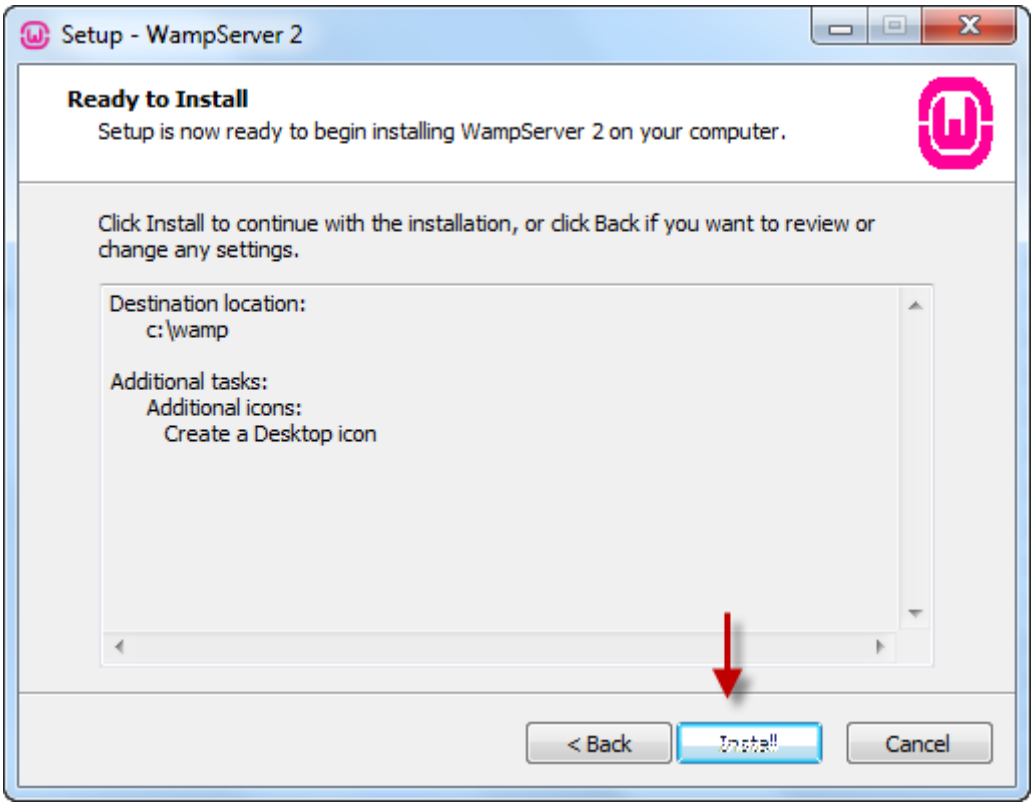
بعد از دانلود Wampserver مراحل زیر را برای نصب آن طی کنید:

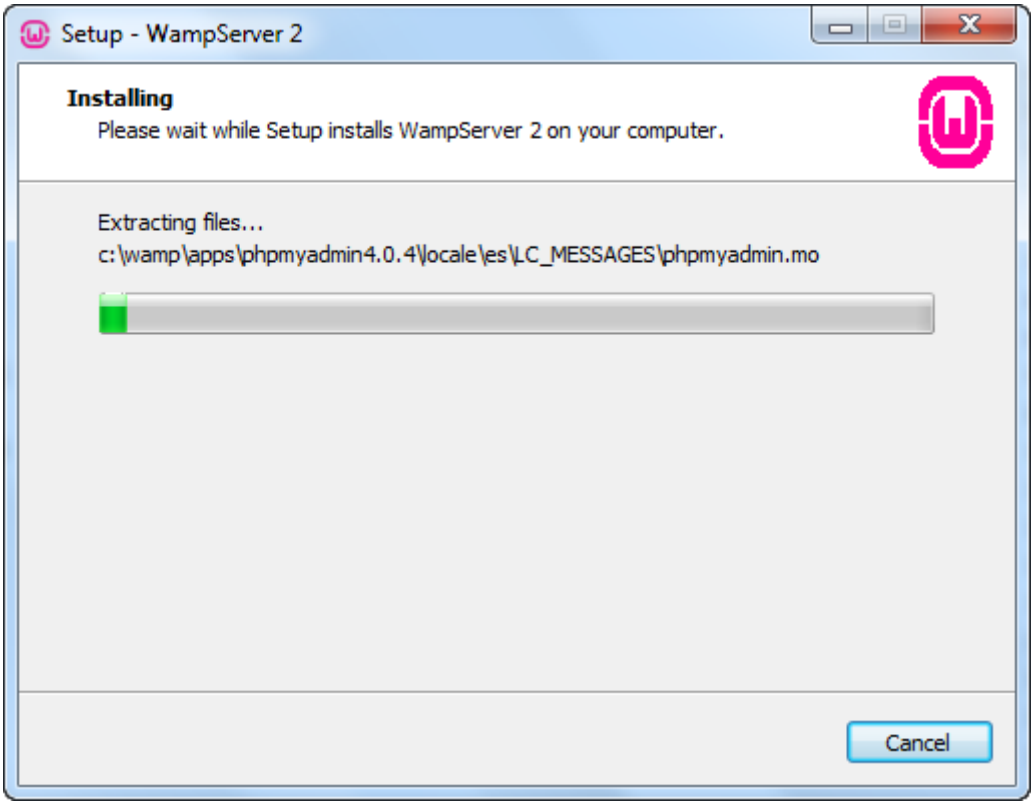


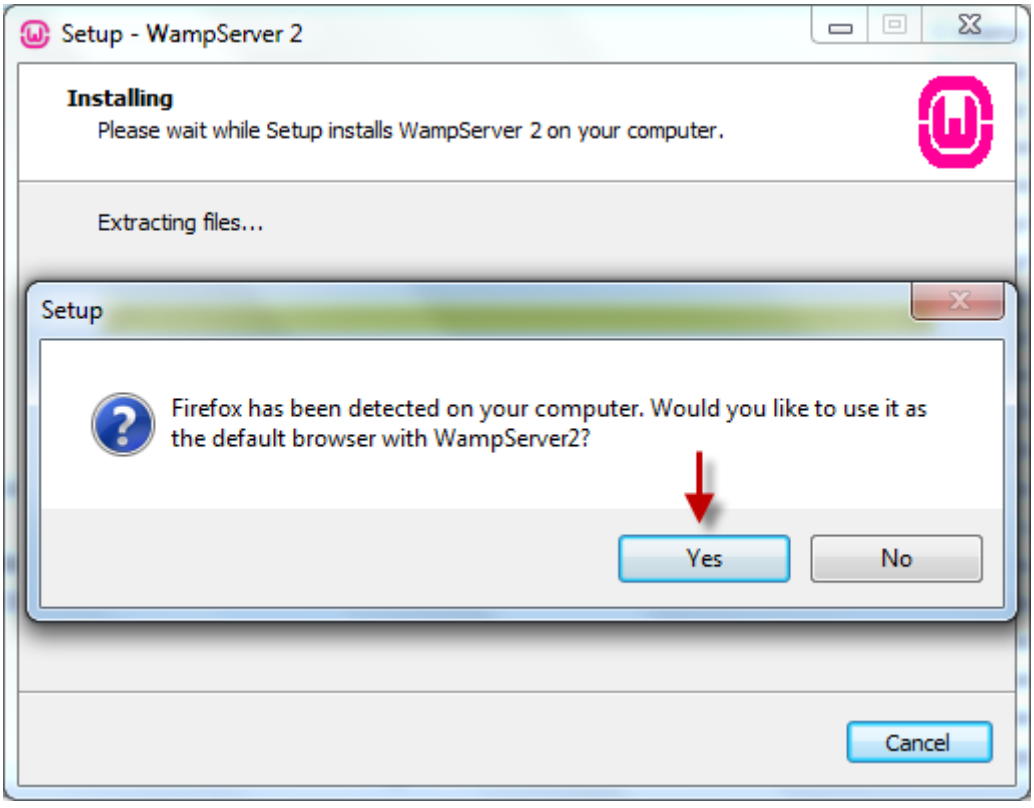

Setup - WampServer 2
License Agreement
Please read the following important information before continuing.
Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.
** WampServer
by
Creator : Romain Bourdon
Maintainer / Upgrade/Roadmap : Herve Leclerc - herve.leclerc@alterway.fr
GNU GENERAL PUBLIC LICENSE
Version 2, June 1991
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
I accept the agreement
I do not accept the agreement
< Back
Next >
Cancel

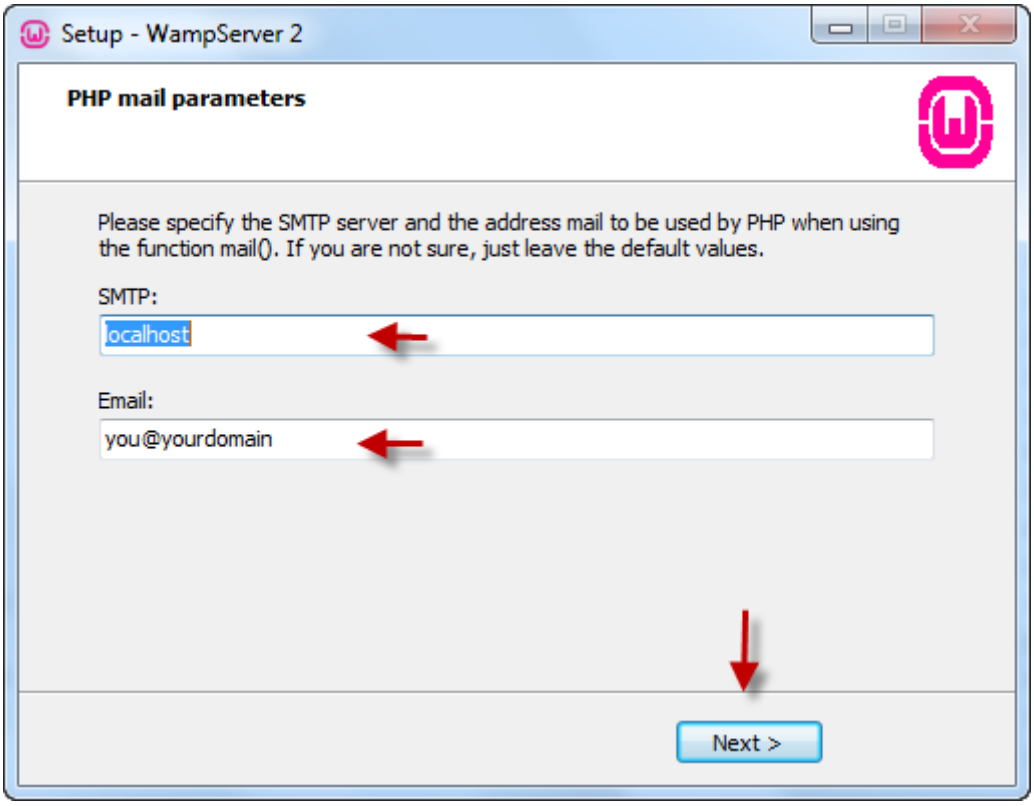

Setup - WampServer 2
Select Destination Location
Where should WampServer 2 be installed?
Setup will install WampServer 2 into the following folder.
To continue, click Next. If you would like to select a different folder, click Browse.
c:\wamp
Browse...
At least 398.4 MB of free disk space is required.
< Back
Next >
Cancel

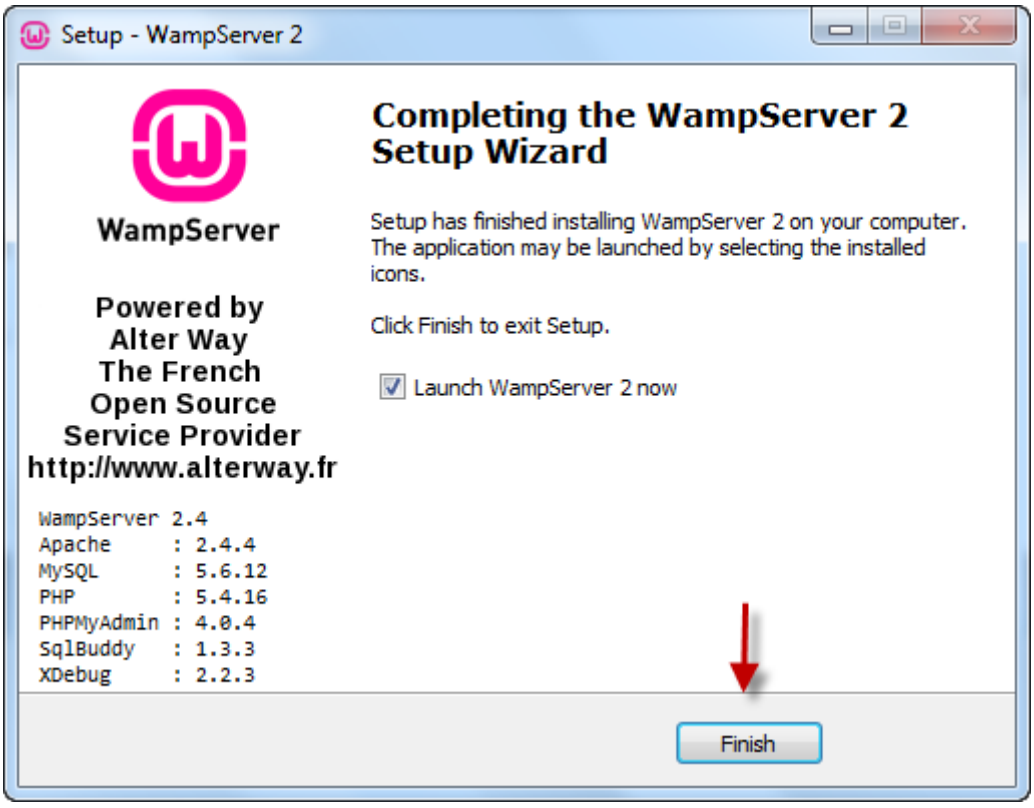

Setup - WampServer 2
Select Additional Tasks
Which additional tasks should be performed?
Select the additional tasks you would like Setup to perform while installing WampServer 2, then click Next.
Additional icons:
Create a Quick Launch icon
Create a Desktop icon
< Back
Next >
Cancel


Setup - WampServer 2
Ready to Install
Setup is now ready to begin installing WampServer 2 on your computer.
Click Install to continue with the installation, or click Back if you want to review or change any settings.
Destination location:
c:\wamp
Additional tasks:
Additional icons:
Create a Desktop icon
< Back
Install
Cancel


Setup - WampServer 2
Installing
Please wait while Setup installs WampServer 2 on your computer.
Extracting files...
c:\wamp\apps\phpmyadmin4.0.4\locale\es\LC_MESSAGES\phpmyadmin.mo
Cancel

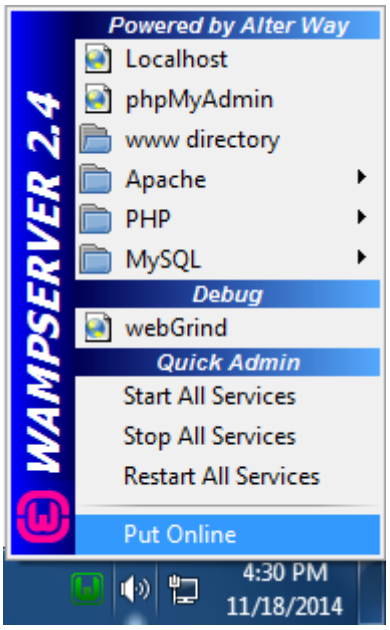

Setup - WampServer 2
Installing
Please wait while Setup installs WampServer 2 on your computer.
Extracting files...
Setup
Firefox has been detected on your computer. Would you like to use it as the default browser with WampServer2?
Yes
No
Cancel


Setup - WampServer 2
PHP mail parameters
Please specify the SMTP server and the address mail to be used by PHP when using the function mail(). If you are not sure, just leave the default values.
SMTP:
localhost
Email:
you@yourdomain
Next >

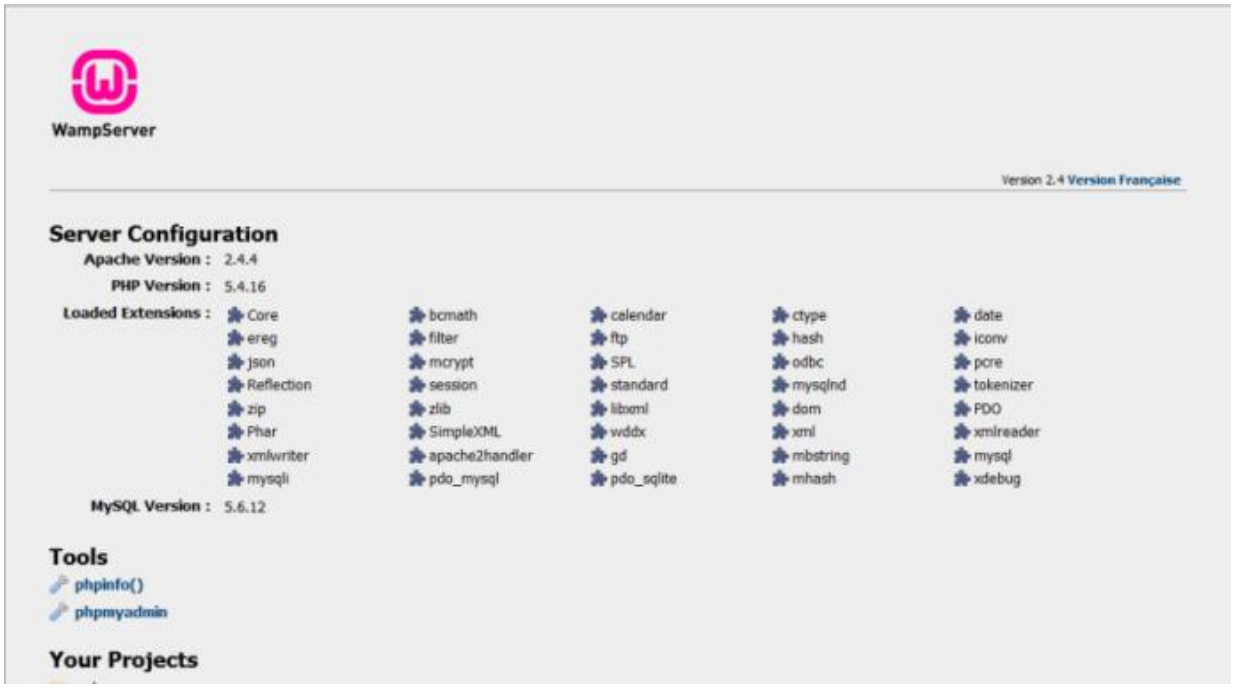

Setup - WampServer 2
WampServer
Powered by
Alter Way
The French
Open Source
Service Provider
http://www.alterway.fr
WampServer 2.4
Apache : 2.4.4
MySQL : 5.6.12
PHP : 5.4.16
PHPMyAdmin : 4.0.4
SqlBuddy : 1.3.3
XDebug : 2.2.3
Completing the WampServer 2 Setup Wizard
Setup has finished installing WampServer 2 on your computer. The application may be launched by selecting the installed icons.
Click Finish to exit Setup.
Launch WampServer 2 now
Finish

بعد از نصب آن یک آیکون جدید مانند شکل زیر در سمت راست و پایین صفحه مانیتور و درست کنار آیکون ساعت مشاهده خواهید کرد:

بر روی آیکونی که می بینید کلیک کنید تا یک منو باز شود.در منوی ظاهر شده می توانید سرور را متوقف کرده، از آن خارج شوید و صفحات پیکربندی را مشاهده نمایید.بر روی localhost کلیک کنید مشاهده می کنید که یک صفحه ظاهر می شود:



(localhost به سروری که بر روی کامپیوتر شما نصب شده است رجوع می کند،راه دیگر برای رجوع به سرور استفاده از IP ، 127.0.0.1 می باشد.)



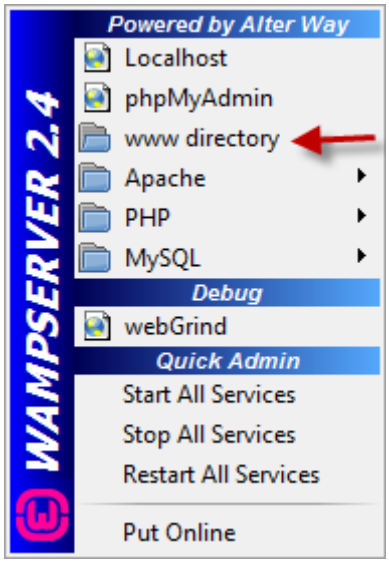
در قسمت Tools بر روی phpinfo() کلیک کنید. اگر همه چیز به خوبی پیش برود صفحه ای به شکل زیر مشاهده خواهید کرد که نشان دهنده نصب صحیح سرور بوده و شما می توانید شروع به کدنویسی کنید.

PHP Version 5.4.16

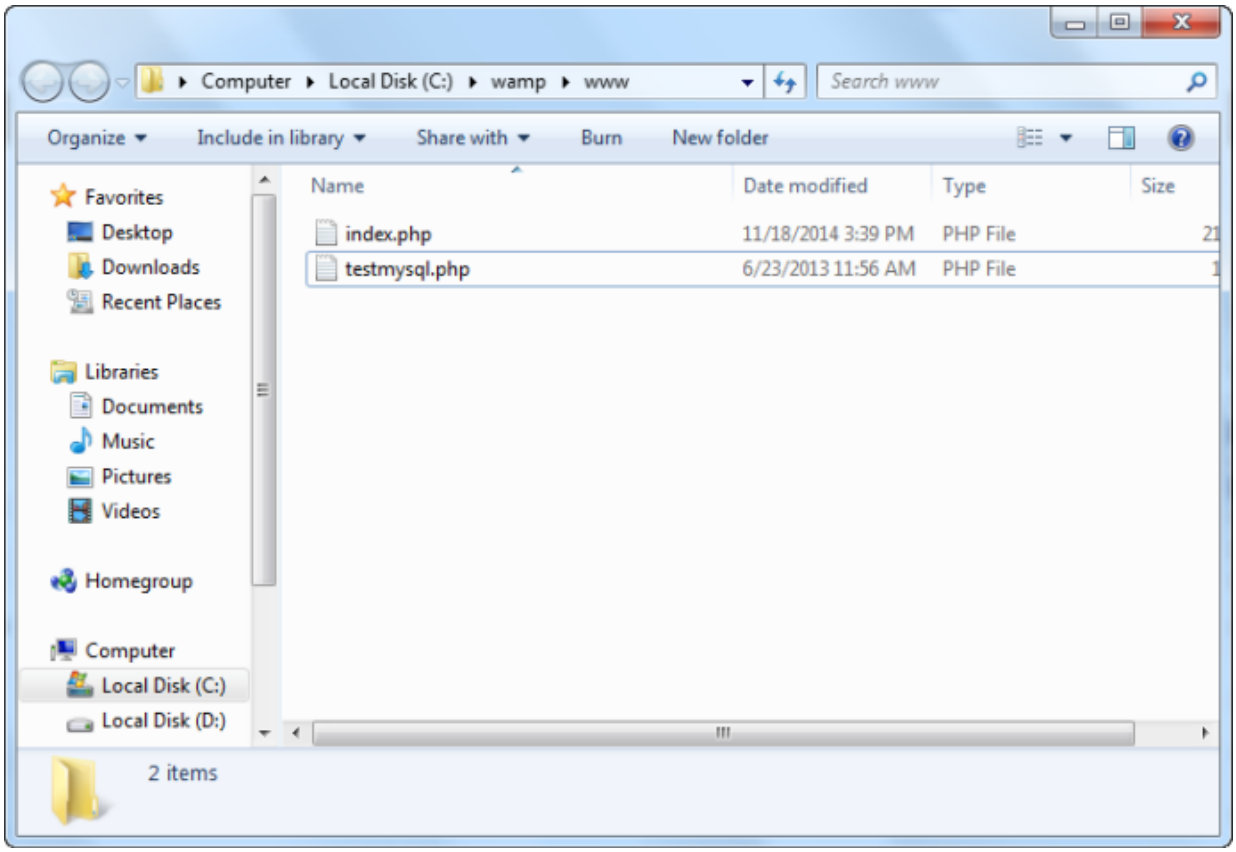
System	Windows NT ARTA-PC 6.1 build 7600 (Windows 7 Ultimate Edition) i586
Build Date	Jun 5 2013 20:58:05
Compiler	MSVC9 (Visual C++ 2008)
Architecture	x86
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8-11g=C:\php-sdk\oracle\instantclient11\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--disable-static-analyze" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	C:\wamp\bin\apache\apache2.4.4\bin\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini	(none)

ذخیره فایل های PHP

وقتی که یک صفحه PHP ایجاد کردید لازم است که آن را در پوشه WWW برنامه Wampserver ذخیره کنید. این پوشه با کلیک بر روی آیکون برنامه قابل مشاهده است. به شکل زیر توجه کنید:



وقتی بر روی WWW کلیک می کنید پنجره ای به شکل زیر ظاهر می شود. در داخل این پنجره ممکن است فقط دو فایل indexو testmysql وجود داشته باشند.



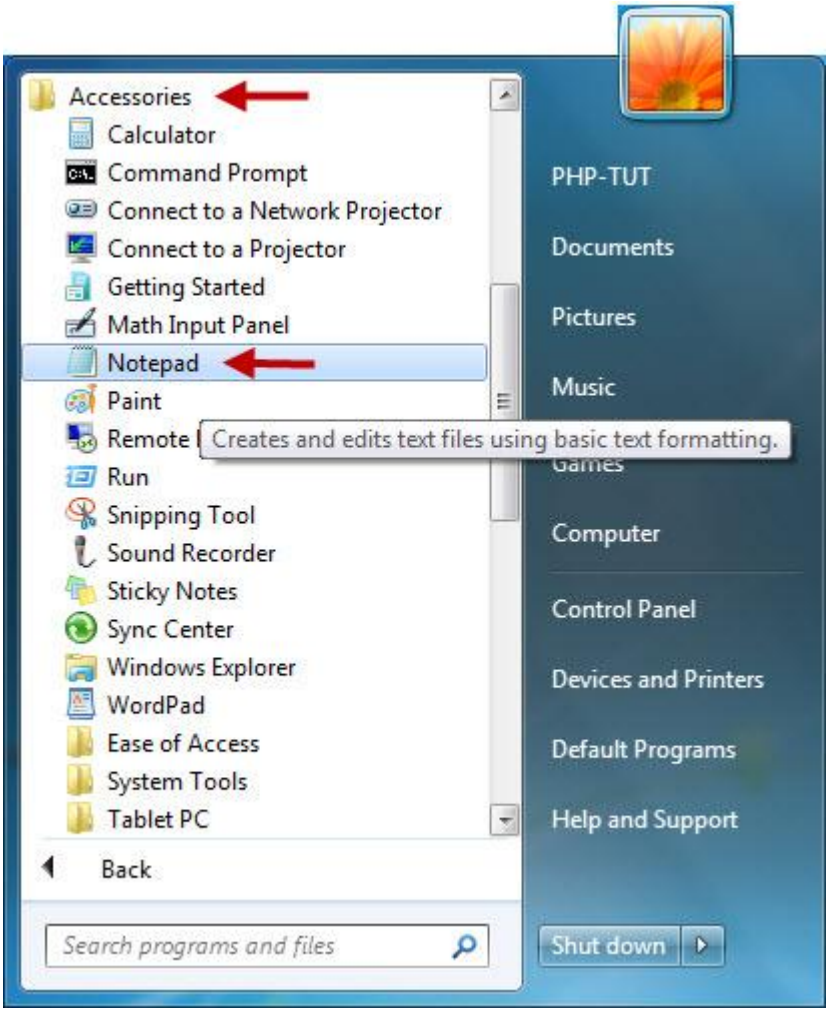
پوشه WWW معمولا در مسیر زیر قرار دارد:

```
c:/wamp/www/
```

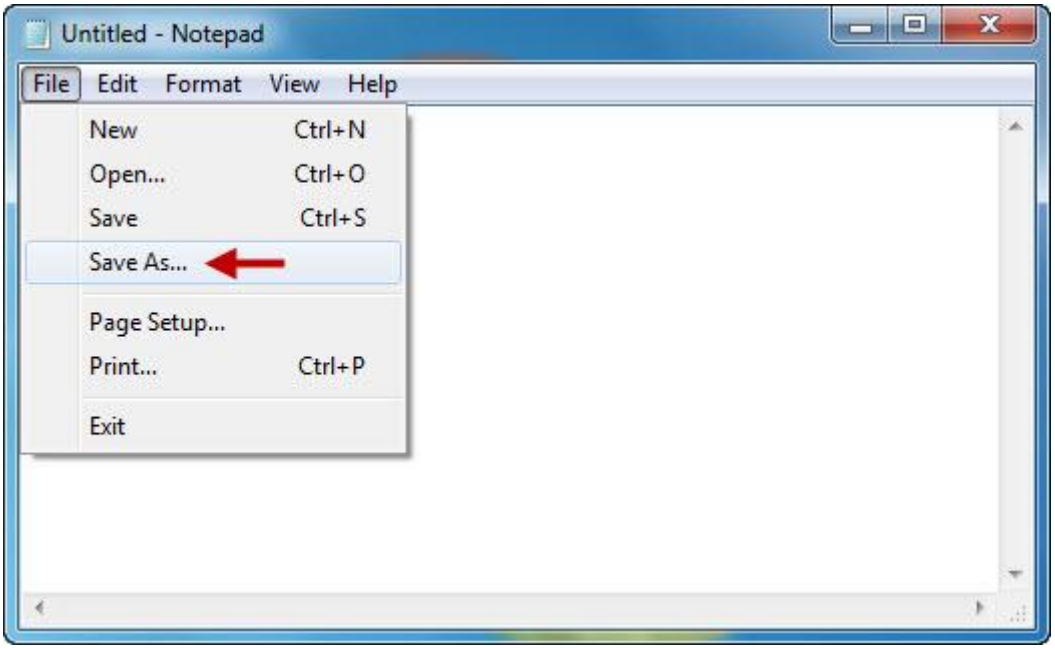
پس برای اجرای فایل های PHP آنها را در این قسمت ذخیره کنید.

استفاده از PHP

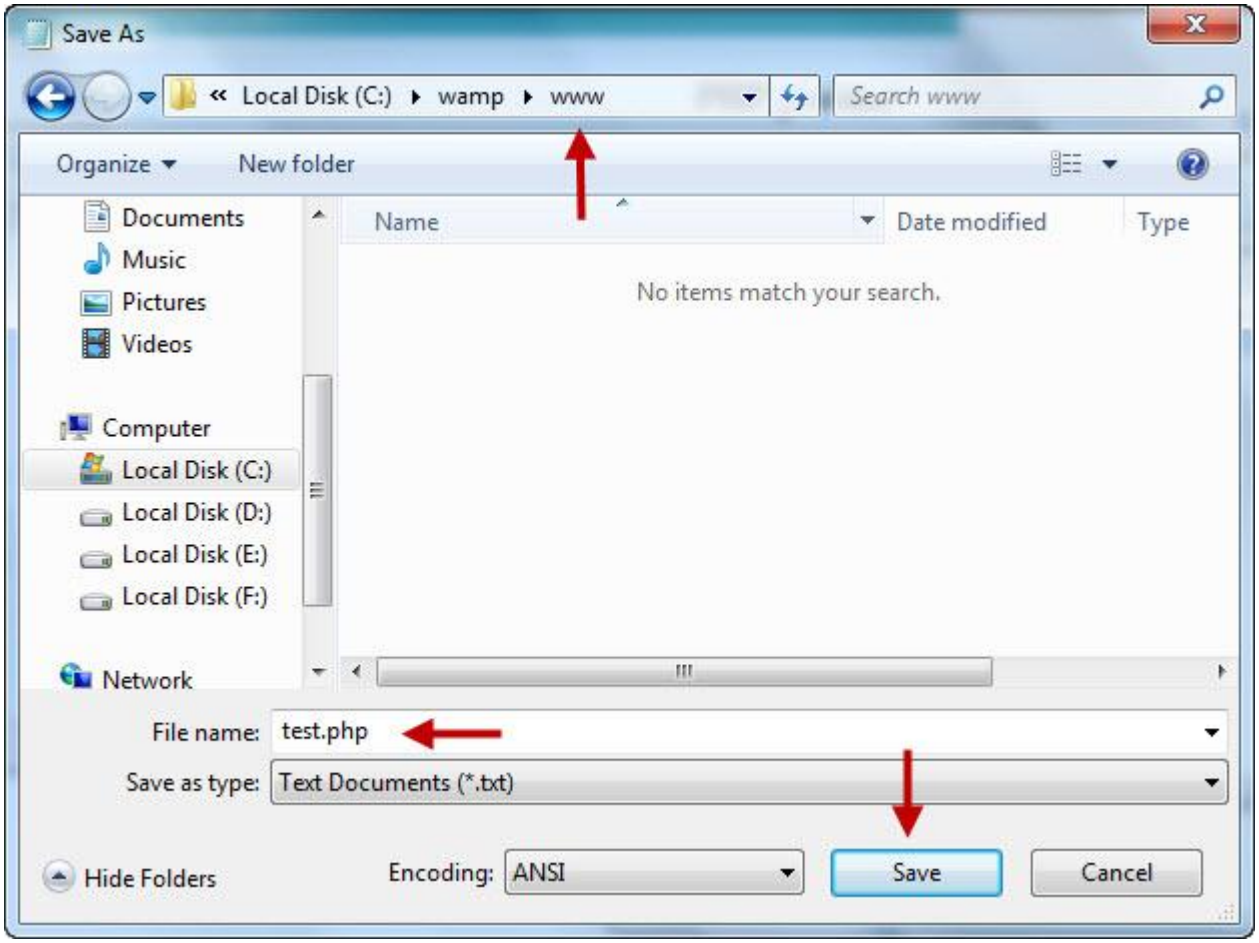
برای شروع کار با PHP ابتدا برنامه notepad ویندوز را از مسیر زیر اجرا کنید:

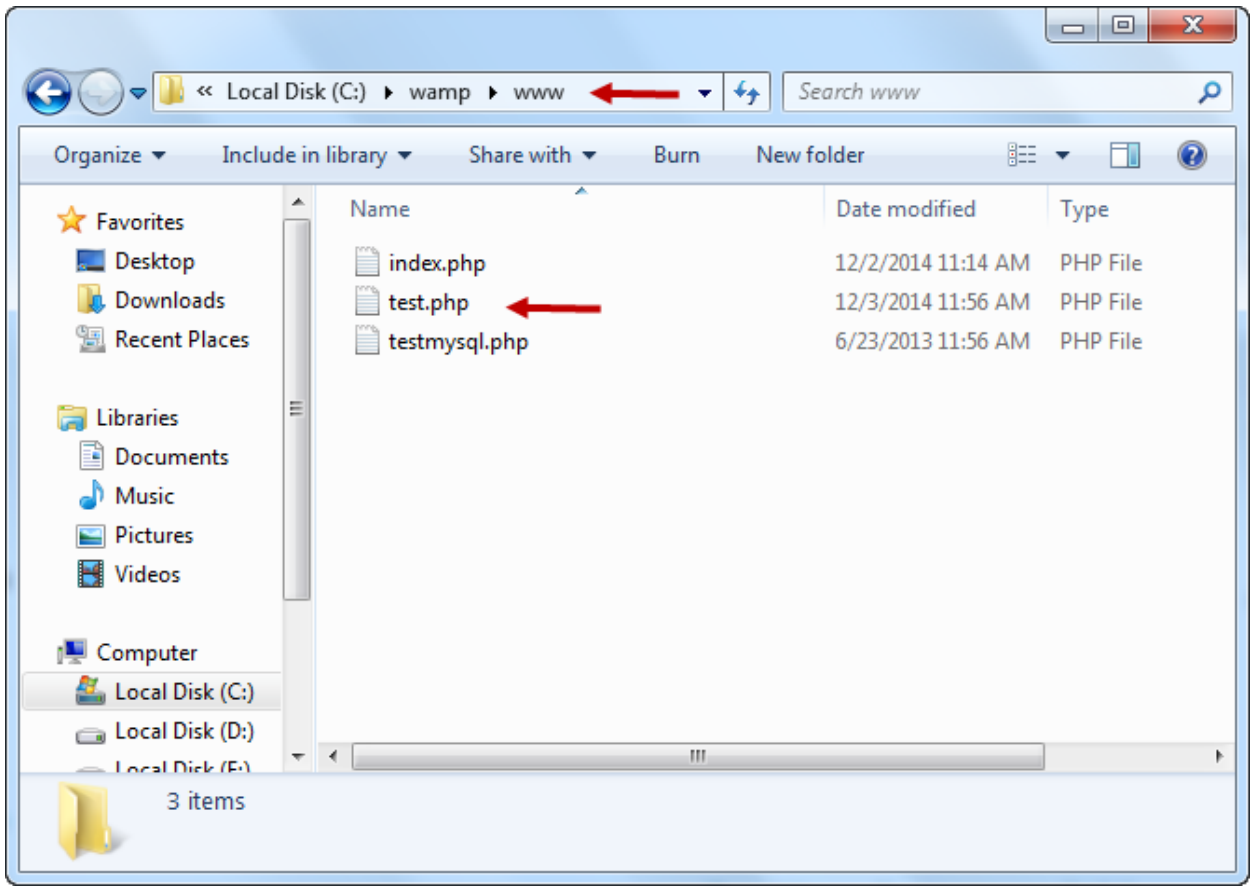


بعد از باز شدن برنامه از منوی File بر روی گزینه Save as کلیک کنید:



بعد از کلیک بر روی گزینه Save as پنجره ای به شکل زیر باز می شود که با استفاده از آن یک فایل با پسوند .php ، با نام test.php در پوشه www ایجاد کنید:


Save As
« Local Disk (C:) ▸ wamp ▸ www
Search www
Organize
New folder
Documents
Music
Pictures
Videos
Computer
Local Disk (C:)
Local Disk (D:)
Local Disk (E:)
Local Disk (F:)
Network
Name
Date modified
Type
No items match your search.
File name: test.php
Save as type: Text Documents (*.txt)
Hide Folders
Encoding: ANSI
Save
Cancel



یک فایل **php** می تواند شامل کدها یا تگ های **HTML** هم باشد. برای این منظور فایل ایجاد شده را با یکی از برنامه های ویرایشگر متن مانند **Notepad** ویندوز، **Dreamweaver** و یا برنامه **++Notepad** (پیشنهاد ما) باز کرده و کدهای **HTML** زیر را در داخل آن بنویسید:

```
<html>
<head>
    <title>PHP Test</title>
</head>
<body>

</body>
</html>
```

اضافه کردن کدهای PHP

برای اضافه کردن کدهای PHP چهار روش وجود دارد:

در حالت اول که استانداردترین حالت است از <?php و ?> اااستفاده می شود. یعنی شما دستورات PHP را در داخل این دو علامت قرار می دهید:

```
<?php … ?>
```

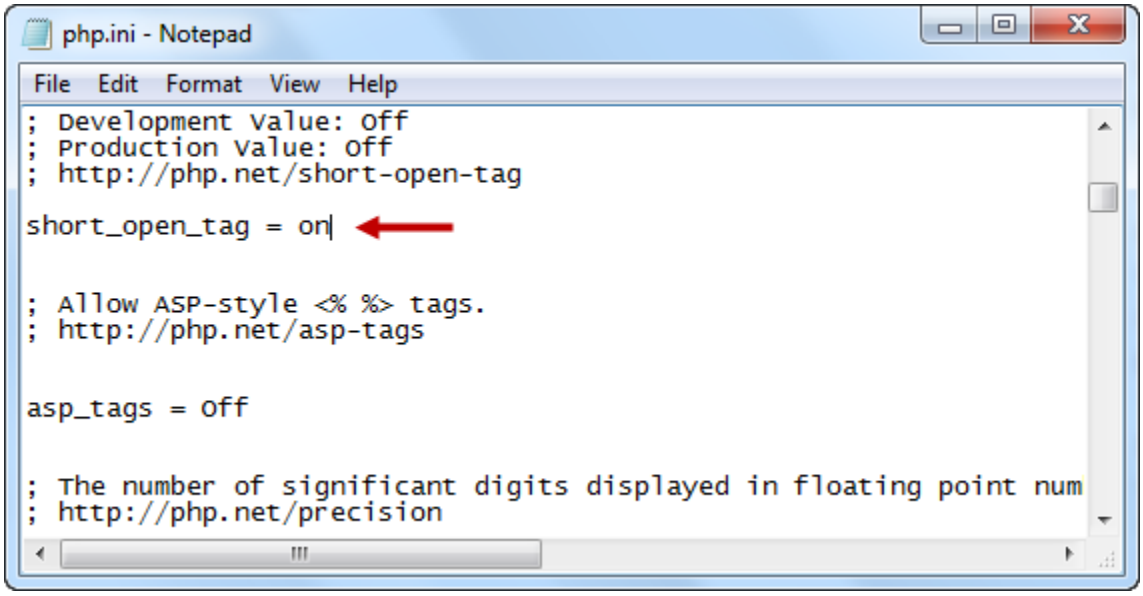
در حالت دوم که خلاصه شده حالت بالاست از <? و ?> استفاده می شود:

```
<?  … ?>
```

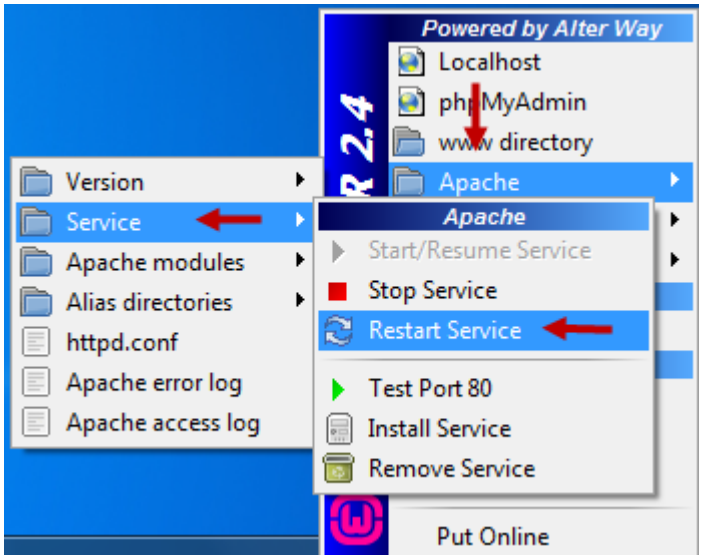
برای استفاده از روش بالا باید تغییراتی را در فایل php.ini اعمال کنید و آن را فعال کنید، بدین منظور به مسیر زیر بروید:

```
C:\wamp\bin\apache\Apache2.4.4\bin
```

و فایل php.ini را باز کرده و مقدار short_open_tag را به صورت زیر تغییر دهید:



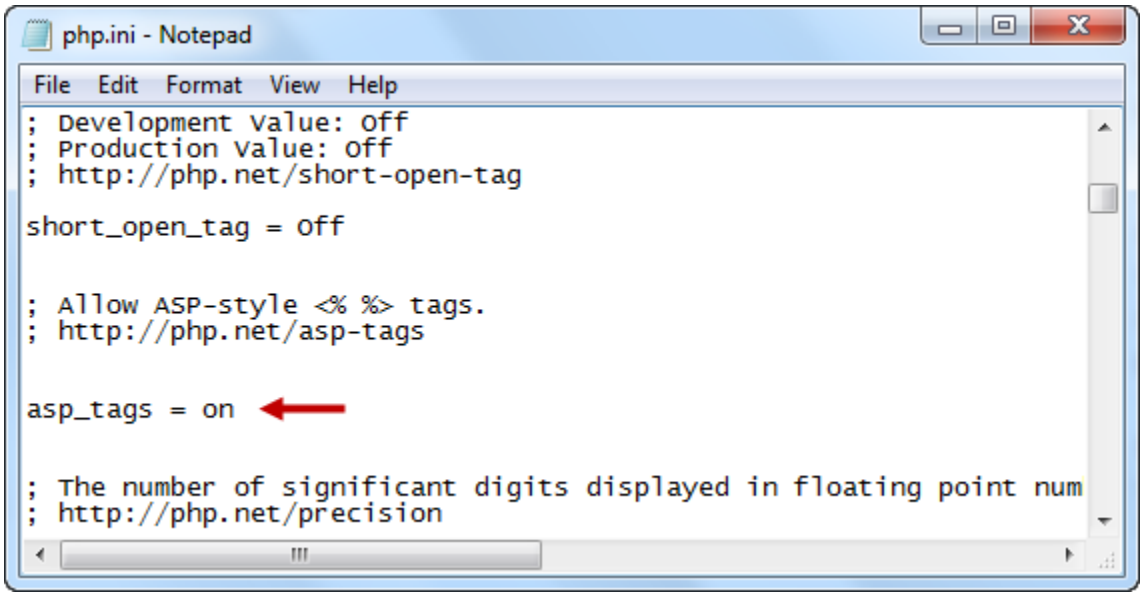
بعد از اعمال تغییرات بالا، apache را ریستارت کنید:



در حالت سوم از %> و <% استفاده می شود:

```
<% ... %>
```

برای استفاده از روش بالا باید تغییراتی را در فایل php.ini اعمال کنید و آن را فعال کنید، بدین منظور باید مقدار asp_tags را به صورت زیر تغییر دهید:



و در حالت چهارم نیز به صورت زیر عمل می شود:

```
<script language="php">...</script>
```

خروجی متن در PHP

برای چاپ مقادیر در **PHP** و نمایش آنها در مرورگر از دو دستور **echo** و **print** استفاده می شود. در پایان هر یک از این دستورات باید از علامت سمیکالن (;) استفاده شود. البته استفاده از سمیکالن در آخرین دستور اختیاری است.

```
<?php
    echo "Hello World";
    print "Hello World"
?>
```

برای تولید خروجی می توان از علامت =?> هم استفاده کرد. این ساختار در **PHP 5.4** ارائه شده است:

```
<?= "Hello World" ?>
```

به یاد داشته باشید که برای نمایش خروجی کد را در داخل تگ **body** به صورت زیر بنویسید:

```
<html>
<head>
    <title>PHP Test</title>
</head>
<body>

<?php echo "Hello World"; ?>

</body>
</html>
```

یک برنامه ساده با PHP

برای اینکه با عملکرد **PHP** بیشتر آشنا شوید ابتدا فایل **test.php** را که در پوشه **www** ایجاد کرده اید را با ویرایشگر متن باز کرده و کدهای زیر را داخل آن بنویسید:

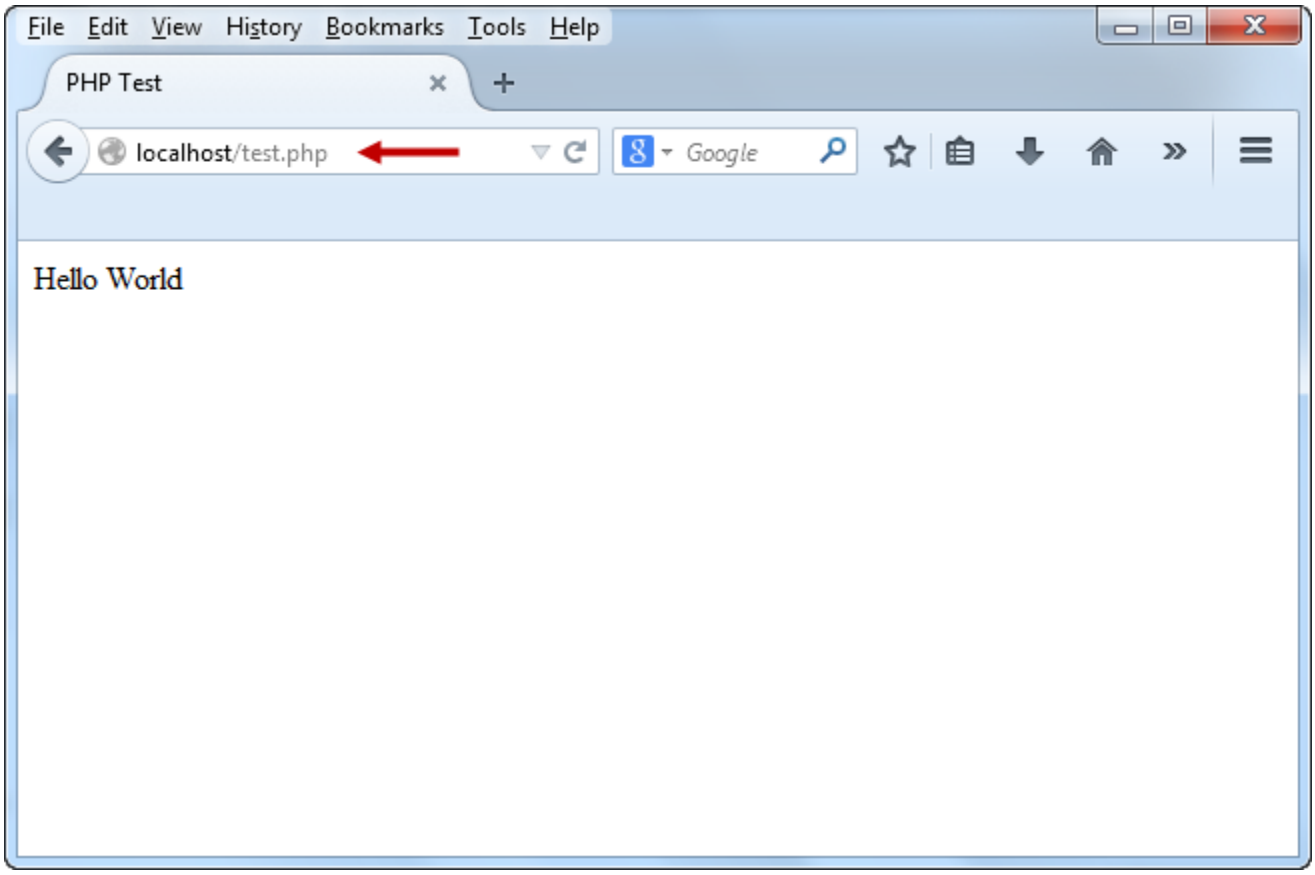
```
<html>
<head>
    <title>PHP Test</title>
</head>
```

```
<body>

<?php echo "Hello World"; ?>

</body>
</html>
```

حال کد را ذخیره کرده و مرورگرتان را باز و دستور زیر را در نوار آدرس آن بنویسید:

```
http://localhost/test.php
```

با زدن دکمه Enter کدهای PHP در سرور پردازش و پس از تبدیل به کدهای HTML، به مرورگر ارسال می شوند. مرورگر هم با مشاهده و خواندن کدهای HTML خروجی مطلوب را به ما ارائه می دهد:

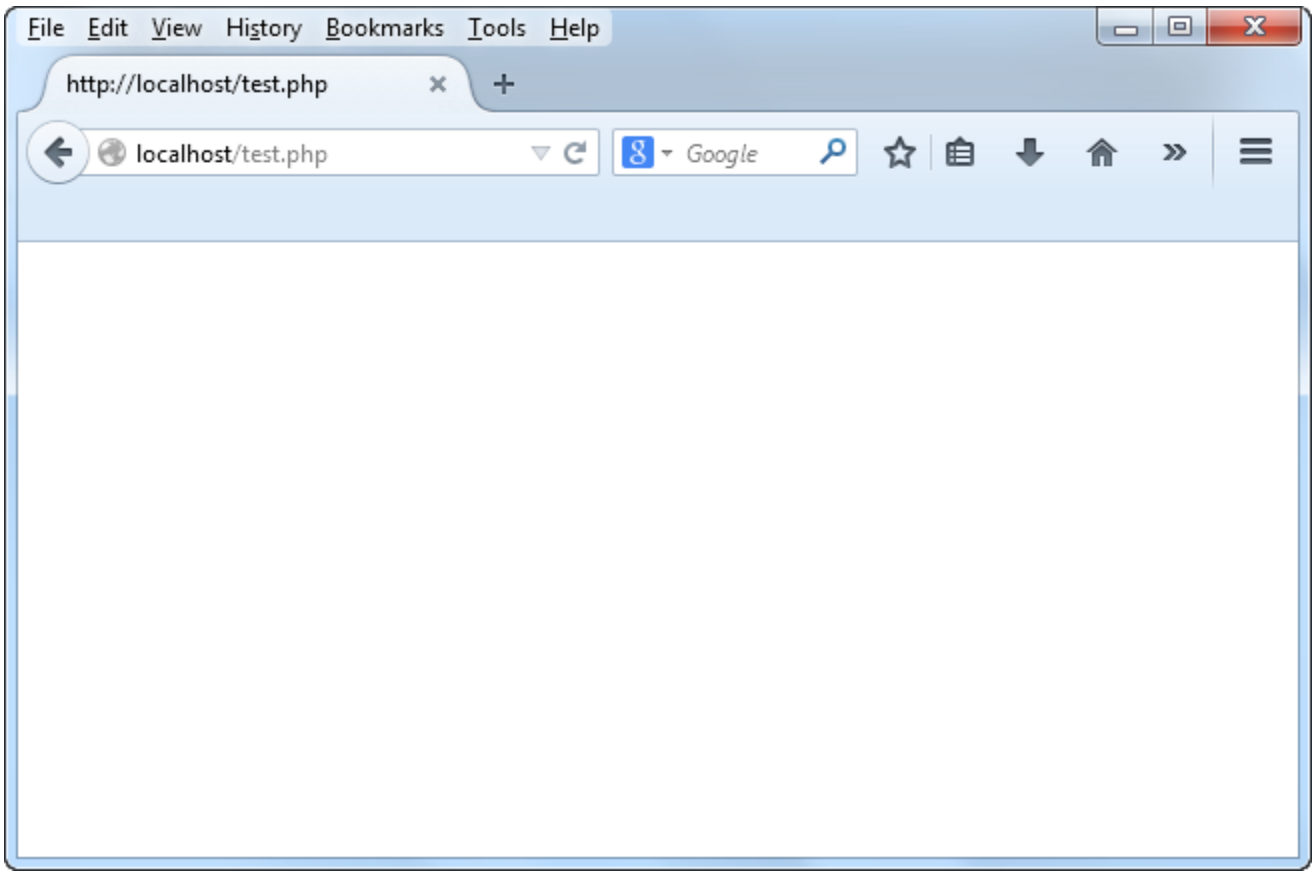


توضیحات

توضیحات در زبان های برنامه نویسی بسیار مفید هستند. آنها در به یاد آوری وظایف کدها به شما کمک می کنند. توضیحات در PHP به سه صورت اعمال می شوند.

```
<?php

// single-line comment

# single-line comment

/* multi-line
   comment */

?>
```

استفاده از توضیحات در برنامه می تواند به شما و دیگران در فهم کدهایتان کمک کند. بدین صورت که در کارهای تیمی کسی که کدهای شما را می بیند با استفاده از توضیحاتی که در مورد کدها داده اید می فهمد که هر کد چه وظیفه ای دارد.دستورات بالا را در داخل فایل test.php نوشته و ذخیره کنید. مشاهده می کنید که با اجرای آن دستورات بالا در مرورگر نمایش داده نمی شوند:



ادغام کدهای HTML و PHP

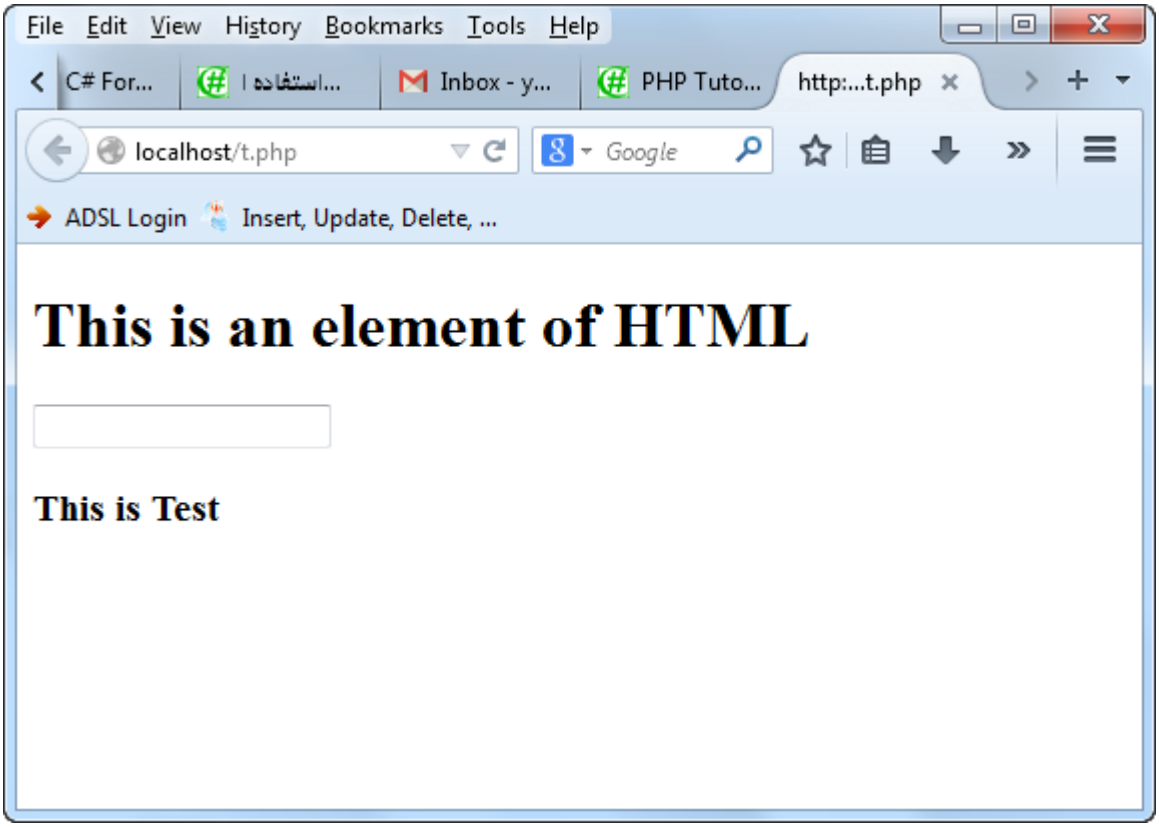
در یک فایل با پسوند .php می توان کدهای HTML و PHP را با هم ادغام کرد. یک مثال می زنیم. فرض کنید که می خواهید یک جعبه متن HTML را به وسیله دستور PHP ایجاد کنید برای این کار از دستور echo به صورت زیر استفاده می شود:

```
<?php
    echo '<h1> This is an element of HTML </H1>';

    echo ' <input type="text"/> ';

    echo '<H3>This is Test</H3>';
?>
```

روش دیگر این است که قبل از ایجاد کد HTML دستور PHP را تمام کنیم:

```
1   <?php
2       echo'<h1> This is an element of HTML </H1>';
3   ?>
4
5
6   <input type="text"/>
7
8
9   <?php
10      echo '<H3>This is Test</H3>';
11  ?>
```

به خط 4 کد اول و به خط 6 کد دوم توجه کنید. درکد اول جعبه متن با استفاده از دستور echo ایجاد شده است، اما در کد دوم قبل از جعبه متن کد PHP را بسته و جعبه متن را ایجاد کرده ایم. خروجی هر دو کد بالا به صورت زیر است:



از کدهای **PHP** در داخل تگ های **HTML** هم می توان استفاده کرد. فرض کنید متن **"Hello World"** را یک بار با و بار دیگر بدون استفاده از **PHP** می خواهیم که در داخل جعبه متن بنویسیم:

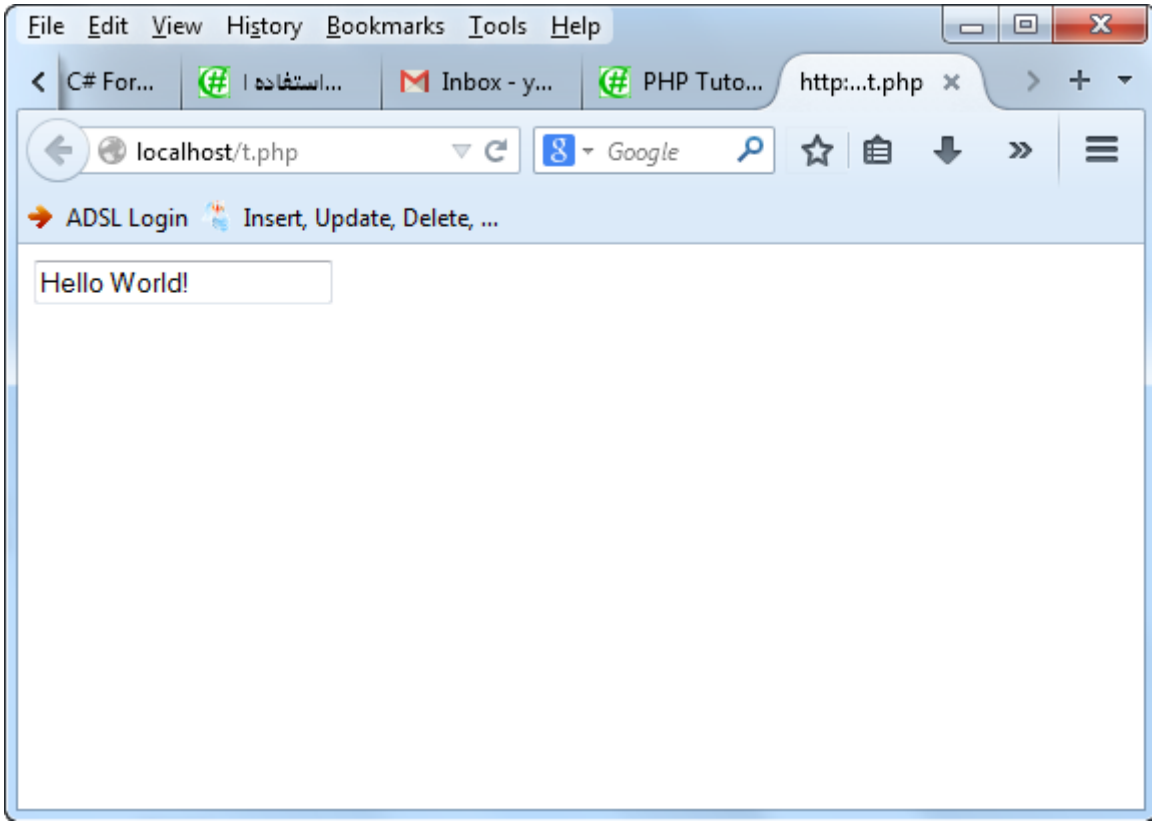
بدون استفاده از دستور PHP

```
<input type="text" value="Hello World!"/>
```

با استفاده از دستور PHP

```
<input type="text" value="<?php echo 'Hello World!'; ?>"/>
```

خروجی هر دو کد بالا به صورت زیر است:


File Edit View History Bookmarks Tools Help
C# For...
استفاده ا...
Inbox - y...
PHP Tuto...
http:...t.php
localhost/t.php
Google
ADSL Login
Insert, Update, Delete, ...
Hello World!

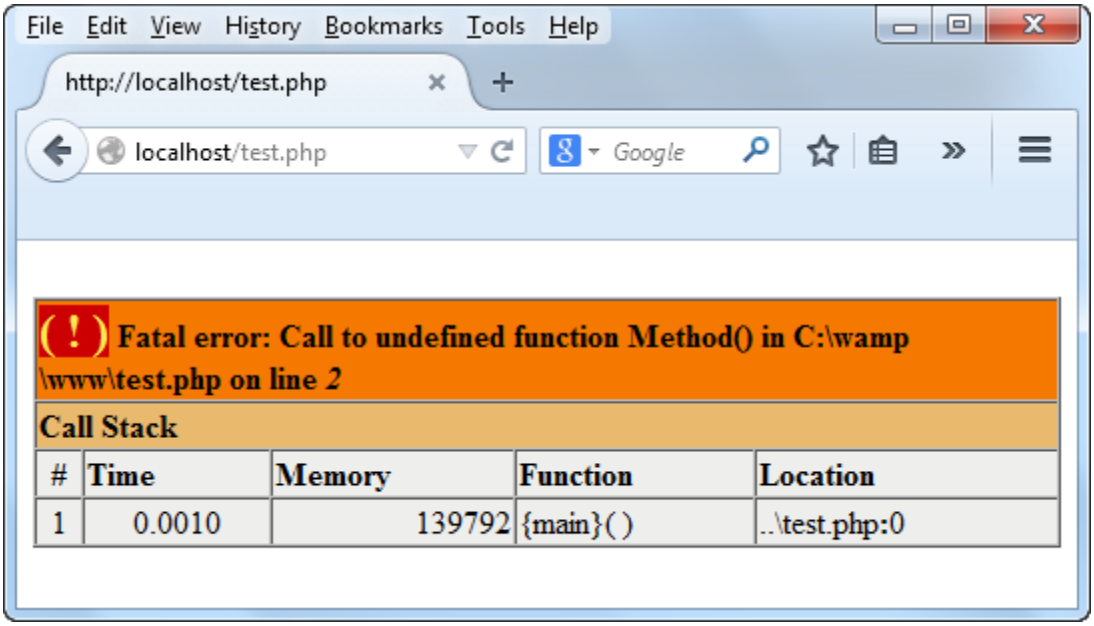
انواع خطاها در PHP

هنگام کدنویسی در PHP ممکن است با خطاهایی مواجه شویم .خطاها ممکن است بر اثر اشتباه تایپی و یا اشتباه در منطق برنامه به وجود بیایند. در جدول زیر لیست خطاهایی که ممکن است در هنگام برنامه نویسی PHP به وجود بیایند آمده است:

خطا	توضیح
Fatal error	این نوع از خطاها که به خطاهای بحرانی هم معروف هستند باعث می شوند که ادامه کار برنامه با مشکل مواجه شده و برنامه اجرا نشود.
Parse error	این نوع خطاها فقط در زمان اجرای برنامه تولید می شوند و اسم دیگر این نوع خطاها Syntax Error می باشد. فراموش کردن یک سمیکالن و یا خطای تایپی باعث به وجود آمدن این خطا ها می شود. این خطاها هم از اجرای برنامه بقیه برنامه جلوگیری می کنند.
Warning	این نوع خطاها توسط PHP به کاربر نمایش داده می شوند، اما مانع از اجرای بقیه برنامه نمی شوند.مثلا وقتی یک عدد رو بر صفر تقسیم می کنیم یک Warning دریافت می کنیم.
Notices	این نوع هم مثل انواع خطاهای قبلی می تواند خودکار توسط خود PHP و یا با استفاده از تابع trigger_error که توسط کاربر ایجاد شده است درست شوند. این نوع خطا بیشتر هشداری است.

مثالی از Fatal Error

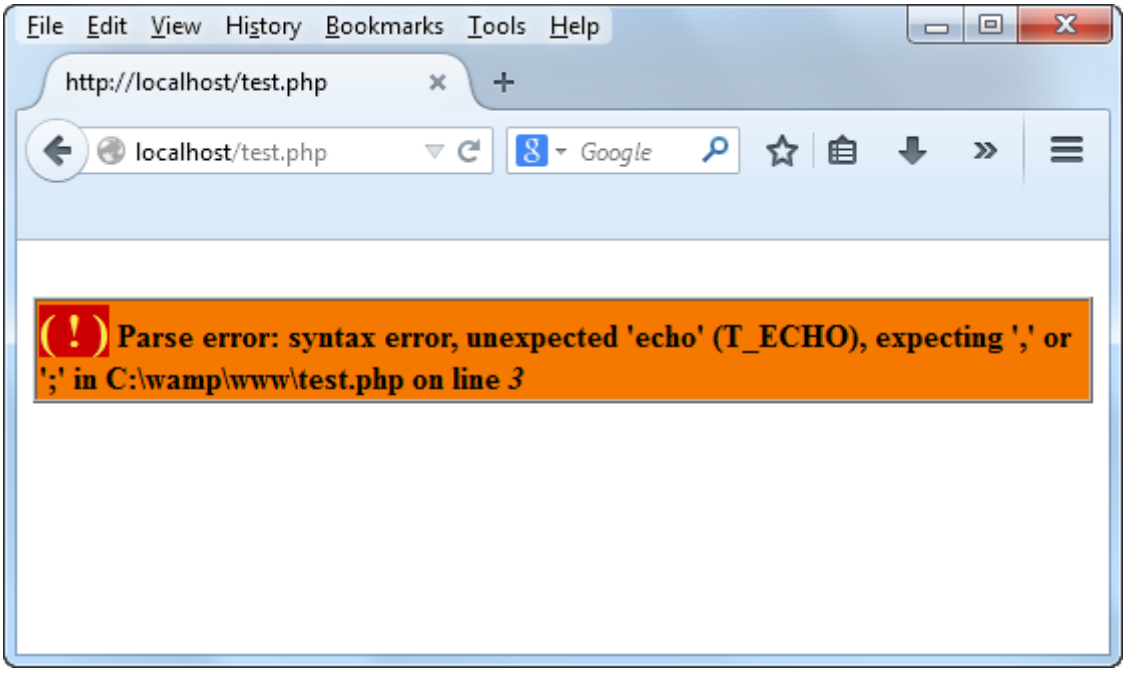
```
<?php
    Method();
    echo "Save Successfully!"
?>
```



پیغام خطای بالا به این دلیل به وجود آمده است، که PHP نتوانسته است تابع Method() را پیدا کند، چون تابع در جایی تعریف نشده است.

مثالی از Parse error

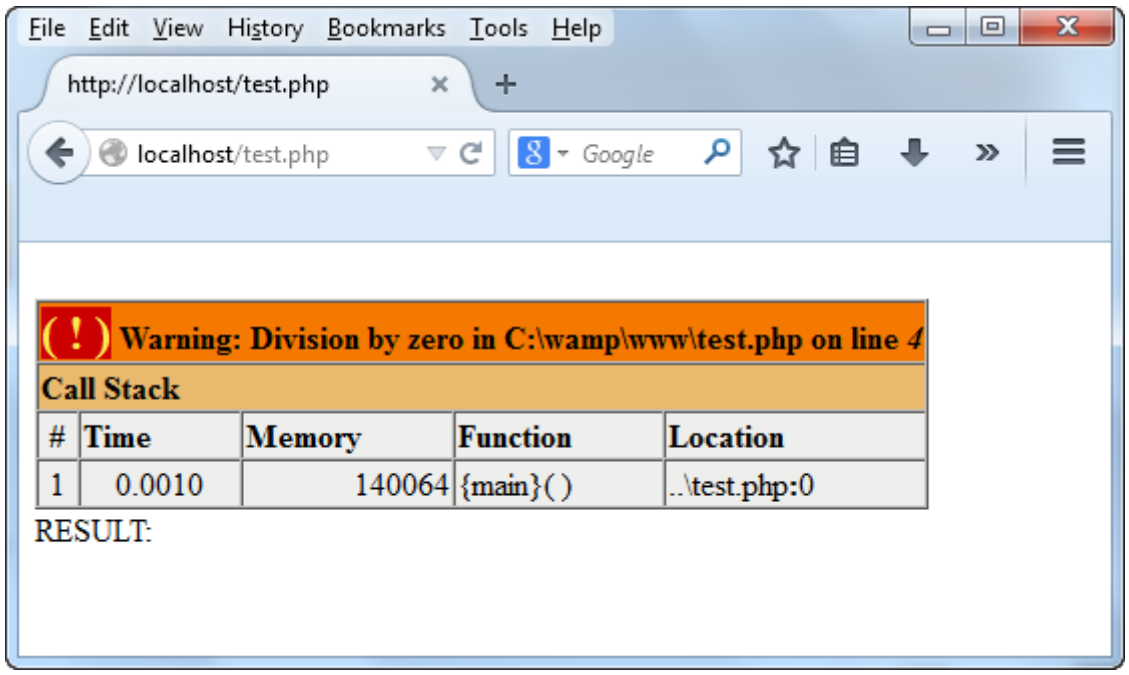
```
<?php
    echo "Save Successfully!"
    echo "PHP Learning";
?>
```



پیغام خطای بالا به این دلیل به وجود آمده است، که در آخر کد اول یا دستور اول علامت سمیکالن (;) قرار داده نشده است.

مثالی از Warning

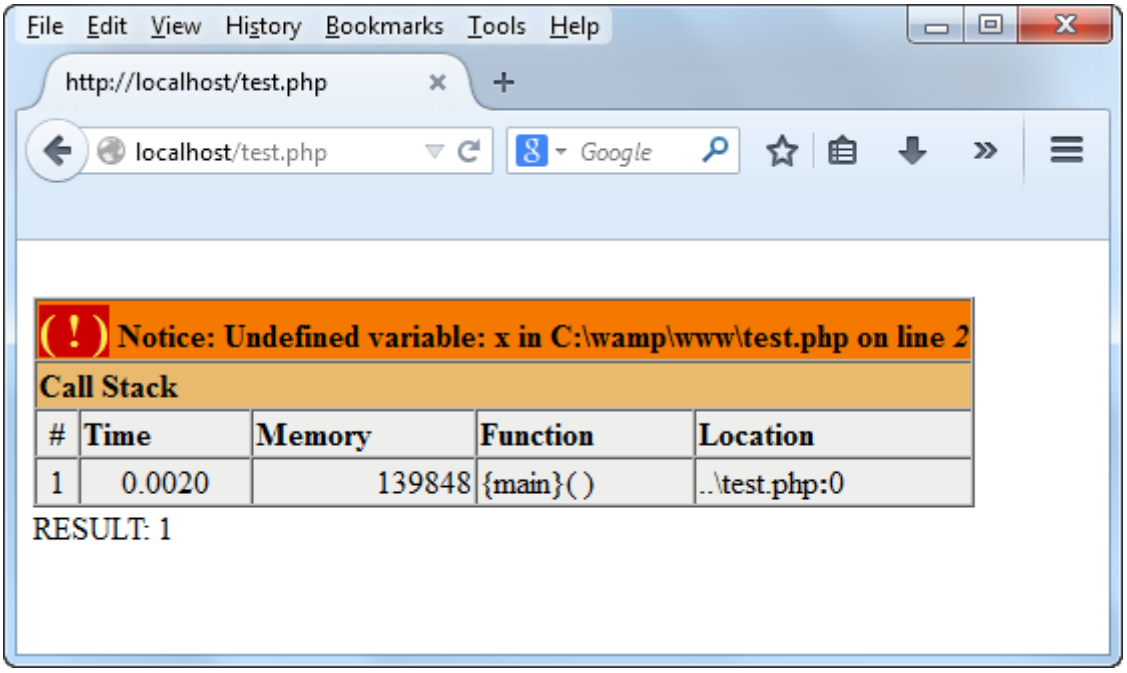
```
<?php
    $x = 200;
    $y = 0;
    $z = $x/$y;
    echo "RESULT: ". $z;
?>
```



همانطور که در شکل بالا مشاهده می کنید، پیغام هشدار نمایش داده شده و بقیه کد هم اجرا شده است.

مثالی از Notices

```
<?php
    $x += 1;
    echo "RESULT: ". $x;
?>
```



همانطور که مشاهده می کنید، برنامه اجرا و یک واحد به متغیر اضافه شده است .در پایان یاد آور می شویم که اگر کدهای بالا برای شما نا مفهوم است نگران نباشید و این بخش صرفا برای آشنایی شما با پیغام خطاهای متداول PHP بود.

کاراکترهای کنترلی

کاراکترهای کنترلی (escape characters) کاراکترهای ترکیبی هستند که با یک بک اسلش (\) شروع می شوند و به دنبال آنها یک حرف یا عدد می آید و یک رشته را با فرمت خاص نمایش می دهند. برای مثال برای ایجاد یک خط جدید و قرار دادن رشته در آن می توان از کاراکتر کنترلی \n استفاده کرد.جدول زیر لیست کاراکترهای کنترلی و کارکرد آنها را نشان می دهد:

Escape	نتیجه
\n	حرکت به سطر بعد
\r	حرکت به ابتدای سطر جاری
\t	کارکتر Tab (معادل 8 کارکترSpace)
\\	کارکتر\
\’	کارکتر ‘ (در رشته های محصور به گیومه تک)
\”	کارکتر ” (در رشته های محصور به گیومه جفت)
\$	کارکتر$
\[0-7]	کارکتری که کد ASCII آن در مبنای 8 در جلوی \ نوشته شده است
\x[0-F]	کارکتری که کد ASCII آن در مبنای 16 در جلوی \x نوشته شده است

ما برای استفاده از کاراکترهای کنترلی از بک اسلش (\) استفاده می کنیم.از آنجاییکه \ معنای خاصی به رشته ها می دهد برای چاپ بک اسلش (\) باید از (\\) استفاده کنیم:

```
<?php
   echo "We can print a \\ by using the \\\\ escape sequence.";
?>
```

```
We can print a \ by using the \\ escape sequence.
```

یکی از موارد استفاده از \\ ،نشان دادن مسیر یک فایل در ویندوز است:

```
<?php
   echo "C:\\Program Files\\Some Directory\\SomeFile.txt";
?>
```

```
C:\Program Files\Some Directory\SomeFile.txt
```

از آنجاییکه از دابل کوتیشن (“) برای نشان دادن رشته ها استفاده می کنیم برای چاپ آن از \”استفاده می کنیم:

```
<?php
   echo "I said, \"Motivate yourself!\".";
?>
```

```
I said, "Motivate yourself!".
```

همچنین برای چاپ کوتیشن (') از \' استفاده می کنیم:

```
<?php
   echo "The programmer\'s heaven.";
?>
```

```
The programmer's heaven.
```

برای ایجاد فاصله بین حروف یا کلمات از t\ استفاده می شود:

```
<?php
   echo "Left\tRight";
?>
```

```
Left  Right
```

برای مشاهده لیست مقادیر مبنای 16 برای کاراکترهای یونیکد به لینک زیر مراجعه نمایید:

http://www.ascii.cl/htmlcodes.htm

اگر کامپایلر به یک کاراکتر کنترلی غیر مجاز برخورد کند، برنامه پیغام خطا می دهد.بیشترین خطا زمانی اتفاق می افتد که برنامه نویس برای چاپ اسلش (\) از \\ استفاده می کند.

متغیر

متغیر مکانی از حافظه است که شما می توانید مقادیری را در آن ذخیره کنید.می توان آن را به عنوان یک ظرف تصور کرد که داده های خود را در آن قرار داده اید.محتویات این ظرف می تواند پاک شود یا تغییر کند.هر متغیر دارای یک نام نیز هست.که از طریق آن میتوان متغیر را از دیگر متغیر ها تشخیص داد و به مقدار آن دسترسی پیدا کرد.همچنین دارای یک مقدار می باشد که می تواند توسط کاربر انتخاب شده باشد یا نتیجه یک محاسبه باشد.مقدار متغیر می تواند تهی نیز باشد.متغیر دارای نوع نیز هست بدین معنی که نوع آن با نوع داده ای که در آن ذخیره می شود یکی است. متغیر دارای عمر نیز هست که از روی آن می توان تشخیص داد که متغیر باید چقدر در طول برنامه مورد استفاده قرار گیرد.و در نهایت متغیر دارای محدوده استفاده نیز هست که به شما می گوید که متغیر در چه جای برنامه برای شما قابل دسترسی است.ما از متغیرها به عنوان یک انبار موقتی برای ذخیره داده استفاده می کنیم. هنگامی که یک برنامه ایجاد می کنیم احتیاج به یک مکان برای ذخیره داده ،مقادیر یا داده هایی که توسط کاربر وارد می شوند داریم.ایم مکان همان متغیر است. برای این از کلمه متغیر استفاده می شود چون ما می توانیم بسته به نوع شرایط هر جا که لازم باشد مقدار آن را تغییر دهیم. متغیرها موقتی هستند و فقط موقعی مورد استفاده قرار می گیرند که برنامه در حال اجراست و وقتی شما برنامه را می بندید محتویات متغیر ها نیز پاک می شود. قبلا ذکر شد که به وسیله نام متغیر می توان به آن دسترسی پیدا کرد. برای نامگذاری متغیرها باید قوانین زیر را رعایت کرد:

تعریف متغیرها

برای تعریف یک متغیر ابتدا علامت ($) و سپس نام را می نویسیم. یکی از روش های نامگذاری متغیرها این است که اگر نام متغیر چند بخش یا چند کلمه ای باشد، بجز کلمه اول، حرف اول سایر کلمات با حروف بزرگ شروع می شود:

```
$myVar;
```

مرحله بعد از نامگذاری، مقداردهی به متغیر است، برای اختصاص مقدار به یک متغیر از علامت مساوی استفاده می شود:

```
$myVar = 10;
```

وقتی که متغیر تعریف و مقدار دهی شد، می توان با استفاده از نام آن به مقدار آن دسترسی یافت. مثلا با استفاده از دستور echo و نام متغیر بالا می توان مقدار آن را چاپ کرد:

```
echo $myVar; // "10"
```

نکاتی در باره نامگذاری متغیرها وجود دارد که در زیر به آنها اشاره شده است:

- **نام متغیرها به حروف بزرگ و کوچک حساس است. یعنی متغیری با نام Myvar با متغیر myVar فرق دارد.**
- **نام متغیر می تواند شامل علامت زیر خط و عدد باشد ولی نمی تواند با عدد شروع شود.**

- **نام متغیر نمی تواند دارای فضای خالی و یا کلماتی باشد که برای PHP دارای معنی خاصی هستند.**

انواع داده

PHPیک زبان با نوع داده ای ضعیف است. بدین معنی که برای تعریف متغیر لازم نیست که نوع مقداری که باید قبول کند را تعیین کنید. بلکه متغیر به طور خودکار به نوعی تبدیل می شود که به آن اختصاص داده شده است .به مثال زیر توجه کنید:

```
$myVar = 1; // int type
$myVar = 1.5; // float type
```

در کد بالا متغیر $myVar در خط اول به نوع int و در خط دوم به نوع float تبدیل شده است. چون مقادیری که به ان اختصاص داده شده است از نوع اعداد صحیح و اعداد اعشاری می باشند. پس نوع متغیر از همان نوعی است که به آن اختصاص داده شده است. در جدول زیر نه نوع داده ای که در PHP وجود دارند، ذکر شده است:

نوع	توضیح
int	اعداد بدون ممیز، صحیح هستند.
float	اعداد با ممیز شناور، اعدادی هستند که شامل یک ممیز هستند یا اعدادی که در قالب نماد ریاضی نشان داده می شوند.
bool	مقدار داده های Boolean می تواند TRUE یا FALSE باشد.
string	رشته، یک توالی از کاراکترهاست
array	آرایه ها انواع خاصی از متغیرها به حساب می آیند که می توانند چندین داده را در قالب یک نام ذخیره کنند.
object	یک شیء نوع داده ای است که هم داده ها و هم اطلاعات مربوط به نحوه پردازش آنها را ذخیره می کند.
resource	مرجعی به یک منبع خارجی مثل DBمی باشد.
callable	متدها و توابع
null	با استفاده از مقدارNULL ، می توان نشان داد که یک متغییر مقدار ندارد

نوع صحیح (Integer type)

عدد صحیح یک عدد کامل است. عدد صحیح می تواند در مبنای 10 ، در مبنای 16، در مبنای 8 و یا در مبنای 2 ، نمایش داده شود. اعداد در مبنای 16 با "0x"، اعداد در مبنای 8 با "0" و اعداد در مبنای 2 با "0 b" شروع می شوند.

```
$myInt = 1234; // decimal number
$myInt = 0b10; // binary number (2 decimal)
$myInt = 0123; // octal number (83 decimal)
$myInt = 0x1A; // hexadecimal number (26 decimal)
```

اعداد صحیح در PHP شامل اعداد منفی و مثبت می باشد.

نوع با ممیز اعشاری(Floating-point type)

در php اعداد اعشاری را میتوان به روشهای متفاوتی نوشت. اندازه ی این نوع داده نیز به نوع سیستم عامل و پردازنده بستگی دارد ولی حداکثر عدد ۱,۸ e308 با ۱۴ رقم اعشار را می توان در نظر گرفت.

```
$myFloat = 1.234;
$myFloat = 3e2; // 3*10^2 = 300
```

نوع بولی(Bool type)

ساده ترین نوع داده درPHP ، داده های Boolean می باشند که تنها دو مقدار True و False را دریافت می کنند.

```
$myBool = true;
```

نوع تهی(Null type)

این نوع داده که از نوع مخصوص می باشد تنها یک مقدار را دریافت می کند و آن هم مقدار NULL است. NULL مشخص میکند که یک متغیر دارای مقدار نمی باشد.

```
$myNull = null; // variable is set to null
```

نکاتی در مورد کار با انواع داده درPHP

- **PHPنوع داده را بصورت اتوماتیک معین می کند. وقتی اسکریپت های PHP را می نویسید احتیاجی به معین کردن نوع داده ای که ذخیره می کنید، ندارید .مثال های زیر دو نوع داده مختلف درست می کنند.**

```
$myVar1 = 123;
$myVar2 = "123";
```

- **مقدار برای $myVar به عنوان integer ذخیره می شود و مقدار متغیر دوم به عنوان string و رشته ای، برای اینکه مقدار این متغیر در داخل کوتیشن قرار گرفته است.**

- **PHP وقتی احتیاج باشد، انواه داده ها را بصورت اتوماتیک تبدیل می کند .برای مثال اگر دو متغیر را جمع کنید و متغیر اول شامل عدد صحیح و دیگری اعشاری باشد، PHP بصورت خودکار integer را به float تبدیل می کند و بدین صورت قابل جمع با متغیر دوم می شود.**

درباره سایر انواع داده ای در درس های آینده توضیح می دهیم.

تست نوع متغیر

برای تشخیص نوع داده ای که در یک متغیر ذخیره شده است از تابع gettype() استفاده می شود. برای این کار کافیست که نام متغیر را در داخل پرانتز تابع قرار دهید تا نوع آن را به شما نمایش دهد. به مثال زیر توجه کنید:

```
<?php
    $x = 10.2;
    echo gettype($x);
?>
```

```
double
```

همانطور که در مثال بالا مشاهده می کنید خروجی کلمه double هست که نشان دهنده نوع داده ای است که در متغیر $x ذخیره شده است. در php توابع دیگری برای تشخیص نوع داده ذخیره شده در متغیر وجود دارد که در جدول زیر لیست آنها آمده است:

تابع	عملکرد
is_int(value)	اگر value از نوع صحیح باشد مقدار 1 یا true را بر می گرداند
is_ string (value)	اگر value از نوع رشته باشد مقدار 1 یا true را بر می گرداند
is_float(value)	اگر value از نوع اعشار باشد مقدار 1 یا true را بر می گرداند
is_ bool (value)	اگر value از نوع boolean باشد مقدار 1 یا true را بر می گرداند
is_ array (value)	اگر value از نوع آرایه باشد مقدار 1 یا true را بر می گرداند
is_ object (value)	اگر value از نوع object باشد مقدار 1 یا true را بر می گرداند
is_ resource (value)	اگر value یک منبع داده باشد مقدار 1 یا true را بر می گرداند
is_ null (value)	اگر value از نوع null باشد مقدار 1 یا true را بر می گرداند

به مثال زیر توجه کنید:

```
<?php
    $name = "jack";
    echo is_string($name);
?>
```

```
1
```

چون متغیر $name از نوع رشته است در نتیجه تابع is_string() مقدار 1 را بر می گرداند.

تغییر نوع متغیر

فرض کنید که می خواهید مقدار یک متغیر را از نوع صحیح به نوع اعشار تغییر دهید. این کار در php توسط تابع settype() انجام می شود .به مثال زیر توجه کنید:

```
<?php
    $number = 10;
    settype($number,"double");
?>
```

همانطور که در مثال بالا مشاهده می کنید، در داخل پرانتز این تابع ابتدا نام متغیر و سپس در داخل یک جفت کوتیشن نام نوعی که قرار است متغیر به آن تبدیل شود را می نویسیم. در مثال بالا یک نوع صحیح به یک نوع double تبدیل شده است. روش دیگری که به آن عمل cast می گویند هم برای انجام این کار وجود دارد. در عمل cast نام نوعی که قرار است متغیر به آن تبدیل شود را در داخل پرانتز و قبل از نام متغیر می نویسیم. مثلا در مثال بالا:

```
<?php
    $number = 10;
    echo gettype($number);

    echo '<br/>';

    $newType = (double)$number;
    echo gettype($newType);
?>
```

```
integer
double
```

لیست تبدیل های صریح در جدول زیر آمده است:

تابع	عملکرد
(int) value or (integer) value	valueرا به نوع صحیح تبدیل می کند
(float) value	valueرا به نوع اعشار تبدیل می کند
(string) value	valueرا به نوع رشته تبدیل می کند
(bool) value or (boolean) value	valueرا به نوع بولی تبدیل می کند
(array) value	valueرا به نوع آرایه تبدیل می کند
(object) value	valueرا به نوع object تبدیل می کند

توابع دیگری در php برای تبدیل انواع متغیرها به هم وجود دارد که در جدول زیر لیست آنها آمده است:

تابع	عملکرد
intval(value)	valueرا به نوع صحیح تبدیل می کند
floatval(value)	valueرا به نوع اعشار تبدیل می کند
strval(value)	valueرا به نوع رشته تبدیل می کند

ثابت ها

ثابت ها انواعی از متغیرها هستند که مقدار آنها در طول برنامه تغییر نمی کند. ثابت ها حتما باید مقدار دهی اولیه شوند و اگر مقدار دهی آنها فراموش شود در برنامه خطا به وجود می آید. بعد از این که به ثابت ها مقدار اولیه اختصاص دادهشد هرگز در زمان اجرای برنامه نمی توان آن را تغییر داد.

تعریف ثابت با کلمه کلیدی const و تابع define

برای تعریف ثابت ها باید از کلمه کلیدی **const** و یا تابع **define** استفاده کرد.معمولا نام ثابت ها را طبق قرارداد با حروف بزرگ می نویسند تا تشخیص آنها در برنامه راحت باشد. نحوه تعریف ثابت به دو روش بالا در زیر آمده است :

```
const CONSTNAME= initial_value;
```

یا

```
define('CONSTNAME', initial_value);
```

مثال:

```
<?php

    const NUMBER = 10;   //with const keyword

    define('PI', 3.14); // with define function

    echo NUMBER;
    echo '<br/>';
    echo PI;

?>
```

```
10
3.14
```

همانطور که در مثال بالا مشاهده می کنید قبل از نام ثابت ها نباید از علامت $ و برای فراخوانی آنها در دستور **echo** نمی بایست از علامت " " استفاده کنیم. مقدار دادن به یک ثابت ،که قبلا مقدار دهی شده برنامه را با خطا مواجه می کند:

```
<?php

```

```
    const NUMBER = 10;

    NUMBER = 12;

    echo NUMBER;

?>
(!) Parse error: syntax error, unexpected '='
```

نکته ی دیگری که نباید فراموش شود این است که نباید مقدار ثابت را با مقدار دیگر متغیرهای تعریف شده در برنامه برابر قرار داد. مثال:

```
<?php

    $variable=12;
    const NUMBER = 10;

    NUMBER = $variable;

    echo NUMBER;

?>
(!) Parse error: syntax error, unexpected '='
```

ممکن است این سوال برایتان پیش آمده باشد که دلیل استفاده از ثابت ها چیست؟ اگر مطمئن هستید که مقادیری در برنامه وجود دارند که هرگز در طول برنامه تغییر نمی کنند بهتر است که آنها را به صورت ثابت تعریف کنید.این کار هر چند کوچک کیفیت برنامه شما را بالا می برد.

عبارات و عملگرها

ابتدا با دو کلمه آشنا شوید:

- **عملگر : نمادهایی هستند که اعمال خاص انجام می دهند.**
- **عملوند : مقادیری که عملگرها بر روی آنها عملی انجام می دهند.**

مثلا X+Y: یک عبارت است که در آن X و Y عملوند و علامت + عملگر به حساب می آیند.

زبانهای برنامه نویسی جدید دارای عملگرهایی هستند که از اجزاء معمول زبان به حساب می آیند PHP .دارای عملگرهای مختلفی از جمله عملگرهای ریاضی ،تخصیصی ، مقایسه ای ، منطقی و بیتی می باشد. از عملگرهای ساده ریاضی می توان به عملگر جمع و تفریق اشاره کرد. سه نوع عملگر در PHP وجود دارد:

- **یگانی – (Unary) به یک عملوند نیاز دارد**
- **دودویی – (Binary) به دو عملوند نیاز دارد**
- **سه تایی – (Ternary) به سه عملوند نیاز دارد**

انواع مختلف عملگر که در ای بخش مورد بحث قرار می گیرند عبارتند از:

- **عملگرهای ریاضی**
- **عملگرهای تخصیصی**
- **عملگرهای مقایسه ای**
- **عملگرهای منطقی**
- **عملگرهای بیتی**
- **عملگر رشته**

عملگرهای ریاضی

PHP از عملگرهای ریاضی برای انجام محاسبات استفاده می کند. جدول زیر عملگرهای ریاضی PHP را نشان می دهد:

عملگر	دسته	مثال	نتیجه
+	Binary	var1 = var2 + var3;	Var1برابر است با حاصل جمع var2 وvar3
-	Binary	var1 = var2 – var3;	Var1برابر است با حاصل تفریق var2 وvar3
*	Binary	var1 = var2 * var3;	Var1برابر است با حاصلضرب var2 درvar3
/	Binary	var1 = var2 / var3;	Var1برابر است با حاصل تقسیم var2 برvar3
%	Binary	var1 = var2 % var3;	Var1برابر است با باقیمانده تقسیم var2 وvar3
+	Unary	var1 = +var2;	Var1برابر است با مقدارvar2
-	Unary	var1 = -var2;	Var1برابر است با مقدار var2 ضربدر -1

مثال بالا در از نوع عددی استفاده شده است.اما استفاده از عملگرهای ریاضی برای نوع رشته ای نتیجه متفاوتی دارد.همچنین در جمع دو کاراکتر کامپایلر معادل عددی آنها را نشان می دهد. اگر از عملگر + برای رشته ها استفاده کنیم دو رشته را با هم ترکیب کرده و به هم می چسباند. دیگر عملگرهای PHP عملگرهای کاهش و افزایش هستند.این عملگرها مقدار 1 را از متغیر ها کم یا به آنها اضافه می کنند. از این متغیر ها اغلب در حلقه ها استفاده می شود:

عملگر	دسته	مثال	نتیجه
++	Unary	var1 = ++var2;	مقدار var1 برابر است با var2 بعلاوه 1
- -	Unary	var1 = – -var2;	مقدار var1 برابر است با var2 منهای 1
++	Unary	var1 = var2++;	مقدار var1 برابر است باvar2 به متغیر var2 یک واحد اضافه می شود
- -	Unary	var1 = var2- -;	مقدار var1 برابر است باvar2 از متغیر var2 یک واحد کم می شود

به این نکته توجه داشته باشید که محل قرار گیری عملگر در نتیجه محاسبات تاثیر دارد.اگر عملگر قبل از متغیر var2 بیاید افزایش یا کاهش var1 اتفاق می افتد.چنانچه عملگرها بعد از متغیر var2 قرار بگیرند ابتدا var1 برابر var2 می شود و سپس متغیر var2 افزایش یا کاهش می یابد. به مثال های زیر توجه کنید:

```
<?php
    $x = 0;
    $y = 1;

    $x = ++$y;

    echo('x = ' . $x).'<br/>';
```

```
    echo('y = ' . $y);
?>
```

```
x=2
y=2
```

```
<?php
    $x = 0;
    $y = 1;

    $x = --$y;

    echo('x = ' . $x).'<br/>';
    echo('y = ' . $y);
?>
```

```
x=0
y=0
```

همانطور که در دو مثال بالا مشاهده می کنید،درج عملگرهای — و ++ قبل از عملوند y باعث می شود که ابتدا یک واحد از y کم و یا یک واحد به y اضافه شود و سپس نتیجه در عملوند x قرار بگیرد. حال به دو مثال زیر توجه کنید:

```
<?php
    $x = 0;
    $y = 1;

    $x = $y--;

    echo('x = ' . $x).'<br/>';
    echo('y = ' . $y);
?>
```

```
x=1
y=0
```

```
<?php
    $x = 0;
    $y = 1;

    $x = $y++;

    echo('x = ' . $x).'<br/>';
    echo('y = ' . $y);
?>
```

```
x=1
y=2
```

همانطور که در دو مثال بالا مشاهده می کنید،درج عملگرهای — و ++ بعد از عملوند y باعث می شود که ابتدا مقدار y در داخل متغیر x قرار بگیرد و سپس یک واحد از y کم و یا یک واحد به ان اضافه شود.

عملگرهای تخصیصی

نوع دیگر از عملگرهای PHP عملگرهای جایگزینی نام دارند.این عملگرها مقدار متغیر سمت راست خود را در متغیر سمت چپ قرار می دهند. جدول زیر انواع عملگرهای تخصیصی در PHP را نشان می دهد:

عملگر	مثال	نتیجه
=	var1 = var2;	مقدار var1 برابر است با مقدارvar2
+=	var1 += var2;	مقدار var1 برابر است با حاصل جمع var1 وvar2
-=	var1 -= var2;	مقدار var1 برابر است با حاصل تفریق var1 وvar2
*=	var1 *= var2;	مقدار var1 برابر است با حاصل ضرب var1 درvar2
/=	var1 /= var2;	مقدار var1 برابر است با حاصل تقسیم var1 برvar2
%=	var1 %= var2;	مقدار var1 برابر است با باقیمانده تقسیم var1 برvar2

استفاده از این نوع عملگرها در واقع یک نوع خلاصه نویسی در کد است. مثلا دو کد زیر معادل هم هستند:

```
var1 += var2
var1 = var1 + var2
```

این حالت کدنویسی زمانی کارایی خود را نشان می دهد که نام متغیرها طولانی باشد. برنامه زیر چگونگی استفاده از عملگرهای تخصیصی و تاثیر آنها را بر متغیر ها نشان می دهد.

```
<?php
    $number = 10;

    echo 'Number = '. $number . '<br/>';

    $number += 10;
    echo 'Number = '. $number . '<br/>';

    $number -= 10;
    echo 'Number = '. $number . '<br/>';
?>
```

```
Number = 10
Number = 20
Number = 10
```

در برنامه از 3 عملگر تخصیصی استفاده شده است. ابتدا یک متغیر و مقدار 10 با استفاده از عملگر = به آن اختصاص داده شده است.سپس به آن با استفاده از عملگر += مقدار 10 اضافه شده است.و در آخر به وسیله عملگر -= عدد 10 از آن کم شده است.

عملگرهای مقایسه ای

از عملگرهای مقایسه ای برای مقایسه مقادیر استفاده می شود.نتیجه این مقادیر یک مقدار بولی(منطقی) است.این عملگرها اگر نتیجه مقایسه دو مقدار درست باشد مقدار یک یا true و اگر نتیجه مقایسه اشتباه باشد مقدار تهی یا false را نشان می دهند. این عملگرها به طور معمول در دستورات شرطی به کار می روند به این ترتیب که باعث ادامه یا توقف دستور شرطی می شوند.جدول زیر عملگرهای مقایسه ای در PHP را نشان می دهد:

عملگر	دسته	مثال	نتیجه
==	Binary	var1 = var2 == var3	var1 در صورتی true است که مقدار var2 با مقدار var3 برابر باشد در غیر اینصورت false است
!=	Binary	var1 = var2 != var3	var1 در صورتی true است که مقدار var2 با مقدار var3 برابر نباشد در غیر اینصورت false است
===	Binary	var1 = var2=== var3	var1 در صورتی true است که var2 با var3 هم از لحاظ مقدار و هم از لحاظ نوع یکی باشند، در غیر اینصورت false است
!==	Binary	var1 = var2 !== var3	var1 در صورتی true است که var2 با var3 از لحاظ مقدار و نوع یکی نباشند، در غیر اینصورت false است
<	Binary	var1 = var2 < var3	var1 در صورتی true است که مقدار var2 کوچکتر از var3 مقدار باشد در غیر اینصورت false است
>	Binary	var1 = var2 > var3	var1 در صورتی true است که مقدارvar2 بزرگتر ازمقدار var3باشد در غیر اینصورت false است
<=	Binary	var1 = var2 <= var3	var1 در صورتی true است که مقدار var2 کوچکتر یا مساوی مقدار var3 باشد در غیر اینصورت false است
>=	Binary	var1 = var2 >= var3	var1 در صورتی true است که مقدار var2 بزرگتر یا مساوی var3 مقدار باشد در غیر اینصورت false است

برنامه زیر نحوه عملکرد ای عملگرها را نشان می دهد:

```
<?php
    $firstnumber  = 10;
    $secondnumber = 5;

    echo ($firstnumber  == $secondnumber)  . '<br/>';
    echo ($firstnumber  != $secondnumber)  . '<br/>';
    echo ($firstnumber === $secondnumber)  . '<br/>';
    echo ($firstnumber !== $secondnumber)  . '<br/>';
    echo ($firstnumber   < $secondnumber)  . '<br/>';
    echo ($firstnumber   > $secondnumber)  . '<br/>';
```

```
    echo ($firstnumber  <= $secondnumber)  . '<br/>';
    echo ($firstnumber  >= $secondnumber)  . '<br/>';
?>
```

```
1

1

1

1
```

درمثال بالا ابتدا دو متغیر را که می خواهیم با هم مقایسه کنیم را ایجاد کرده و به آنها مقادیری اختصاص می دهیم.سپس با استفاده از یک عملگر مقایسه ای آنها را با هم مقایسه کرده و نتیجه را چاپ می کنیم.به این نکته توجه کنید که هنگام مقایسه دو متغیر باید از عملگر == به جای عملگر = استفاده شود. عملگر = عملگر تخصیصی است و در عبارتی مانند x = y مقدار y را در به x اختصاص می دهد. عملگر == عملگر مقایسه ای است که دو مقدار را با هم مقایسه می کند مانند x==y و اینطور خوانده می شود x برابر است با. y

عملگرهای منطقی

عملگرهای منطقی بر روی عبارات منطقی عمل می کنند و نتیجه آنها نیز یک مقدار بولی است. از این عملگرها اغلب برای شرطهای پیچیده استفاده می شود. همانطور که قبلا یاد گرفتید مقادیر بولی می توانند false یا true باشند. فرض کنید که var2و var3 دو مقدار بولی هستند.

عملگر	مثال	توضیحات
&&	var1 = var2 && var3;	در صورتی مقدار var1 برابر true است که هم var2 و هم var3 مقدارشان true باشد
and	var1 = var2 and var3;	در صورتی مقدار var1 برابر true است که هم var2 و هم var3 مقدارشان true باشد
||	var1 = var2 || var3;	در صورتی مقدار var1 برابر true است که var2 یا var3 مقدارشان true باشد
or	var1 = var2 or var3;	در صورتی مقدار var1 برابر true است که var2 یا var3 مقدارشان true باشد
xor	var1 = var2 xor var3;	در صورتی مقدار var1 برابر true است که var2 یا var3 (نه هر دو) مقدارشان true باشد
!	var1 = !var1;	در صورتی مقدار var1 برابر true است که var1 سمت راست مقدارش false باشد

عملگر منطقی and یا (&&)

اگر مقادیر دو طرف عملگر and ، trueباشند عملگر and مقدار true را بر می گرداند. در غیر اینصورت اگر یکی از مقادیر یا هر دوی آنها false باشند مقدار false را بر می گرداند. در زیر جدول درستی عملگر and نشان داده شده است:

X && Y	Y	X
true	true	true
false	false	true
false	true	false
false	false	false

برای درک بهتر تاثیر عملگر and یاد آوری می کنم که این عملگر فقط در صورتی مقدار true را نشان می دهد که هر دو عملوند مقدارشان true باشد.در غیر اینصورت نتیجه تمام ترکیبهای بعدی false خواهد شد. استفاده از عملگر and مانند استفاده از عملگرهای مقایسه ای است. به عنوان مثال نتیجه عبارت زیر درست (true) است اگر سن (age) بزرگتر از 18 و salary کوچکتر از 1000 باشد.

```
result = (age > 18) && (salary < 1000);
```

عملگر and زمانی کارامد است که ما با محدود خاصی از اعداد سرو کار داریم. مثلا عبارت 10 <= x <= 100بدین معنی است که x می تواند مقداری شامل اعداد 10 تا 100 را بگیرد. حال برای انتخاب اعداد خارج از این محدوده می توان از عملگر منطقی and به صورت زیر استفاده کرد.

```
inRange = (number <= 10) && (number >= 100);
```

عملگر منطقی or یا (||)

اگر یکی یا هر دو مقدار دو طرف عملگر or ،درست (true) باشد،عملگر or مقدار true را بر می گرداند. جدول درستی عملگر or در زیر نشان داده شده است :

X	Y	X || Y
true	true	true
true	false	true
false	true	true
false	false	false

در جدول بالا مشاهده می کنید که عملگر or در صورتی مقدار false را بر میگرداند که مقادیر دو طرف آن false باشند. کد زیر را در نظر بگیرید.نتیجه این کد در صورتی درست (true) است که رتبه نهایی دانش آموز (finalGrade) بزرگتر از 75 یا یا نمره نهایی امتحان آن 100 باشد.

```
isPassed = (finalGrade >= 75) || (finalExam == 100);
```

عملگر منطقی not یا (!)

برخلاف دو اپراتور or و and عملگر منطقی not یک عملگر یگانی است و فقط به یک عملوند نیاز دارد. این عملگر یک مقدار یا اصطلاح بولی را نفی می کند .مثلا اگر عبارت یا مقدار true باشد آنرا false و اگر false باشد آنرا true می کند. جدول زیر عملکرد اپراتور not را نشان می دهد:

X	!X
true	false
false	true

نتیجه کد زیر در صورتی درست است که age (سن) بزرگتر یا مساوی 18 نباشد.

```
isMinor = !(age >= 18);
```

عملگرهای بیتی

عملگرهای بیتی به شما اجازه می دهند که شکل باینری انواع داده ها را دستکاری کنید.برای درک بهتر این درس توصیه می شود که شما سیستم باینری و نحوه تبدیل اعداد دهدهی به باینری را یاد بگیرید.در سیستم باینری (دودویی) که کامپیوتر از آن استفاده می کند وضعیت هر چیز یا خاموش است یا روشن.برای نشان دادن حالت روشن از عدد 1 و برای نشان دادن حالت خاموش از عدد 0 استفاده می شود.بنابراین اعداد باینری فقط می توانند صفر یا یک باشند.اعداد باینری را اعداد در مبنای 2 و اعداد اعشاری را اعداد در مبنای 10 می گویند. یک بیت نشان دهنده یک رقم باینری است و هر بایت نشان دهنده 8 بیت است. به عنوان مثال برای یک داده از نوع int به 32 بیت یا 8 بایت فضا برای ذخیره آن نیاز داریم،این بدین معناست که اعداد از 32 رقم 0 و 1 برای ذخیره استفاده می کنند. برای مثال عدد 100 وقتی به عنوان یک متغیر از نوع int ذخیره می شود در کامپیوتر به صورت زیر خوانده می شود:

```
00000000000000000000000001100100
```

عدد 100 در مبنای ده معادل عدد 1100100 در مبنای 2 است.در اینجا 7 رقم سمت راست نشان دهنده عدد 100 در مبنای 2 است و مابقی صفرهای سمت راست برای پر کردن بیتهایی است که عدد از نوع int نیاز دارد.به این نکته توجه کنید که اعداد باینری از سمت راست به چپ خوانده می شوند. عملگرهای بیتی PHP در جدول زیر نشان داده شده اند:

مثال	دسته	نام	عملگر
x = y & z;	Binary	بیتیAND	&
x = y | z;	Binary	بیتیOR	|
x = y ^ z;	Binary	بیتیXOR	^
x = ~y;	Unary	بیتیNOT	~

عملگر بیتی AND(&)

عملگر بیتی AND کاری شبیه عملگر منطقی AND انجام می دهد با این تفاوت که این عملگر بر روی بیتها کار می کند.اگر مقادیر دو طرف آن 1 باشد مقدار 1 را بر می گرداند و اگر یکی یا هر دو طرف آن صفر باشد مقدار صفر را بر می گرداند. جدول درستی عمگر بیتی AND در زیر آمده است:

X AND Y	Y	X
1	1	1
0	0	1
0	1	0
0	0	0

در زیر نحوه استفاده از عملگر بیتی AND آمده است:

```
$result = 5 & 3;

echo $result;
```

```
1
```

همانطور که در مثال بالا مشاهده می کنید نتیجه عملکرد عملگر AND بر روی دو مقدار 5 و 3 عدد یک می شود.اجازه بدهید ببینیم که چطور این نتیجه را به دست می آید:

```
5:  00000000000000000000000000000101
3:  00000000000000000000000000000011
---------------------------------
1:  00000000000000000000000000000001
```

ابتدا دو عدد 5 و 3 به معادل باینری شان تبدیل می شوند. از آنجاییکه هر عدد صحیح 32 (int)بیت است از صفر برای پر کردن بیتهای خالی استفاده می کنیم. با استفاده از جدول درستی عملگر بیتیAND می توان فهمید که چرا نتیجه عدد یک می شود.

عملگر بیتی (|)OR

اگر مقادیر دو طرف عملگر بیتی OR هر دو صفر باشند نتیجه صفر در غیر اینصورت 1 خواهد شد.جدول درستی این عملگر در زیر آمده است:

X	Y	X OR Y
1	1	1
1	0	1
0	1	1
0	0	0

نتیجه عملگر بیتی OR در صورتی صفر است که عملوند های دو طرف آن صفر باشند.اگر فقط یکی از دو عملوند یک باشد نتیجه یک خواهد شد. به مثال زیر توجه کنید:

```
$result = 7 | 9;

echo $result;
```

```
15
```

وقتی که از عملگر بیتی OR برای دو مقدار در مثال بالا (7 و 9) استفاده می کنیم نتیجه 15 می شود.حال بررسی می کنیم که چرا این نتیجه به دست آمده است؟

```
7: 00000000000000000000000000000111
9: 00000000000000000000000000001001
---------------------------------
15: 00000000000000000000000000001111
```

با استفاده از جدول درستی عملگر بیتی OR می توان نتیجه استفاده از این عملگر را تشخیص داد.عدد 1111 باینری معادل عدد 15 صحیح است.

عملگر بیتی XOR(^)

جدول درستی این عملگر در زیر آمده است:

X	Y	X XOR Y
1	1	0
1	0	1
0	1	1
0	0	0

در صورتیکه عملوندهای دو طرف این عملگر هر دو صفر یا هر دو یک باشند نتیجه صفر در غیر اینصورت نتیجه یک می شود.در مثال زیر تاثیر عملگر بیتی XOR را بر روی دو مقدار مشاهده می کنید:

```
$result = 5 ^ 7;

echo $result;
```

```
2
```

در زیر معادل باینری اعداد بالا (5 و 7) نشان داده شده است.

```
5: 00000000000000000000000000000101
7: 00000000000000000000000000000111
---------------------------------
2: 00000000000000000000000000000010
```

با نگاه کردن به جدول درستی عملگر بیتی XOR ، می توان فهمید که چرا نتیجه عدد 2 می شود.

عملگر بیتی (~)NOT

این عملگر یک عملگر یگانی است و فقط به یک عملوند نیاز دارد.در زیر جدول درستی این عملگر آمده است:

NOT X	X
0	1
1	0

عملگر بیتی NOT مقادیر بیتها را معکوس می کند.در زیر چگونگی استفاده از این عملگر آمده است:

```
$result = ~7;

echo $result;
```

به نمایش باینری مثال بالا که در زیر نشان داده شده است توجه نمایید.

```
7: 00000000000000000000000000000111
------------------------------------
-8: 11111111111111111111111111111000
```

عملگر بیتی تغییر مکان(shift)

این نوع عملگرها به شما اجازه می دهند که بیتها را به سمت چپ یا راست جا به جا کنید.دو نوع عملگر بیتی تغییر مکان وجود دارد که هر کدام دو عملوند قبول می کنند.عملوند سمت چپ این عملگرها حالت باینری یک مقدار و عملوند سمت راست تعداد جابه جاییبیت ها را نشان می دهد.

مثال	دسته	نام	عملگر
x = y << 2;	Binary	تغییر مکان به سمت چپ	<<
x = y >> 2;	Binary	تغییر مکان به سمت راست	>>

عملگر تغییر مکان به سمت چپ

این عملگر بیتهای عملوند سمت چپ را به تعداد n مکان مشخص شده توسط عملوند سمت راست، به سمت چپ منتقل می کند. به عنوان مثال:

```
$result = 10 << 2;

echo $result;
```

```
40
```

در مثال بالا ما بیتهای مقدار 10 را دو مکان به سمت چپ منتقل کرده ایم، حال بیایید تاثیر این انتقال را بررسی کنیم:

```
10: 00000000000000000000000000001010
----------------------------------
40: 00000000000000000000000000101000
```

مشاهده می کنید که همه بیت ها به اندازه دو واحد به سمت چپ منتقل شده اند.در این انتقال دو صفر از صفرهای سمت چپ کم می شود و در عوض دو صفر به سمت راست اضافه می شود.

عملگر تغییر مکان به سمت راست

این عملگر شبیه به عمگر تغییر مکان به سمت چپ است با این تفاوت که بیت ها را به سمت راست جا به جا می کند. به عنوان مثال:

```
$result = 100 >> 4;

echo $result;
```

```
6
```

با استفاده از عملگرتغییر مکان به سمت راست بیت های مقدار 100 را به اندازه 4 واحد به سمت چپ جا به جا می کنیم. اجازه بدهید تاثیر این جا به جایی را مورد بررسی قرار دهیم:

```
100: 00000000000000000000000001100100
----------------------------------
  6: 00000000000000000000000000000110
```

هر بیت به اندازه 4 واحد به سمت راست منتقل می شود،بنابراین 4 بیت اول سمت راست حذف شده و چهار صفر به سمت چپ اضافه می شود.

عملگر رشته

در رشته ها تنها عملگری که استفاده می شود نقطه "." می باشد که دو رشته را به همدیگه متصل می کند. به مثال زیر توجه کنید:

```
<?php
    $string1 = "Hello";
    $string2 = $string1 . " world!";
    echo $string2;

    echo "<br>";

    $string1="Hello";
    $string1 .= " world!";
    echo $string1;
?>
```

```
Hello world!
Hello world!
```

رشته ها

رشته ها در PHP مجموعه ای از کارکترهای متوالی هستند که بین دو گیومه (کوتِیشِن) تک ‘ ’ یا جفت “ ” قرار می گیرند.
مثال:

```
$string = 'This is a text.';
$string = "This is also a text.";
```

درصورتی که بخواهید از خود کارکتر گیومه (تک یا جفت) در وسط رشته ای که ابتدا و انتهای آن توسط همان نوع گیومه مشخص شده است، استفاده کنید، باید قبل از گیومه میان رشته، کارکتر \ استفاده کنید.

```
echo 'It\'s mine. My name is "Mohammad"';
echo "My friend's name is \"Ali\".";
```

ادغام رشته ها

برای ادغام رشته ها در PHP از کارکتر نقطه (.) استفاده می شود:

```
$string1 = 'PHP';
$string2 = 'Programming';
echo $string1 . ' ' . $string2 . ' is full of enjoy.<br />';
```

```
PHP Programming is full of enjoy.
```

در PHP میتوانید رشته ها را با اعداد نیز ترکیب کنید:

```
$number = 5;
echo 'Test' . $number;
```

```
Test5
```

تفاوت رشته های محصور در گیومه جفت و تک

دستورات زیر را درنظر بگیرید:

```
$string = 5;
$text1 = "String is $string";
$text2 = 'String is $string';

echo $text1 . '<br/>';
echo $text2;
```

```
String is 5
String is $string
```

با اجرای دستورات فوق، عبارت String is 5 در متغیر $text1و عبارت String is $string در متغیر $text2ذخیره خواهد شد. علت این تفاوت در عملکرد، آن است که اسامی متغیرها در رشته هایی که بین دو گیومه جفت قرار دارند، پردازش شده و بجای نام آنها، مقدارشان در عبارت مورد استفاده قرار میگیرد. درمقابل رشته های محصور به گیومه تک، مورد هیچ پردازشی قرار نمیگیرند و به همان شکل که نوشته میشوند، مورد استفاده قرار خواهند گرفت. بنابراین اگر در رشته موردنظرتان، متغیری وجود ندارد، بهتر است آنرا در گیومه تک بگذارید تا سرعت پردازش کد شما افزایش یابد. همچنین باید دقت کنید که پردازش رشته های محصور در گیومه جفت نیز تنها محدود به اسامی متغیرهاست و هیچگونه فراخوانی توابع یا عمل محاسباتی ریاضی و... مورد پردازش قرار نخواهد گرفت. برای مثال، حاصل دستورات زیر:

```
$string = 5;
$text = "String = ($string + 5)";

echo $text ;
```

```
String = (5 + 5)
```

ذخیره شدن عبارت String = (5 + 5) در متغیر $text است. درواقع رای محاسبه صحیح مجموع و درج آن در رشته، باید بصورت زیر عمل کنید:

```
$string = 5;
$text = 'String = ' . ($string + 5);

echo $text ;
```

```
String = 10
```

که به موجب آن، ابتدا نتیجه محاسبه ($string + 5) با حاصل 10 و سپس استفاده از نتیجه این محاسبه در عبارت و ادغام با رشته ' = String'و درنتیجه ذخیره عبارت نهایی String = 10در متغیر $textاست.

استفاده از رشته بعنوان عدد

همانطور که در مثال قبل ملاحظه کردید، تبدیل عدد به رشته در زمان نیاز بطور خودکار انجام میشود. عکس این موضوع نیز صحیح است و رشته ها درصورت استفاده در عبارات محاسباتی، بطور خودکار به عدد تبدیل خواهند شد؛ بدین ترتیب که از ابتدای رشته، تا زمان رسیدن به اولین کارکتر غیر عددی، جدا شده و بصورت عدد تعبیر میشود. البته اگر رشته موردنظر با عدد شروع نشده باشد، یک ثابت خاص به نام) NaN مخفف (Not a Number بازگردانده خواهد شد. مثال:

```
$text = '52Ali';
$sum = 7 + $text;

echo $sum;
```

```
59
```

استفاده از Heredocs و Nowdocs

Heredocsو **Nowdocs** دو روش برای تعریف رشته های بزرگ در **PHP** می باشند. نحوه تعریف رشته در این دو روش به صورت زیر است:

```
<?php
    $string = <<<HEREDOC
                    ...
                HEREDOC;
?>
<?php
    $string = <<<'NOWDOC'
                    ...
                'NOWDOC';
?>
```

تفاوت این دو در این است که در **HEREDOC** متغیر ها همانند دابل کوتیشن پردازش می شوند ولی در **NOWDOC** نه. به مثال های زیر توجه کنید:

HEREDOC

```
<?php
    $heredoc = 'HEREDOC'

    $string = <<<HEREDOC
"The HEREDOC syntax is much cleaner to me and it is really useful ."
HEREDOC;

    echo $string;
?>
```

```
The HEREDOC syntax is much cleaner to me and it is really useful .
```

NOWDOC

```
<?php
    $nowdoc= 'NOWDOC'

    $string = <<<'NOWDOC'
"The $nowdoc syntax is much cleaner to me and it is really useful ."
'NOWDOC';
```

```
    echo $string;
?>
```

```
The $nowdoc syntax is much cleaner to me and it is really useful .
```

به این نکته هم توجه کنید که NOWDOC و HEREDOC را با حروف بزرگ بنویسید .در ضمن NOWDOC هم باید در داخل تک کوتیشن قرار گیرد. شما به جای این دو کلمه هر کلمه دیگری می توانید به کار ببرید. فقط باید تگ ابتدا و انتها دقیقا عین هم باشند.

آرایه ها

آرایه نوعی متغیر است که لیستی از آدرسهای مجموعه ای از داده های هم نوع را در خود ذخیره می کند، البته در PHP داده ها هم نوع نباشند مهم نیست.تعریف چندین متغیر از یک نوع برای هدفی یکسان بسیار خسته کننده است.مثلا اگر بخواهید صد متغیر از نوع اعداد صحیح تعریف کرده و از آنها استفاده کنید. مطمئنا تعریف این همه متغیر بسیار کسالت آور و خسته کننده است.اما با استفاده از آرایه می توان همه آنها را در یک خط تعریف کرد. در زیر راهی ساده برای تعریف یک آرایه نشان داده شده است:

```
$numbers = array() ;
```

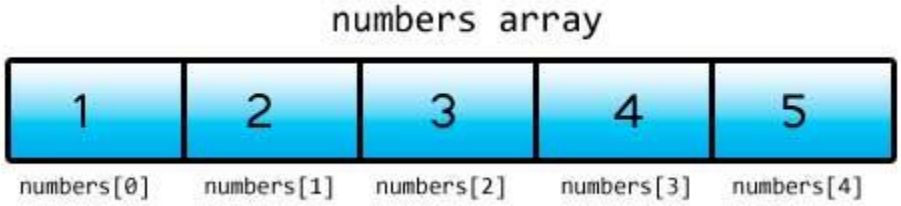
numbers نام آرایه را نشان می دهد.هنگام نامگذاری آرایه بهتر است که نام آرایه نشان دهنده نوع آرایه باشد.به عنوان مثال برای نامگذاری آرایه ای که اعداد را در خود ذخیره می کند از کلمه number استفاده کنید. با استفاده از دستور array یک آرایه ایجاد کرده ایم . حتی می توانیم به هنگام ایجاد آرایه مقادیر خانه های آن را نیز مشخص کنیم:

```
$numbers = array(1, 2, 3, 4, 5) ;
```

در این مثال 5 مقدار در 5 آدرس از فضای حافظه کامپیوتر شما ذخیره می شود. مثالا بالا را به روش زیر هم می توان پیاده سازی کرد:

```
$numbers = array();

$numbers[0] = 1;
$numbers[1] = 2;
$numbers[2] = 3;
$numbers[3] = 4;
$numbers[4] = 5;
```

اندیس یک آرایه از صفر شروع شده و به یک واحد کمتر از طول آرایه ختم می شود.به عنوان مثال شما یک آرایه 5 عضوی دارید،اندیس آرایه از 0 تا 4 می باشد چون طول آرایه 5 است پس 1-5 برابر است با 4.این بدان معناست که اندیس 0 نشان دهنده اولین عضو آرایه است و اندیس 1 نشان دهنده دومین عضو و الی آخر. برای درک بهتر مثال بالا به شکل زیر توجه کنید:



به هر یک از اجزاء آرایه و اندیسهای داخل کروشه توجه کنید.کسانی که تازه شروع به برنامه نویسی کرده اند معمولا در گذاشتن اندیس دچار اشتباه می شوند و مثلا ممکن است در مثال بالا اندیسها را از 1 شروع کنند.

انواع آرایه ها در PHP

در زبان PHP سه نوع آرایه وجود دارد:

- **آرایه های عددی (Numeric Array)**
- **آرایه های انجمنی (Associative Array)**
- **آرایه های چند بعدی (Multidimensional Array)**

آرایه های عددی (Numeric Array)

آرایه عددی یا Numeric Array آرایه ایست که در اکثر زبان های برنامه نویسی وجود دارد . در این آرایه ، به کل آرایه یک نام می دهیم بدین ترتیب هر یک از خانه های آرایه یک اندیس می گیرد و با آن اندیس می توانیم به یک خانه خاص آرایه دست بیابیم. به مثال زیر توجه کنید:

```
<?php

$name = array ('Jack' ,'Estephan' , 'Jordan');
echo $name[2];

?>
```

```
Jordan
```

آرایه های انجمنی (Associative Array)

آرایه انجمنی آرایه ای است که در آن به ازاء هر مقدار (هر خانه) یک فیلد بنام کلید نیز در نظر گرفته می شود که این کلید باید در بین کلید های خانه های دیگر منحصر به فرد و یکتا باشد . برای دستیابی به خانه های یک آرایه انجمنی ، دیگر از اندیس عددی استفاده نمی شود بلکه از فیلد کلید به عنوان رشته کلیدی دستیابی استفاده می شود . به مثال زیر توجه کنید:

```
<?php

$ages = array('Jack' => 21 , 'Estephan' => 24 , 'Jordan' => 19);
```

```
echo $ages["Jack"];

?>
```

```
21
```

این کد یک آرایه انجمنی با سه عضو ایجاد می کند و بعد به هنگام استفاده از کلید هایی که به هنگام تعریف داده شده برای دستیابی به عضو مربوطه استفاده می شود در اینجا باید حواستان باشد که این کلید مانند سایر موارد دیگر در PHP به حروف بزرگ و کوچک حساس است.(Case Sensitive)

آرایه های چند بعدی (Multidimensional Array)

این نوع آرایه ، آرایه ای است که هر عضو آن می تواند خود آرایه باشد . در حقیقت این نوع آرایه ، آرایه ای از آرایه ها می باشد . حال این آرایه می تواند عددی باشد یا انجمنی. به مثال زیر توجه کنید:

```
<?php

    $Families = array(
                "Griffin" =>array("Peter", "Lois", "Megan") ,
                "Quagmire"=>array("Glenn") ,
                "Brown"   =>array("Cleveland","Loretta","Junior")
                    );

    echo $Families['Griffin'][2];

?>
```

```
Megan
```

همانطور که از مثال فوق نیز پیداست یکی از تفاوت های عمده ای که آرایه در PHP با آرایه در سایر زبانها دارد اینست که در آرایه های چند بعدی PHP لازم نیست تعداد اعضای بعد های دوم به بعد با هم برابر باشند. و در آخر یاد آور می شویم که برای چاپ یک مقدار مورد نظر در آرایه های چند بعدی ابتدا نام آرایه اصلی سپس نام آرایه مورد نظر و بعد اندیس مقداری که قرار است چاپ شود را می نویسیم (مانند خط آخر مثال بالا)

دستورات شرطی

تقریبا همه زبانهای برنامه نویسی به شما اجازه اجرای کد را در شرایط مطمئن می دهند.حال تصور کنید که یک برنامه دارای ساختار تصمیم گیری نباشد و همه کدها را اجرا کند. این حالت شاید فقط برای چاپ یک پیغام در صفحه مناسب باشد ولی فرض کنید که شما بخواهید اگر مقدار یک متغیر با یک عدد برابر باشد سپس یک پیغام چاپ شود ان وقت با مشکل مواجه خواهید شد PHP .راه های مختلفی برای رفع این نوع مشکلات ارائه می دهد.

در این بخش با مطالب زیر آشنا خواهید شد:

- **دستور if**
- **دستور if…else**
- **عملگر سه تایی**
- **دستور switch**

دستور if

می توان با استفاده از دستور **if** و یک شرط خاص که باعث ایجاد یک کد می شود یک منطق به برنامه خود اضافه کنید.دتور **if** ساده ترین دستور شرطی است که برنامه می گوید اگر شرطی برقرار است کد معینی را انجام بده. ساختار دستور **if** به صورت زیر است:

```
if (condition)
code to execute;
```

قبل از اجرای دستور **if** ابتدا شرط بررسی می شود.اگر شرط برقرار باشد یعنی درست باشد سپس کد اجرا می شود.شرط یک عبارت مقایسه ای است. می توان از عملگرهای مقایسه ای برای تست درست یا اشتباه بودن شرط استفاده کرد.اجازه بدهید که نگاهی به نحوه استفاده از دستور **if** در داخل برنامه بیندازیم. برنامه زیر پیغام **Hello World** را اگر مقدار **number** کمتر از ۱۰ و **Goodbye World** را اگر مقدار **number** از ۱۰ بزرگتر باشد در صفحه نمایش می دهد:

```
1   <?php
2
3       $number = 5;
4       if ($number < 10)
5       echo ("Hello World.");
6
7       echo '<br/>';
8
9           $number = 15;
10      if ($number > 10)
11      echo("Goodbye World.");
12
13  ?>
```

```
Hello World.
Goodbye World.
```

در خط **3** یک متغیر با نام **number** تعریف و مقدار ۵ به آن اختصاص داده شده است. وقتی به اولین دستور **if** در خط **4** می رسیم برنامه تشخیص می دهد که مقدار **number** از ۱۰ کمتر است یعنی ۵ کوچکتر از ۱۰ است. منطقی است که نتیجه مقایسه درست می باشد بنابراین دستور **if** دستور را اجرا می کند(خط **5**) و پیغام **Hello World** چاپ می شود. حال مقدار **number** را به ۱۵ تغییر می دهیم (خط **9**).وقتی به دومین دستور **if** در خط **10** می رسیم برنامه مقدار **number** را با ۱۰ مقایسه می کند و چون مقدار **number** یعنی ۱۵ از ۱۰ بزرگتر است برنامه پیغام **Goodbye World** را چاپ می کند (خط **11**). به این نکته توجه کنید که دستور **if** را می توان در یک خط نوشت:

```
if ($number > 10) echo("Goodbye World.");
```

شما می توانید چندین دستور را در داخل دستور **if** بنویسید. کافیست که از یک آکولاد برای نشان دادن ابتدا و انتهای دستورات استفاده کنید. همه دستورات داخل بین آکولاد جز بدنه دستور **if** هستند. نحوه تعریف چند دستور در داخل بدنه **if** به صورت زیر است:

```
if (condition)
{
    statement1;
    statement2;
    .
    .
    .
    statementN;
}
```

این هم یک مثال ساده:

```
<?php

    $number = 15;
    if ($number > 10)
    {
        echo ("x is greater than 10.").'<br/>';
        echo ("This is still part of the if statement.");
    }

?>
```

```
x is greater than 10.
This is still part of the if statement.
```

در مثال بالا اگر مقدار **x** از ۱۰ بزرگتر باشد دو پیغام چاپ می شود. حال اگر به عنوان مثال آکولاد را حذف کنیم و مقدار **x** از ۱۰ بزرگتر نباشد مانند کد زیر:

```
<?php

    $number = 5;
    if ($number > 10)
    echo ("x is greater than 10.").'<br/>';
    echo ("This is still part of the if statement.");

?>
```

کد بالا در صورتی بهتر خوانده می شود که بین دستورات فاصله بگذاریم.

```
<?php
```

```
    $number = 5;
    if ($number > 10)
    echo ("x is greater than 10.").'<br/>';

    echo ("This is still part of the if statement.");

?>
```

```
This is still part of the if statement.
```

می بینِد که دستور دوم(خط 8) در مثال بالا جز دستور **if** نیست.اینجاست که چون ما فرض را بر این گذاشته ایم که مقدار **x** از ۱۰ کوچکتر است پس خط **This is still part of the if statement. (Really?)** چاپ می شود. در نتیجه اهمیت وجود آکولاد مشخص می شود.به عنوان تمرین همیشه حتی اگر فقط یک دستور در بدنه **if** داشتید برای آن یک آکولاد بگذارید.فراموش نکنید که از قلم انداختن یک آکولاد باعث به وجود آمدن خطا شده و یافتن آن را سخت می کند. یکی از خطاهای معمول کسانی که برنامه نویسی را تازه شروع کرده اند قرار دادن سیمیکولن در سمت راست پرانتز **if** است. به عنوان مثال:

```
<?php
    if ($number > 10);
    echo ("x is greater than 10.").'<br/>';
?>
```

به یاد داشته باشید که **if** یک مقایسه را انجام می دهد و دستور اجرایی نیست.بنابراین برنامه شما با یک خطای منطقی مواجه می شود.

دستور if...else

فرق این دستور با دستور **if** در این است که شما حالت دومی را نیز برای شرط در نظر دارید، یعنی اگر شرط اول برقرار بود، دستورالعمل مناسب اجرا میشود ولی اگر شرط برقرار نبود دستورالعمل جایگزین اجرا میشود (بر خلاف دستور **if** که اگر شرط اول برقرار نباشد هیچ دستورالعملی اجرا نمی شود). نحوه ی استفاده از این دستور نیز مانند دستور **if** است با این تفاوت که بعد از بسته شدن آکولاد دستور**if** ، دستور **else** اجرا می شود:

```
if (condition)
{
    code to execute;
}
else
{
    another code to execute;
}
```

مثال درس قبلی را کاملتر می کنیم، می خواهیم اگر مقدار **number** از **10** کمتر باشد پیغام **Hello World** و در غیر اینصورت پیام **Goodbye World** چاپ شود:

```
<?php

    $number = 15;
    if ($number < 10)
    {
        echo ("Hello World.");
    }
    else
    {
        echo("Goodbye World.");
    }

?>
```

```
Goodbye World.
```

همانطور که مشاهده می کنید قسمت **else** اجرا می شود چون مقدار متغیر **number** از عدد **10** بیشتر است.

دستور if...else if

این دستور این امکان را فراهم می کند تا در صورت عدم برقراری شرط دستورif ، شرطهای دیگری را نیز بررسی کنیم. ساختار کلی استفاده از این دستور به شرح زیر است:

```
if (condition)
{
    code to execute;
}
else if (another condition)
{
    another code to execute;
}
```

به مثال درس قبل بر می گردیم. ابتدا مقدار متغیر را چک می کنیم که آیا از مقدار 10 کمتر است، سپس چک می کنیم که آیا از مقدار 10 بزرگتر است و در نهایت اگر هیچکدام از این دو حالت برقرار نبود پیغام مناسب به کاربر نمایش داده شود:

```
<?php

    $number = 15;
    if ($number < 10)
    {
        echo ("Hello World.");
    }
    else if ($number > 10)
    {
        echo("Goodbye World.");
    }
    else
    {
        echo("Number is Equal 10");
    }

?>
```

```
Goodbye World.
```

در اینجا مثال های بسیار ساده ای زده شد ولی برخی اوقات شما نیاز دارید تا موارد پیچیده تری را محاسبه کنید و از عملگرها بهره بگیرید.

عملگر سه تایی

عملگر شرطی (:?) در PHP مانند دستور شرطی if...else عمل می کند. در زیر نحوه استفاده از این عملگر آمده است:

```
<condition> ? <result if true> : <result if false>
```

عملگر شرطی در PHP نیاز به سه عملوند دارد:

- شرط
- یک مقدار زمانی که شرط درست باشد
- یک مقدار زمانی که شرط نادرست باشد.

اجازه بدهید که نحوه استفاده این عملگر را در داخل برنامه مورد بررسی قرار دهیم.

```
<?php

    $number = 10;

    $Result = ($number == 10) ?"This is Ten Number":"Error";

    echo $Result;

?>
```

برنامه بالا نحوه استفاده از این عملگر شرطی را نشان می دهد. خطی که به صورت پررنگ نمایش داده شده است به این صورت ترجمه می شود که : اگر مقدار متغیر number با عدد 10 برابر بود پیغام This is Ten Number و در غیر اینصورت پیغام Errorچاپ شود.

دستور Switch

در **PHP** ساختاری به نام **switch** وجود دارد که به شما اجازه می دهد که با توجه به مقدار ثابت یک متغیر چندین انتخاب داشته باشید. دستور **switch** معادل دستور **if** تو در تو است با این تفاوت که در دستور **switch** متغیر فقط مقادیر ثابتی از اعداد ، رشته ها و یا کاراکترها را قبول می کند. مقادیر ثابت مقادیری هستند که قابل تغیر نیستند. در زیر نحوه استفاده از دستور **switch** آمده است:

```
switch (testVar)
{
    casecompareVal1:
        code to execute if testVar == compareVa11;
    break;
    casecompareVa12:
        code to execute if testVar == compareVa12;
    break;
       .
       .
       .
    casecompareVa1N:
        code to execute if testVer == compareVa1N;
    break;
    default:
        code to execute if none of the values above match the testVar;
    break;
}
```

ابتدا یک مقدار در متغیر **switch** که در مثال بالا **testVar** است قرار می دهید.این مقدار با هر یک از عبارتهای **case**داخل بلوک **switch** مقایسه می شود. اگر مقدار متغیر با هر یک از مقادیر موجود در دستورات **case**برابر بود کد مربوط به آن **case** اجرا خواهد شد. به این نکته توجه کنید که حتی اگر تعداد خط کدهای داخل دستور **case**از یکی بیشتر باشد نباید از آکولاد استفاده کنیم. آخر هر دستور **case** با کلمه کلیدی **break** تشخیص داده می شود که باعث می شود برنامه از دستور **switch** خارج شده ودستورات بعد از آن اجرا شوند. اگر این کلمه کلیدی از قلم بیوفتد برنامه با خطا مواجه می شود. دستور **switch** یک بخش **default** دارد. این دستور در صورتی اجرا می شود که مقدار متغیر با هیچ یک از مقادیر دستورات **case**برابر نباشد. دستور **default** اختیاری است و اگر از بدنه **switch** حذف شود هیچ اتفاقی نمی افتد. مکان این دستور هم مهم نیست اما بر طبق تعریف آن را در پایان دستورات می نویسند. به مثالی در مورد دستور **switch** توجه کنید:

```
<?php

    $number = 2;
    switch ($number)
    {
    case 1:
        echo 'One';
        break;
```

```
    case 2:
        echo 'TWo';
        break;
    case 3:
        echo 'Tree';
        break;
        default:
        break;
    }

?>
```

```
TWo
```

همانطور که در کد بالا مشاهده می کنید مقداری که به متغیر **number** اختصاص داده اید با مقادیر **case** مقایسه می شود و با هر کدام از آن مقادیر که برابر بود پیغام مناسب نمایش داده خواهد شد. اگر هم با هیچ کدام از مقادیر **case**ها برابر نبود دستور **default** اجرا می شود. یکی دیگر از ویژگیهای دستور **switch** این است که شما می توانید از دو یا چند **case** برای نشان داده یک مجموعه کد استفاده کنید. در مثال زیر اگر مقدار **number** ، ۱، ۲یا ۳ باشد یک کد اجرا می شود. توجه کنید که **case** ها باید پشت سر هم نوشته شوند.

```
<?php

    $number = 2;
    switch($number)
    {
        case 1:
        case 2:
        case 3:
        echo "This code is shared by three values." ;
            break;
    }

?>
```

```
This code is shared by three values.
```

دستورات تکرار

ساختارهای تکرار به شما اجازه می دهند که یک یا چند دستور کد را تا زمانی که یک شرط برقرار است تکرار کنید.بدون ساختارهای تکرار شما مجبورید همان تعداد کدها را بنویسید که بسیار خسته کننده است.مثلا شما مجبورید 10 بار جمله "Hello World." را تایپ کنید مانند مثال زیر :

```
echo "Hello World." . '<br/>';
echo "Hello World." . '<br/>';
echo "Hello World." . '<br/>';
echo "Hello World." . '<br/>';
echo "Hello World." . '<br/>';
echo "Hello World." . '<br/>';
echo "Hello World." . '<br/>';
echo "Hello World." . '<br/>';
echo "Hello World." . '<br/>';
echo "Hello World.";
```

```
Hello World.
Hello World.
Hello World.
Hello World.
Hello World.
Hello World.
Hello World.
Hello World.
Hello World.
Hello World.
```

البته شما می توانید با کپی کردن، این تعداد کد را راحت بنویسید ولی این کار در کل کیفیت کدنویسی را پایین می آورد.راه بهتر برای نوشتن کدهای بالا استفاده از حلقه ها است.

ساختارهای تکرار در PHP عبارتند از :

- **while**
- **do while**
- **for**
- **foreach**

دستور While

ابتدایی ترین ساختار تکرار در PHP حلقه While است.ابتدا یک شرط را مورد بررسی قرار می دهد و تا زمانیکه شرط برقرار باشد کدهای درون بلوک اجرا می شوند. ساختار حلقه While به صورت زیر است:

```
while(condition)
{
    code to loop;
}
```

می بینید که ساختار While مانند ساختار if بسیار ساده است. ابتدا یک شرط را که نتیجه آن یک مقدار بولی است مینویسیم اگر نتیجه درست یا trueباشد سپس کدهای داخل بلوک While اجرا می شوند. اگر شرط غلط یا false باشد وقتی که برنامه به حلقه While برسد هیچکدام از کدها را اجرا نمی کند. برای متوقف شدن حلقه باید مقادیر داخل حلقه While اصلاح شوند.

به یک متغیر شمارنده در داخل بدنه حلقه نیاز داریم. این شمارنده برای آزمایش شرط مورد استفاده قرار می گیرد و ادامه یا توقف حلقه به نوعی به آن وابسته است. این شمارنده را در داخل بدنه باید کاهش یا افزایش دهیم. در برنامه زیر نحوه استفاده از حلقه While آمده است:

```
1   <?php
2
3        $number = 1;
4        while ($number <= 10)
5        {
6           echo 'Hello World!'.'<br/>';
7               $number++ ;
8        }
9
10   ?>
```

```
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

برنامه بالا ۱۰ بار پیغام Hello World! را چاپ می کند. اگر از حلقه در مثال بالا استفاده نمی کردیم مجبور بودیم تمام ۱۰ خط را تایپ کنیم. اجازه دهید که نگاهی به کدهای برنامه فوق بیندازیم. ابتدا در خط 3 یک متغیر تعریف و از آن به عنوان شمارنده

حلقه استفاده شده است.سپس به آن مقدار ۱ را اختصاص می دهیم چون اگر مقدار نداشته باشد نمی توان در شرط از آن استفاده کرد. در خط 4 حلقه While را وارد می کنیم. در حلقه While ابتدا مقدار اولیه شمارنده با ۱۰ مقایسه می شود که آیا از ۱۰ کمتر است یا با آن برابر است. نتیجه هر بار مقایسه ورود به بدنه حلقه While و چاپ پیغام است. همانطور که مشاهده می کنید بعد از هر بار مقایسه مقدار شمارنده یک واحد اضافه می شود (خط 7). حلقه تا زمانی تکرار می شود که مقدار شمارنده از ۱۰ کمتر باشد.

اگر مقدار شمارنده یک بماند و آن را افزایش ندهیم و یا مقدار شرط هرگز false نشود یک حلقه بینهایت به وجود می آید. به این نکته توجه کنید که در شرط بالا به جای علامت < از => استفاده شده است. اگر از علامت < استفاده می کردیم کد ما ۹ بار تکرار می شد چون مقدار اولیه ۱ است و هنگامی که شرط به ۱۰ برسد falseمی شود چون ۱۰ 10 < نیست. اگر می خواهید یک حلقه بی نهایت ایجاد کنید که هیچگاه متوقف نشود باید یک شرط ایجاد کنید که همواره درست (true) باشد.

```
while(true)
{
    //code to loop
}
```

این تکنیک در برخی موارد کارایی دارد و آن زمانی است که شما بخواهید با استفاده از دستورات break و return که در آینده توضیح خواهیم داد از حلقه خارج شوید.

حلقه do while

حلقه **do while** یکی دیگر از ساختارهای تکرار است. این حلقه بسیار شبیه حلقه **while** است با این تفاوت که در این حلقه ابتدا کد اجرا می شودو سپس شرط مورد بررسی قرار می گیرد. ساختار حلقه **do while** به صورت زیر است :

```
do
{
    //code to repeat;
} while (condition);
```

همانطور که مشاهده می کنید شرط در آخر ساختار قرار دارد. این بدین معنی است که کدهای داخل بدنه حداقل یکبار اجرا می شوند.برخلاف حلقه **while** که اگر شرط نادرست باشد دستورات داخل بدنه اجرا نمی شوند. به مثال زیر توجه کنید :

```
1   <?php
2
3       $number = 1;
4       do
5       {
6           echo 'Hello World!' . '<br/>';
7           $number ++;
8       }
9       while ($number > 10);
10
11  ?>
```

```
Hello World!
```

همانطور که در کد بالا مشاهده می کنید ابتدا در خط 6 یک بار رشته **Hello World** چاپ می شود و سپس یک واحد به متغیر **number** در خط 7 اضافه می شود. در این خط مقدار متغیر **number** عدد 2 است و سپس عدد 2 در خط 9 با عدد 10 مقایسه می شود و چون از عدد 10 بزرگتر نیست شرط نادرست و حلقه متوقف می شود.در نتیجه حداقل یکبار حلقه اجرا می شود. اما همین مثال در صورتیکه شرط درست باشد نتیجه ای به صورت زیر خواهد داشت:

```
<?php

    $number = 1;
    do
    {
        echo 'Hello World!' . '<br/>';
        $number ++;
    }
    while ($number < 10);

?>
```

```
Hello World!
```

```
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

حلقه for

یکی دیگر از ساختارهای تکرار حلقه **for** است.این حلقه عملی شبیه به حلقه **while** انجام می دهد و فقط دارای چند خصوصیت اضافی است. ساختار حلقه **for** به صورت زیر است :

```
for(initialization; condition; operation)
{
    //code to repeat;
}
```

مقدار دهی اولیه (**initialization**) اولین مقداری است که به شمارنده حلقه می دهیم.شمارنده فقط در داخل حلقه **for** قابل دسترسی است.

شرط (**condition**) در اینجا مقدار شمارنده را با یک مقدار دیگر مقایسه می کند و تعیین می کند که حلقه ادامه یابد یا نه.

عملگر (**operation**) که مقدار اولیه متغیر را کاهش یا افزایش می دهد. در زیر یک مثال از حلقه **for** آمده است:

```
<?php

    for($i = 1; $i <= 10; $i++)
    {
        echo 'Hello World!' . '<br/>';
    }

?>
```

```
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

در برنامه بالا، ابتدا یک متغیر به عنوان شمارنده تعریف می کنیم و آن را با مقدار **1** مقدار دهی اولیه می کنیم. سپس با استفاده از شرط آن را با مقدار **10** مقایسه می کنیم که آیا کمتر است یا مساوی؟

توجه کنید که قسمت سوم حلقه (**i++**) فورا اجرا نمی شود. کد اجرا می شود و ابتدا رشته **Hello World!** را چاپ می کند. آنگاه یک واحد به مقدار **i** اضافه شده و مقدار **i** برابر **2** می شود و بار دیگر **i** با عدد **10** مقایسه می شود و این حلقه تا زمانی که

مقدار شرط true شود ادامه می یابد. اگر بخواهید معکوس بازه ای از اعداد را پیاده سازی کنید یعنی اعداد از بزرگ به کوچک چاپ شوند باید به صورت زیر عمل کنید :

```
for ($i = 10; $i > 0; $i--)
{
    echo $i . '<br/>';
}
```

```
10
9
8
7
6
5
4
3
2
1
```

کد بالا اعداد را از 10 به 1 چاپ می کند (از بزرگ به کوچک). مقدار اولیه شمارنده را 10 می دهیم و با استفاده از عملگر کاهش (--) برنامه ای که شمارش معکوس را انجام می دهد ایجاد می کنیم.

حلقه foreach

حلقه **foreach** یکی دیگر از ساختارهای تکرار در **PHP** می باشد که مخصوصا برای آرایه ها و مجموعه ها طراحی شده است.حلقه **foreach** با هر بار گردش در بین اجزاء ، مقادیر هر یک از آنها را در داخل یک متغیر موقتی قرار می دهد و شما می توانید بواسطه این متغیر به مقادیر دسترسی پیدا کنید. در زیر نحوه استفاده از حلقه **foreach** آمده است:

```
foreach (array as temporaryVar)
{
    //code to execute;
}
```

array نام آرایه است.سپس کلمه کلیدی **as** و بعد از آن **temporaryVar** (این نام کاملا اختیاری است) متغیری که مقادیر اجزای آرایه را در خود نگهداری می کند. در زیر نحوه استفاده از حلقه **foreach** امده است:

```
<?php

    $number = array(1, 2, 3, 4, 5);
    foreach ($number as $tempNumber)
    {
          echo $tempNumber . '<br/>';
    }

?>
```

```
1
2
3
4
5
```

در برنامه آرایه ای با ۵ جزء تعریف شده و مقادیر ۱ تا ۵ در آنها قرار داده شده است (خط **3**). در خط ۹ حلقه **foreach** شروع می شود.ما یک متغیر موقتی تعریف کرده ایم که اعداد آرایه را در خود ذخیره می کند. در هر بار تکرار از حلقه **foreach** متغیر موقتی **tempNumber** ، مقادیر عددی را از آرایه استخراج می کند. حلقه **foreach** مقادیر اولین تا آخرین جزء آرایه را در اختیار ما قرار می دهد. حلقه **foreach** برای دریافت هر یک از مقادیر آرایه کاربرد دارد.بعد از گرفتن مقدار یکی از اجزای آرایه ،مقدار متغیر موقتی را چاپ می کنیم. روش دیگر استفاده از حلقه **foreach** که معمولا برای آرایه های انجمنی و عددی به کار می رود به صورت زیر می باشد:

```
foreach (arrayas $key => $value)
{
    //statement
}
```

برای درک بهتر روش بالا به مثال زیر توجه کنید:

```php
<?php

    $number = array('one' =>'a', 'two' =>'b', 'three' =>'c');
    foreach ($number as $key => $value)
    {
        echo $key. ' => '. $value . '<br/>';
    }

?>
```

```
one => a
two => b
three => c
```

همانطور که در کد بالا مشاهده می کنید یک آرایه در خط 3 ایجاد کرده ایم که شامل کلید و مقدار است. برای به دست آوردن کلیدها از حالت دوم حلقه **foreach** استفاده کرده ایم. البته یاد آور می شویم که این روش زمانی مفید است که ما به اندیس ها و یا کلیدها در آرایه نیاز داشته باشیم مثلا در مورد آرایه مثال اول هم می توانیم اندیس اعداد را به این روش به دست بیاوریم:

```php
<?php

      $number = array(1, 2, 3, 4, 5);
    foreach ($number as $key => $value)
    {
        echo $key. ' => '. $value . '<br/>';
    }

?>
```

```
0 => 1
1 => 2
2 => 3
3 => 4
4 => 5
```

خارج شدن از حلقه با استفاده از break و continue

گاهی اوقات با وجود درست بودن شرط می خواهیم حلقه متوقف شود. سوال اینجاست که چطور این کار را انجام دهید؟ با استفاده از کلمه کلیدی break حلقه را متوقف کرده و با استفاده از کلمه کلیدی continue می توان بخشی از حلقه را رد کرد و به مرحله بعد رفت. برنامه زیر نحوه استفاده از continue و break را نشان می دهد:

```php
<?php

    echo 'Demonstrating the use of break.'.'<br/>';

    for ($x = 1; $x < 10; $x++)
    {
        if ($x == 5)
        break;

        echo $x . '<br/>';
    }

    echo '<br/>'. 'Demonstrating the use of continue.'.'<br/>';

    for ($x = 1; $x < 10; $x++)
    {
        if ($x == 5)
        continue;

        echo $x . '<br/>';
    }

?>
```

```
Demonstrating the use of break.
1
2
3
4

Demonstrating the use of continue.
1
2
3
4
6
7
8
9
```

در این برنامه از حلقه for برای نشان دادن کاربرد دو کلمه کلیدی فوق استفاده شده است اگر به جای for از حلقه های while و do…while استفاده می شد نتیجه یکسانی به دست می امد. همانطور که در شرط برنامه (خط 7) آمده است وقتی که مقدار xبه عدد ۵ رسید سپس دستور break اجرا شود(خط 8). حلقه بلافاصله متوقف می شود حتی اگر شرط 10 > x برقرار باشد.از طرف دیگر در خط 17 حلقه for فقط برای یک تکرار خاص متوقف شده و سپس ادامه می یابد).وقتی مقدار x برابر ۵ شود حلقه از ۵ رد شده و مقدار ۵ را چاپ نمی کند و بقیه مقادیر چاپ می شوند.

تابع

توابع به شما اجازه می دهند که یک رفتار یا وظیفه را تعریف کنید و مجموعه ای از کدها هستند که در هر جای برنامه می توان از آنها استفاده کرد. توابع در PHP و اکثر زبانهای برنامه نویسی بر دو نوعند:

- **توابع از پیش تعریف شده**
- **توابعی که توسط کاربر تعریف می شوند.**

ساده ترین ساختار یک تابع به صورت زیر است:

```
function MethodName()
{
    //code to execute;
}
```

به برنامه ساده زیر توجه کنید.در این برنامه از یک تابع برای چاپ یک پیغام در صفحه نمایش استفاده شده است:

```
<?php

    function PrintMessage()
    {
          echo 'Hello World!';
    }

    PrintMessage ();

?>
```

در خطوط 6-3 یک تابع تعریف کرده ایم. همانطور که مشاهده می کنید در خط 3 و برای تعریف تابع از کلمه کلیدی function سپس نام تابع و بعد از آن پرانتز باز و بسته استفاده کرده ایم. نام تابع ما PrintMessage() است. به این نکته توجه کنید که در نامگذاری تابع از روش پاسکال (حرف اول هر کلمه بزرگ نوشته می شود) استفاده کرده ایم. این روش نامگذاری قراردادی است و می توان از این روش استفاده نکرد، اما پیشنهاد می شود که از این روش برای تشخیص توابع استفاده کنید. بهتر است در نامگذاری توابع از کلماتی استفاده شود که کار ان تابع را مشخص می کند مثلا نام هایی مانند GoToBed یا OpenDoor. همچنین به عنوان مثال اگر مقدار برگشتی (در درس های آینده توضیح می دهیم (تابع یک مقدار بولی باشد می توانید اسم تابع خود را به صورت یک کلمه سوالی انتخاب کنید مانند IsLeapyear یا IsTeenager... . ولی از گذاشتن علامت سوال در آخر اسم تابع خودداری کنید. دو پرانتزی که بعد از نام می آید نشان دهنده آن است که نام متعلق به یک تابع است. بعد از پرانتزها دو آکولاد قرار می دهیم که بدنه تابع را تشکیل می دهد و کدهایی را که می خواهیم اجرا شوند را در داخل این آکولاد ها می نویسیم. در خط 8 تابع را صدا می زنیم. برای صدا زدن یک تابع کافیست نام آن را نوشته و بعد از نام پرانتزها را قرار دهیم.

برای اجرای تابع PrintMessage() برنامه از خط به محل تعریف تابع PrintMessage() می رود. مثلا وقتی ما تابع PrintMessage() را در خط 8 صدا می زنیم برنامه از خط 8 به خط 3 ، یعنی جایی که تابع تعریف شده می رود و کدهای آن را اجرا می کند.

مقدار برگشتی از یک تابع

توابع می توانند مقدار برگشتی از هر نوع داده ای داشته باشند. این مقادیر می توانند در محاسبات یا به دست آوردن یک داده مورد استفاده قرار بگیرند. در زندگی روزمره فرض کنید که کارمند شما یک تابع است و شما او را صدا می زنید و از او می خواهید که کار یک سند را به پایان برساند. سپس از او می خواهید که بعد از اتمام کارش سند را به شما تحویل دهد. سند همان مقدار برگشتی تابع است. نکته مهم در مورد یک تابع، مقدار برگشتی و نحوه استفاده شما از آن است. برگشت یک مقدار از یک تابع آسان است. کافیست در تعریف تابع به روش زیر عمل کنید:

```
function MethodName()
{
    return value;
}
```

همانطور که در خط مشاهده می کنید مقدار بازگشتی از تابع را جلوی دستور **return** می نویسیم. مثال زیر یک تابع که دارای مقدار برگشتی است را نشان می دهد:

```
<?php

    function CalculateSum()
    {
        $firstNumber = 10;
        $secondNumber = 5 ;
        $sum = $firstNumber + $secondNumber ;

        return $sum;
    }

    $result = CalculateSum();
    echo $result;

?>
```

```
15
```

در خطوط **5** و **6** مثال فوق، دو متغیر تعریف و مقدار دهی شده اند. توجه کنید که این متغیرها، متغیرهای محلی هستند. و این بدان معنی است که این متغیرها در سایر تابعها قابل دسترسی نیستند و فقط در تابعی که در آن تعریف شده اند قابل استفاده هستند. در خط **7** جمع دو متغیر در متغیر **sum** قرار می گیرد. در خط **9** مقدار برگشتی **sum** توسط دستور **return** فراخوانی می شود. در خط **12** یک متغیر به نام **result** تعریف می کنیم و تابع **()CalculateSum** را فراخوانی می کنیم.

تابع **()CalculateSum** مقدار **15** را بر می گرداند که در داخل متغیر **result** ذخیره می شود. در خط **13** مقدار ذخیره شده در متغیر **result** چاپ می شود. تابعی که در این مثال ذکر شد تابع کاربردی و مفیدی نیست.با وجودیکه کدهای زیادی در تابع بالا نوشته شده ولی همیشه مقدار برگشتی **15** است، در حالیکه می توانستیم به راحتی یک متغیر تعریف کرده و مقدار **15** را به آن

اختصاص دهیم. این تابع در صورتی کارامد است که پارامترهایی به آن اضافه شود که در درسهای آینده توضیح خواهیم داد. هنگامی که می خواهیم در داخل یک تابع از دستور **if** یا **switch** استفاده کنیم باید تمام کدها دارای مقدار برگشتی باشند. برای درک بهتر این مطلب به مثال زیر توجه کنید:

```
1   <?php
2
3       function GetNumber()
4       {
5           $number = 11 ;
6
7           if ($number > 10)
8           {
9               return $number;
10          }
11          else
12          {
13              return 0;
14          }
15      }
16
17      $result = GetNumber();
18      echo $result;
19
20  ?>
```

در خطوط 16-3 یک تابع با نام **GetNumber()** تعریف شده است. در خط **5** متغیری با مقدار **11** مقداردهی شده است که در خط خط **7** با مقدار **10** مقایسه می شود و چون مقدار این متغیر از **10** بیشتر است پس دستور **return** اول مقدار **11** را برمی گرداند. حال اگر مقدار این متغیر از **10** کمتر باشد دستور **return** مربوط به قسمت **else** اجرا و مقدار صفر چاپ می شود. که از کاربر یک عدد بزرگتر از **10** را می خواهد. اگر قسمت **else** دستور **if** و یا دستور **return** را از آن حذف کنیم در هنگام اجرای برنامه نتیجه چاپ نمی شود. چون اگر شرط دستور **if** نادرست باشد برنامه به قسمت **else** می رود تا مقدار صفر را بر گرداند و چون قسمت **else** حذف شده است برنامه هیچ مقداری را چاپ نمی کند و همچنین اگر دستور **return** حذف شود چون برنامه نیاز به مقدار برگشتی دارد برنامه هیچ مقداری را چاپ نمی کند.و آخرین مطلبی که در این درس می خواهیم به شما آموزش دهیم این است که شما می توانید از یک تابع که مقدار برگشتی ندارد خارج شوید. استفاده از **return** باعث خروج از بدنه تابع و اجرای کدهای بعد از آن می شود.

```
<?php

    function TestReturnExit()
    {
        echo 'Line 1 inside the method TestReturnExit()';
        return;
        echo 'Line 2 inside the method TestReturnExit()';
    }
```

```
    TestReturnExit();

?>
```

```
Line 1 inside the method TestReturnExit()
```

در برنامه بالا نحوه خروج از تابع با استفاده از کلمه کلیدی return و نادیده گرفتن همه کدهای بعد از این کلمه کلیدی نشان داده شده است. در پایان برنامه تابع تعریف شده (TestReturnExit()) فراخوانی و اجرا می شود.

پارامترها و آرگومان ها

پارامترها داده های خامی هستند که متد آنها را پردازش می کند و سپس اطلاعاتی را که به دنبال آن هستید در اختیار شما قرار می دهد .فرض کنید پارامترها مانند اطلاعاتی هستند که شما به یک کارمند می دهید که بر طبق آنها کارش را به پایان برساند. یک متد می تواند هر تعداد پارامتر داشته باشد. هر پارامتر می تواند از انواع مختلف داده باشد. در زیر یک متد با N پارامتر نشان داده شده است:

```
function MethodName(param1,param2, ...paramN)
{
    //code to execute;
}
```

پارامترها بعد از نام متد و بین پرانتزها قرار می گیرند. بر اساس کاری که متد انجام می دهد می توان تعداد پارامترهای زیادی به متد اضافه کرد. بعد از فراخوانی یک متد باید آرگومانهای آن را نیز تامین کنید. آرگومانها مقادیری هستند که به پارامترها اختصاص داده می شوند. ترتیب ارسال آرگومانها به پارامترها مهم است.اجازه بدهید که یک مثال بزنیم:

```
<?php

    function CalculateSum($number1,$number2)
    {
        return $number1 + $number2;
    }

    $result = CalculateSum(10,5);
    echo $result;

?>
```

در برنامه بالا یک متد به نام **CalculateSum()** (خطوط 3-6) تعریف شده است که وظیفه آن جمع مقدار دو عدد است. متد دارای دو پارامتر است که اعداد را به آنها ارسال می کنیم. در بدنه متد دستور **return** نتیجه جمع دو عدد را بر می گرداند. در خط 8 دو عدد 5 و 10 را به عنوان آرگومان به متد ارسال می کنیم. بعد از ارسال مقادیر 5 و 10 به پارامترها ، پارامترها آنها را دریافت می کنند. به این نکته نیز توجه کنید که نام پارامترها طبق قرارداد به شیوه کوهان شتری یا **camelCasing** (حرف اول دومین کلمه بزرگ نوشته می شود) نوشته می شود. در داخل بدنه متد (خط 5) دو مقدار با هم جمع می شوند و در خط 9 نتیجه چاپ می شود. دانستن مبانی مقادیر برگشتی و ارسال آرگومانها باعث می شود که شما متدهای کارامد تری تعریف کنید. تکه کد زیر نشان می دهد که شما حتی می توانید مقدار برگشتی از یک متد را به عنوان آرگومان به متد دیگر ارسال کنید.

```
<?php

    function MyMethod()
    {
        return 5;
```

```
    }

    function AnotherMethod($number)
    {
        echo $number;
    }

    AnotherMethod(MyMethod());

?>
```

```
5
```

چون مقدار برگشتی متد MyMethod()عدد 5 است و به عنوان آرگومان به متد AnotherMethod() ارسال می شود خروجی کد بالا هم عدد 5 است.

پارامترهای اختیاری

پارامترهای اختیاری همانگونه که از اسمشان پیداست اختیاری هستند و می توان به آنها آرگومان ارسال کرد یا نه. این پارامترها دارای مقادیر پیشفرضی هستند.اگر به اینگونه پارامترها آرگومانی ارسال نشود از مقادیر پیشفرض استفاده می کنند. به مثال زیر توجه کنید:

```
1   <?php
2
3       function PrintMessage($String = "Welcome to PHP Tutorials!")
4       {
5           echo $String . '<br/>';
6       }
7
8       PrintMessage();
9       PrintMessage("Learn PHP Today!");
10
11  ?>
```

```
Welcome to PHP Tutorials!
Learn PHP Today!
```

متد **PrintMessage()** (خطوط 3-6) یک پارامتر اختیاری دارد .برای تعریف یک پارامتر اختیاری می توان به آسانی و با استفاده از علامت = یک مقدار را به یک پارامتر اختصاص داد (مثال بالا خط 3). دو بار متد را فراخوانی می کنیم. در اولین فراخوانی (خط 8) ما آرگومانی به متد ارسال نمی کنیم بنابراین متد از مقدار پیشفرض **(Welcome to PHP Tutorials!)** استفاده می کند. در دومین فراخوانی(خط 9) یک پیغام (آرگومان) به متد ارسال می کنیم که جایگزین مقدار پیشفرض پارامتر می شود.

ارسال آرگومان به روش ارجاع و مقدار

آرگومانها را می توان به کمک ارجاع ارسال کرد. این بدان معناست که شما آدرس متغیری را ارسال می کنید نه مقدار آن را. ارسال با ارجاع زمانی مفید است که شما بخواهید یک آرگومان که دارای مقدار بزرگی است (مانند یک آبجکت) را ارسال کنید. در این حالت وقتی که آرگومان ارسال شده را در داخل متد اصلاح می کنیم مقدار اصلی آرگومان در خارج از متد هم تغییر می کند. اما در روش مقدار مقدار اصلی آرگومان در خارج از متد تغییر نمی کند .در زیر دستورالعمل پایه ای تعریف پارامتر ها که در آنها به جای مقدار از آدرس استفاده شده است نشان داده شده:

```
function FunctionName(& param1)
{
   //code to execute;
}
```

همانطور که در کد بالا مشاهده می کنید باید قبل از پارامتری که قرار است به روش ارجاع مقداری به آن ارسال شود علامت (&) قرار داده شود .اجازه دهید که تفاوت بین ارسال با ارجاع و ارسال با مقدار آرگومان را با یک مثال توضیح دهیم.

```
1  <?php
2      function ModifyNumberVal($number)
3      {
4          $number += 10;
5          echo 'Value of number inside method is '.$number.'<br/>';
6      }
7
8      function ModifyNumberRef(&$number)
9      {
10         $number += 10;
11         echo 'Value of number inside method is '.$number.'<br/>';
12     }
13
14     $num = 5;
15
16     echo 'num = '.$num;
17
18     echo '<br/><br/>';
19
20     echo 'Passing num by value to method ModifyNumberVal() ...<br/>';
21     ModifyNumberVal($num);
22     echo 'Value of num after exiting the method is '.$num.'<br/>';
23
24     echo '<br/><br/>';
25
26     echo 'Passing num by ref to method ModifyNumberRef() ...<br/>';
27     ModifyNumberRef($num);
28     echo 'Value of num after exiting the method is  '. $num.'<br/>';
29 ?>
```

```
30
31
```

```
num = 5

Passing num by value to method ModifyNumberVal() ...
Value of number inside method is 15.
Value of num after exiting the method is 5.

Passing num by ref to method ModifyNumberRef() ...
Value of number inside method is 15.
```

در برنامه بالا دو متد که دارای یک هدف یکسان هستند تعریف شده اند و آن اضافه کردن عدد 10 به مقداری است که به آنها ارسال می شود. اولین متد (خطوط 6-2) دارای یک پارامتر است که نیاز به یک مقدار آرگومان دارد. وقتی که متد را صدا می زنیم و آرگومانی به آن اختصاص می دهیم (خط 21) ،کپی آرگومان به پارامتر متد ارسال می شود. بنابراین مقدار اصلی متغیر خارج از متد ($num) هیچ ارتباطی به پارامتر متد ندارد.سپس مقدار 10 را به متغیر پارامتر (number) اضافه کرده و نتیجه را چاپ می کنیم (خطوط 5 و 4). برای اثبات اینکه متغیر $num هیچ تغییری نکرده است مقدار آن را یکبار دیگر چاپ کرده و مشاهده می کنیم که تغییری نکرده است (خط 22). دومین متد (خطوط 12-8) نیاز به یک مقدار با ارجاع دارد. در این حالت به جای اینکه یک کپی از مقدار به عنوان آرگومان به ان ارسال شود آدرس متغیر به آن ارسال می شود .حال پارامتر به مقدار اصلی متغیر که زمان فراخوانی متد به آن ارسال می شود دسترسی دارد. وقتی که ما مقدار متغیر پارامتری که شامل آدرس متغیر اصلی است را تغییر می دهیم (خط 10) در واقع مقدار متغیر اصلی در خارج از متد را تغییر داده ایم. در نهایت مقدار اصلی متغیر را وقتی که از متد خارج شدیم را نمایش می دهیم و مشاهده می شود که مقدار آن واقعا تغییر کرده است.

محدوده متغیر

متغیرها در PHP دارای محدوده هستند. محدوده یک متغیر به شما می گوید که در کجای برنامه می توان از متغیر استفاده کرد و یا متغیر قابل دسترسی است. به عنوان مثال متغیری که در داخل یک متد تعریف می شود فقط در داخل بدنه متد قابل دسترسی است. می توان دو متغیر با نام یکسان در دو متد مختلف تعریف کرد. برنامه زیر این ادعا را اثبات می کند :

```
<?php

    function firstLocalVariable()
    {
        $number = 10;
        echo $number;
    }

    function secondLocalVariable()
    {
        $number = 5;
        echo $number;
    }

    firstLocalVariable ();
    echo '<br/>';
    secondLocalVariable ();

?>
```

```
10
5
```

مشاهده می کنید که حتی اگر ما دو متغیر با نام یکسان تعریف کنیم (خطوط 5 و 11) که دارای محدوده های متفاوتی هستند، می توان به هر کدام از آنها مقادیر مختلفی اختصاص داد. متغیر تعریف شده در داخل متد firstLocalVariable () هیچ ارتباطی به متغیر داخل متد secondLocalVariable() ندارد. وقتی به مبحث کلاسها رسیدیم در این باره بیشتر توضیح خواهیم داد. php دارای چهار محدوده است:

- **متغیرهای محلی (Local)**
- **متغیرهای سراسری (Global)**
- **متغیرهای ایستا (Static)**

متغیرهای محلی

متغیرهایی که داخل توابع تعریف می شوند محلی هستند و فقط داخل همان تابع قابل استفاده اند. به مثال زیر توجه کنید:

```
1   <?php
2
3       function LocalVariable()
4       {
5           $number = 10;
6           echo $number;
7       }
8
9       LocalVariable ();
10      echo $number;
11
12  ?>
```

```
10
Notice: Undefined variable: number in C:\wamp\www\test.php on line 10
```

همانطور که مشاهده می کنید با فراخوانی متد در خط 9 مقدار متغیر number چاپ می شود ولی در خط 10 که سعی در چاپ مقدار این متغیر داریم با پیغام خطا مواجه می شویم چون طول عمر این متغیر تا زمانی است که تابع به پایان نرسیده است. با پایان تابع متغیر و مقدار آن هم از بین می رود در نتیجه در خارج از تابع نمی توان مقدار آن را چاپ کرد.

متغیرهای سراسری

متغیرهایی که در بیرون تابع تعریف می شوند از نوع سراسری هستند. به مثال زیر توجه کنید:

```
<?php

    $firstNumber  = 10;
    $secondNumber = 5;
    $Sum;

    function GlobalVariable()
    {
        global $firstNumber,$secondNumber,$Sum;
        $Sum = $firstNumber + $secondNumber;
    }

    GlobalVariable ();
    echo $Sum;

 ?>
```

15

متغیرهای **firstNumber** و **secondNember** و **Sum** در بیرون تابع تعریف شده اند و از نوع سراسری هستند، در داخل تابع اگر بخواهیم به مقدار آنها دسترسی پیدا کنیم باید ابتدا با کلمه کلیدی **global** در تابع تعریف کنیم سپس از آن استفاده نماییم.روش دیگر برای دسترسی به متغیرهای سراسری استفاده از آرایه فوق سراسری **GLOBALS$** است. یعنی کد بالا را به صورت زیر هم می توان نوشت:

```
<?php

    $firstNumber  = 10;
    $secondNumber = 5;
    $Sum;

    function GlobalVariable()
    {
        $GLOBALS["Sum"] = $GLOBALS["firstNumber"] + $GLOBALS["secondNumber"];
    }

    GlobalVariable ();
    echo $Sum;

?>
```

در مورد آرایه های فوق سراسری در درس های بعد توضیح می دهیم.

متغیرهای ایستا

با اتمام اجرای تابع تمام متغیرها و آن تابع از بین می شوند، به غیر از متغیرهایی که بصورت **static** تعریف شده باشند، به مثال زیر توجه کنید:

```
<?php

    function StaticVariable()
    {
        static $firstNumber  = 10;
        echo $firstNumber . '<br/>';
        $firstNumber ++;
    }

    StaticVariable();
    StaticVariable();
    StaticVariable();
```

```
?>
```

```
10
11
12
```

همانطور که در کد بالا مشاهده می کنید هر بار که تابع فراخوانی می‌شود، متغیر firstNumber که به صورت static تعریف شده، مقدار قبلی خود را حفظ می کند. بنابراین با هر بار فراخوانی، مقداری آن به اضافه 1 شده و ذخیره می‌گردد.در خطوط 8-3 یک متد و در داخل آن یک متغیر از نوع ایستا (static) تعریف شده است (خط 5). در خطوط 12-10 سه بار متد را فراخوانی کرده ایم. در فراخوانی اول مقدار 10 چاپ می شود. در خط 7 یک واحد به این متغیر اضافه می شود و این مقدار در فراخوانی دوم چاپ می شود (مقدار 11). در فراخوانی سوم هم یک واحد به مقدار قبلی اضافه شده (1+11) و این مقدار یعنی 12 چاپ می شود.

بازگشت (Recursion)

بازگشت فرایندی است که در آن متد مدام خود را فراخوانی می کند تا زمانی که به یک مقدار مورد نظر برسد. بازگشت یک مبحث پیچیده در برنامه نویسی است و تسط به آن کار را حتی نیست. به این نکته هم توجه کنید که بازگشت باید در یک نقطه متوقف شود وگرنه برای بی نهایت بار، متد، خود را فراخوانی می کند. در این درس یک مثال ساده از بازگشت را برای شما توضیح می دهیم. فاکتوریل یک عدد صحیح مثبت **(n!)** شامل حاصل ضرب همه اعداد مثبت صحیح کوچکتر یا مساوی آن می باشد. به فاکتوریل عدد **5** توجه کنید.

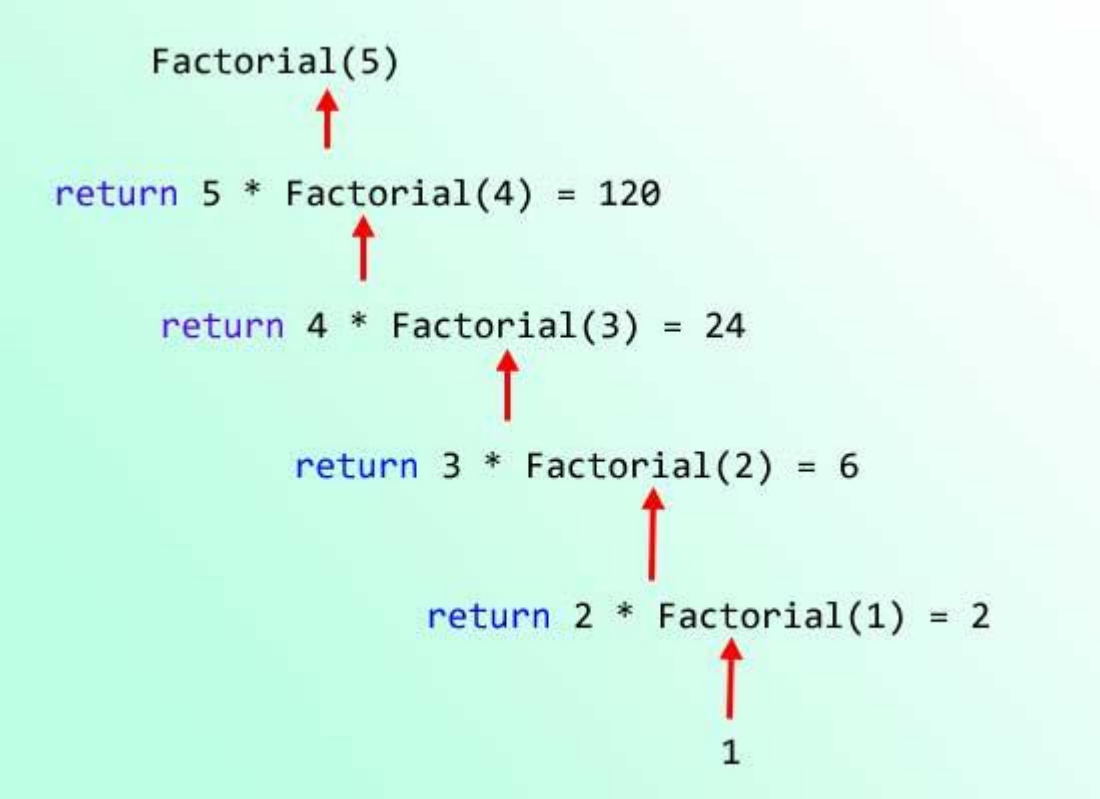
```
5! = 5 * 4 * 3 * 2 * 1 = 120
```

بنابراین برای ساخت یک متد بازگشتی باید به فکر توقف آن هم باشیم.بر اساس توضیح بازگشت ، فاکتوریل فقط برای اعداد مثبت صحیح است. کوچکترین عدد صحیح مثبت **1** است. در نتیجه از این مقدار برای متوقف کردن بازگشت استفاده می کنیم.

```
<?php
    function Factorial($number)
    {
        if ($number == 1)
        return 1;

        return $number * Factorial($number - 1);
    }

    echo Factorial(5);
?>
```

```
120
```

متد مقدار بزرگی را بر می گرداند چون محاسبه فاکتوریل می تواند خیلی بزرگ باشد. متد یک آرگومان که یک عدد است و می تواند در محاسبه مورد استفاده قرار گیرد را می پذیرد. در داخل متد یک دستور **if** می نویسیم و در خط **4**می گوییم که اگر آرگومان ارسال شده برابر **1** باشد سپس مقدار **1** را برگردان در غیر اینصورت به خط بعد برو. این شرط باعث توقف تکرارها نیز می شود. در خط **7** مقدار جاری متغیر **number** در عددی یک واحد کمتر از خودش **(number – 1)** ضرب می شود.در این خط متد **Factorial** خود را فراخوانی می کند و آرگومان آن در این خط همان **number – 1** است. مثلا اگر مقدار جاری **number** عدد **10** باشد یعنی اگر ما بخواهیم فاکتوریل عدد **10** را به دست بیاوریم آرگومان متد **Factorial** در اولین ضرب **9** خواهد بود. فرایند ضرب تا زمانی ادامه می یابد که آرگومان ارسال شده با عدد **1** برابر نشود. شکل زیر فاکتوریل عدد **5** را نشان می دهد.



کد بالا را به وسیله یک حلقه for نیز می توان نوشت:

```
<?php
      $factorial = 1;

      for ( $counter = 5; $counter >= 1; $counter-- )
         $factorial *= $counter;

      echo $factorial;
?>
```

```
120
```

این کد از کد معادل بازگشتی آن آسان تر است. از بازگشت در زمینه های خاصی در علوم کامپیوتر استفاده می شود. استفاده از بازگشت حافظه زیادی اشغال می کند پس اگر سرعت برای شما مهم است از آن استفاده نکنید.

سربارگذاری متدها

در درس های قبل با چگونگی ایجاد متدها آشنا شدید. در این درس می خواهیم شما را با یک مفهوم دیگر درباره متدها به نام سربارگذاری متدها (Method Overloading) آشنا کنیم. در PHP تعریف دو متد با نام یکسان که دارای تعداد پارامترهای متفاوتی باشند امکان پذیر نیست. به مثال زیر توجه کنید:

```
<?php

    function ShowMessage()
    {
        echo 'Hello World !';
    }

    function ShowMessage($string)
    {
        echo 'Hello '.$string;
    }

    ShowMessage();
?>
```

```
(!) Fatal error: Cannot redeclare Person::showMessage() in
C:\wamp\www\Tuts\index.php on line 11
```

با اجرای کد بالا با خطا مواجه می شوید، چون PHP نمی داند که شما کدام متد را فراخوانی کرده اید. در PHP توابعی وجود دارند که توسط توسعه دهندگان این زبان برای مقاصد خاصی تعریف شده اند و همراه با نصب PHP به صورت توکار وجود دارند. با این توابع، توابع از پیش تعریف شده (Predefined functions) می گویند. مثلا از برخی از این توابع برای به دست اوردن طول یک رشته، به دست آوردن تعداد عناصر موجود در یک آرایه ، کار با تاریخ و ساعت، کار با پوشه ها و فایل ها ... استفاده می شود. قدرت PHP در همین توابع از پیش تعریف شده است و تعداد آنها در هر نسخه جدید از PHP تغییر کرده و بیشتر می شود .یکی از این متدها ، متد func_get_args() است، که می توان با استفاده از آن یک متد با تعداد پارامترهای متفاوت ایجاد کرد:

```
<?php
    function showMessage()
    {
        $string = func_get_args();
        foreach($string as $str)
        {
            echo 'Hello '.$str.'<br/>';
        }
    }

    showMessage('World!');
    showMessage('Jack','Smith','John');
```

```
?>
```

```
Hello World!
Hello Jack
Hello Smith
```

همانطور که در کد بالا مشاهده می کنید با استفاده از این متد توانستیم در هربار فراخوانی متد ShowMessage تعداد پارامترهای متفاوتی به آن ارسال کنیم.

برنامه نویسی شیء گرا

برنامه نویسی شی گرا (OOP) شامل تعریف کلاسها و ساخت اشیاء مانند ساخت اشیاء در دنیای واقعی است.برای مثال یک ماشین را در نظر بگیرید.این ماشین دارای خواصی مانند رنگ ، سرعت ، مدل ، سازنده و برخی خواص دیگر است. همچنین دارای رفتارها و حرکاتی مانند شتاب و پیچش به چپ و راست و ترمز است.اشیاء در PHP تقلیدی از یک شی مانند ماشین در دنیای واقعی هستند.برنامه نویسی شی گرا با استفاده از کدهای دسته بندی شده کلاسها و اشیاء را بیشتر قابل کنترل می کند. در ابتدا ما نیاز به تعریف یک کلاس برای ایجاد اشیاء مان داریم. شیء در برنامه نویسی شی گراء از روی کلاسی که شما تعریف کرده اید ایجاد می شود.برای مثال نقشه ساختمان شما یک کلاس است که ساختمان از روی آن ساخته شده است.کلاس شامل خواص یک ساختمان مانند مساحت،بلندی و مواد مورد استفاده در ساخت خانه می باشد. در دنیای واقعی ساختمان ها نیز بر اساس یک نقشه (کلاس) پایه گذاری(تعریف)شده اند. برنامه نویسی شیء گرا یک روش جدید در برنامه نویسی است که بوسیله برنامه نویسان مورد استفاده قرار می گیرد و به آنها کمک می کند که برنامه هایی با قابلیت استفاده مجدد، خوانا و راحت طراحی کنند. PHP نیز یک برنامه شی گراست .در درس بعد به شما نحوه تعریف کلاس و استفاده از اشیاء آموزش داده خواهد شد.همچنین شما با مفهوم وراثت که از مباحث مهم در برنامه نویسی شیء گرا است در آینده آشنا می شوید.

کلاس

کلاس به شما اجازه می دهد یک نوع داده ای که توسط کاربر تعریف می شود و شامل متغیرها و خواص **(properties)** و متدها است را ایجاد کنید. کلاس در حکم یک نقشه برای یک شی می باشد. شی یک چیز واقعی است که از ساختار، خواص و یا رفتارهای کلاس پیروی می کند. وقتی یک شی می سازید یعنی اینکه یک نمونه از کلاس ساخته اید (در درس ممکن است از کلمات شی و نمونه به جای هم استفاده شود).برای تعریف یک کلاس از کلمه کلیدی **class** استفاده شود:

```
class ClassName
{
    Variable1;
    Variable2;
    ...
    VariableN;

    method1;
    method2;
    ...
    methodN;
}
```

این کلمه کلیدی را قبل از نامی که برای کلاسمان انتخاب می کنیم می نویسیم. در نامگذاری کلاسها هم از روش نامگذاری **Pascal** استفاده می کنیم. در بدنه کلاس متغیرها و متدهای آن قرار داده می شوند. متغیرها اعضای داده ای خصوصی هستند که کلاس از آنها برای رفتارها و ذخیره مقادیر خاصیت هایش **(property)** استفاده می کند. متدها رفتارها یا کارهایی هستند که یک کلاس می تواند انجام دهد. در زیر نحوه تعریف و استفاده از یک کلاس ساده به نام **person** نشان داده شده است:

```
1   <?php
2
3       class Person
4       {
5           public $name;
6           public $age;
7           public $height;
8
9           public function TellInformation()
10          {
11              echo 'Name: '   . $this -> name .'<br/>';
12              echo 'Age:  '   . $this -> age .'<br/>';
13              echo 'Height: '. $this -> height;
14          }
15      }
16
17
18      $person1 = new Person();
19      $person2 = new Person();
```

```
20
21      $person1 -> name = 'Jack';
22      $person1 -> age = 21;
23      $person1 -> height = 180;
24      $person1 -> TellInformation ();
25
26      echo "<br/><br/>";//Separator
27
28      $person2 -> name = 'Mike';
29      $person2 -> age = 23;
30      $person2 -> height = 158;
31      $person2 -> TellInformation ();
32
33  ?>
```

```
Name: Jack
Age: 21
Height: 160

Name: Mike
Age: 23
Height: 158
```

همانطور که در کد بالا مشاهده می کنید، در خطوط 15-3 کلاسی به نام **Person** تعریف شده است. در خط ۳ یک نام به کلاس اختصاص داده ایم تا به وسیله آن قابل دسترسی باشد .در داخل بدنه کلاس متغیرهای آن تعریف شده اند (خطوط 5-7). همچنین سطح دسترسی آنها را **public** تعریف کرده ایم تا در دیگر کلاسها قابل شناسایی باشند. درباره سطوح دسترسی در یک درس جداگانه بحث خواهیم کرد.

این سه متغیر تعریف شده خصوصیات واقعی یک فرد در دنیای واقعی را در خود ذخیره می کنند. یک فرد در دنیای واقعی دارای نام، سن، و قد می باشد. در خطوط 14-9 یک متد هم در داخل کلاس به نام **TellInformation()** تعریف شده است که رفتار کلاسمان است و مثلا اگر از فرد سوالی بپرسیم در مورد خودش چیزهایی می گوید. در داخل متد کدهایی برای نشان دادن مقادیر موجود در متغیرها نوشته شده است. نکته ای درباره متغیرها وجود دارد و این است که چون متغیرها در داخل کلاس تعریف و به عنوان اعضای کلاس در نظر گرفته شده اند، محدوده آنها یک کلاس است.

این بدین معناست که متغیرها فقط می توانند در داخل کلاس یعنی جایی که به آن تعلق دارند و یا به وسیله نمونه ایجاد شده از کلاس مورد استفاده قرار بگیرند.در خطوط 18 و 19 دو نمونه یا دو شی از کلاس **Person** ایجاد می کنیم .برای ایجاد یک نمونه از یک کلاس باید از کلمه کلیدی **new** و به دنبال آن نام کلاس و یک جفت پرانتز قرار دهیم:

```
$person1 = new Person();
$person2 = new Person();
```

در خطوط 23-21 مقادیری به متغیرهای اولین شی ایجاد شده از کلاس **person1** اختصاص داده شده است. برای دسترسی به متغیرها یا متدهای یک شی از علامت -> استفاده می شود. به عنوان مثال کد **$person1 -> name** نشان دهنده متغیر

name از شی person1 می باشد .برای چاپ مقادیر متغیرها باید متد TellInformation() شی person1 را فراخوانی می کنیم(خط 24). به کلمه کلیدی this در خطوط 13-11 توجه کنید. این کلمه کلیدی اشاره به شی جاری دارد. یعنی وقتی مقادیر از طریق شی person1 ارسال می شوند منظور از this شی person1 و وقتی از طریق شی person2 ارسال می شوند منظور از this شی person2 می باشد. همانطور که در خطوط 13-11، 23-21 و 30-28 مشاهده می کنید برای دسترسی به متغیرها، قبل از نام آنها علامت $ را به کار نمی بریم.

در خطوط 30-28 نیز مقادیری به شی دومی که قبلا از کلاس ایجاد شده تخصیص می دهیم و سپس متد TellInformation() را فراخوانی می کنیم. به این نکته توجه کنید که person1 و person2 نسخه های متفاوتی از هر متغیر دارند بنابراین تعیین یک نام برای person2 هیچ تاثیری بر نام person1 ندارد. و نکته آخر اینکه نام کلاس ها بر خلاف متغیر ها به بزرگی و کوچکی حروف حساس نیست یعنی یک شیء را به دو صورت زیر می توان از کلاس ایجاد کرد:

```
$person = new Person();
$person = new PERSON();
```

در مورد اعضای کلاس در درسهای آینده توضیح خواهیم داد.

سازنده

سازنده ها متدهای خاصی هستند که وجود آنها برای ساخت اشیا لازم است. آنها به شما اجازه می دهند که متغیرهای کلاس را مقداردهی اولیه کنید و کدهایی که را که می خواهید هنگام ایجاد یک شی اجرا شوند را به برنامه اضافه کنید. اگر از هیچ سازنده ای در کلاس تان استفاده نکنید، PHPاز سازنده پیشفرض که یک متد بدون پارامتر است استفاده می کند. در مثال زیر یک کلاس که شامل سازنده پیشفرض (خطوط 9-12) است را مشاهده می کنید:

```
1   <?php
2
3       class Person
4       {
5           public $name;
6           public $age;
7           public $height;
8
9           public function Person()
10          {
11
12          }
13
14          public function TellInformation()
15          {
16              echo 'Name: '   . $this -> name .'<br/>';
17              echo 'Age: '    . $this -> age .'<br/>';
18              echo 'Height: '. $this -> height;
19          }
20      }
21
22      $person1 = new Person();
23      var_dump ($person1);
24
25  ?>
```

```
object(Person)[1]
  public 'name' => null
  public 'age' => null
  public 'height' => null
```

می توانیم این سازنده را هم تعریف نکنیم چون PHP به طور خودکار آن را ایجاد می کند.همانطور که در خط 22 و 23 مشاهده می کنید ما یک شی یا یک نمونه از کلاس ایجاد کرده ایم (در درس بعد بیشتر توضیح می دهیم) و با استفاده از تابع var_dump مقادیر موجود در این شی را چاپ کرده ایم. در خروجی مشاهده می کنید که سازنده پیشفرض به هر سه متغیر مقدار null را اختصاص داده است.مثلا بهتر است که با استفاده از سازنده مقدار پیشفرض به متغیرها اختصاص دهیم. مثلا فردی که به دنیا می آید نام (name) ندارد ولی سن (age) و قد (height) دارد. پس می توانیم به صورت زیر این مقادیر را به متغیرها با استفاده از سازنده پیشفرض اختصاص دهیم:

```php
<?php

   class Person
   {
       public $name;
       public $age;
       public $height;

       public function Person()
       {
           $this -> name   ='';
           $this -> age    = 9;
           $this -> height = 30;
       }

       public function TellInformation()
       {
           echo 'Name: '   . $this -> name   .'<br/>';
           echo 'Age: '    . $this -> age    .' Month'.'<br/>';
           echo 'Height: '. $this -> height .' cm';
       }
   }

   $person1 = new Person();
   $person1 -> TellInformation ();

?>
```

```
Name:
Age: 9 Month
Height: 30 cm
```

به این نکته توجه کنید که سازنده درست شبیه به یک متد است با این تفاوت که

- مقدار برگشتی ندارد.
- نام سازنده باید دقیقا شبیه نام کلاس باشد.

حال فرض کنید می خواهیم سازنده ای ایجاد کنیم که بعد از ایجاد یک شی از کلاس، متغیرهای شی ایجاد شده را خودمان و با استفاده از سازنده ای که تعریف کرده ایم مقداردهی کنیم. به کد زیر توجه کنید:

```php
<?php

  class Person
  {
        public $name;
        public $age;
        public $height;
```

```
8
9           public function Person($n, $a, $h)
10          {
11              $this -> name   = $n;
12              $this -> age    = $a;
13              $this -> height = $h;
14          }
15
16          public function TellInformation()
17          {
18              echo 'Name: '   . $this -> name .'<br/>';
19              echo 'Age: '    . $this -> age  .'<br/>';
20              echo 'Height: '. $this -> height;
21          }
22      }
23
24      $person1 = new Person("Jack", 21, 160);
25      $person1 -> TellInformation();
26
27      echo '<br/><br/>';
28
29      $person2 = new Person("Mike", 32, 158);
30      $person2 -> TellInformation();
31
32  ?>
```

```
 Name: Jack
Age: 21
Height: 160

Name: Mike
Age: 32
Height: 158
```

همانطور که مشاهده می کنید در مثال بالا سازنده ای را سه آرگومان قبول می کند به کلاس **Person** اضافه کرده ایم (خطوط 9-14). در خطوط 24 و 29 بعد از ایجاد شی و در داخل پرانتزها سه مقدار را به سازنده (خط 9) ارسال می کنیم و سازنده این مقادیر را به متغیرها (خطوط 5-7) اختصاص می دهد.

مخرب

مخرب ها نقطه مقابل سازنده ها هستند. مخرب ها متدهای خاصی هستند که هنگام تخریب یک شی فراخوانی می شوند. اشیا از حافظه کامپیوتر استفاده می کنند و اگر پاک نشوند ممکن است با کمبود حافظه مواجه شوید. می توان از مخرب ها برای پاک کردن منابعی که در برنامه مورد استفاده قرار نمی گیرند استفاده کرد. معمولا **PHP** به صورت اتوماتیک از زباله روب (**garbage collection**) برای پاک کردن حافظه استفاده می کند و لازم نیست شما به صورت دستی اشیا را از حافظه پاک کنید. به عنوان مثال وقتی کاربر یک فایل متنی را برای خواندن باز می کند و آن را نمی بندد، می توان عمل بستن فایل را با استفاده از مخرب انجام داد.دستور نوشتن مخرب به صورت زیر است:

```
public function __destruct()
{
   //code to execute;
}
```

برنامه زیر نحوه فراخوانی سازنده و مخرب را نشان می دهد:

```
<?php

    class Test
    {
        public function Test()
        {
            echo "Constructor was called." . '<br/>';
        }

        public function __destruct()
        {
            echo "Destructor was called.";
        }
    }

    $test = new Test();

?>
```

```
Constructor was called.
Destructor was called.
```

در کلاس **Test** یک سازنده (خطوط **5-8**) و یک مخرب (خطوط **10-13**) تعریف شده است. سپس یک نمونه از کلاس ایجاد کرده ایم (خط **16**). وقتی یک نمونه از کلاس ایجاد می کنیم (خط **16**) سازنده و مخرب فراخوانی شده و پیغام مناسب نمایش داده می شود.

سطح دسترسی

سطح دسترسی مشخص می کند که متدها یک کلاس یا متغیرها در چه جای برنامه قابل دسترسی هستند.در **PHP** سه سطح دسترسی وجود دارد:

- **public** (عمومی)
- **private** (خصوصی)
- **protect** (محافظت شده)

در این درس می خواهیم به سطح دسترسی **private** و **public** نگاهی بیندازیم .سطح دسترسی **public** زمانی مورد استفاده قرار می گیرد که شما بخواهید به یک متد یا متغیر در خارج از کلاس و حتی پروژه دسترسی یابید. به عنوان مثال به کد زیر توجه کنید:

```php
<?php

    class Test
    {
        public  $number1 = 10;
        private $number2 = 20;
    }

    $x = new Test();

    echo $x -> number1;
    echo $x -> number2;

?>
```

```
10
Fatal error: Cannot access private property Test::$number2 in
C:\wamp\www\test.php on line 13
```

در این مثال یک کلاس با نام **Test** تعریف کرده ایم. سپس دو متغیر، یکی به صورت **public** (خط **5**)و دیگری به صورت **private** در داخل کلاس **Test** تعریف می کنیم (خط **6**). در خط **9** یک نمونه از کلاس ایجاد کرده و در خطوط **11** و **12** سعی می کنیم که مقدار متغیرهای آن را چاپ کنیم. همانطور که در خروجی مشاهده می کنید متغیر **number1** که به صورت **public**تعریف شده است قابل دسترسی و متغیر **number2** که به صورت **private** تعریف شده است غیر قابل دسترسی می باشد. به طور کلی متغیرها و متدهایی که به صورت **public** تعریف می شوند در داخل کلاس و نمونه های ایجاد شده از آن قابل دسترسی و متغیرها و متدهایی که به صورت **private** تعریف می شوند فقط در داخل کلاس قابل دسترسی هستند و برای دسترسی به آنها در خارج از کلاس باید از خاصیت ها استفاده کرد که در درس بعد توضیح می دهیم. سطح دسترسی **protect** را بعد از مبحث وراثت در درسهای آینده آموزش می دهیم.

کپسوله سازی

کپسوله کردن(تلفیق داده ها با یکدیگر) یا مخفی کردن اطلاعات فرایندی است که طی آن اطلاعات حساس یک موضوع از دید کاربر مخفی می شود و فقط اطلاعاتی که لازم باشد برای او نشان داده می شود.

وقتی که یک کلاس تعریف می کنیم معمولا تعدادی فیلد برای ذخیره مقادیر مربوط به شی نیز تعریف می کنیم. برخی از این فیلدها توسط خود کلاس برای عملکرد متدها و برخی دیگر از آنها به عنوان یک متغیر موقت به کار می روند .لازم نیست که کاربر به تمام فیلدها یا متدهای کلاس دسترسی داشته باشد .اینکه فیلدها را طوری تعریف کنیم که در خارج از کلاس قابل دسترسی باشند بسیار خطرناک است چون ممکن است کاربر رفتار و نتیجه یک متد را تغییر دهد .به برنامه ساده زیر توجه کنید :

```php
<?php

    class Test
    {
        public $five = 5;

        public function AddFive($number)
        {
            $this -> five += $number;
            return $this -> five;
        }
    }

    $test = new Test;

    $test -> five = 10;
    echo $test -> AddFive(100)

?>
```

```
110
```

متد داخل کلاس **Test** به نام **AddFive** (خطوط 7-11) دارای هدف ساده ای است و آن اضافه کردن مقدار **5** به هر عدد می باشد (همانطور که از اسم متد پیداست). در خط **14** یک نمونه از کلاس **Test** ایجاد کرده ایم و مقدار فیلد آن را در خط **16** از **5** به **10** تغییر می دهیم (در اصل نباید تغییر کند چون ما از برنامه خواسته ایم هر عدد را با **5** جمع کند ولی کاربر به راحتی آن را به **10** تغییر می دهد). همچنین متد **()AddFive** را در خط **17** فراخوانی و مقدار **100** را به آن ارسال می کنیم. مشاهده می کنید که قابلیت متد **()AddFive** به خوبی تغییر می کند و شما نتیجه متفاوتی مشاهده می کنید. اینجاست که اهمیت کپسوله سازی مشخص می شود. اینکه ما در درسهای قبلی فیلدها را به صورت **public** تعریف کردیم و به کاربر اجازه دادیم که در خارج از کلاس به آنها دسترسی داشته باشد کار اشتباهی بود. فیلدها باید همیشه به صورت **private** تعریف شوند.

خواص

property (خصوصیت) استانداردی برای دسترسی به متغیرهایی با سطح دسترسی **private** در داخل یک کلاس می باشد. هر **property** دارای دو بخش می باشد، یک بخش جهت مقدار دهی (بلوک**set**) و یک بخش برای دسترسی به مقدار (بلوک **get**) یک داده **private** می باشد **property** .در مثال زیر نحوه تعریف و استفاده از **property** آمده است:

```
<?php

    class Person
    {
        private $name;

        public function set_name($my_name)
        {
            $this->name = $my_name;
        }

        public function get_name()
        {
            return $this->name;
        }
    }

    $person1 = new Person();

    $person1 -> set_name('Jack');
    echo $person1 -> get_name();

?>
```

```
Jack
```

در برنامه بالا نحوه استفاده از **property** آمده است.همانطور که مشاهده می کنید در این برنامه ما یک خصوصیت تعریف کرده ایم و یک متغیر با سطح دسترسی **private**.

```
private $name;
```

دسترسی به مقدار این متغیر قط از طریق **property** های ارائه شده امکان پذیر است.

```
public function set_name($my_name)
{
    $this->name = $my_name;
}

public function get_name()
```

```
{
    return $this->name;
}
```

در نامگذاری **property** ها به صورت قراردادی ابتدا کلمه **set** یا **get** و سپس علامت زیر و بعد نام متغیر را بنویسید:

```
set_name()
{

}
get_name()
{

}
```

در داخل کلاس دو بخش می بینید ، یکی بخش **set (7-10)** و دیگری بخش **get (12-15)**. بخش **get** ، که با کلمه **get** نشان داده شده است به شما اجازه می دهد که یک مقدار را از متغیرهای **private** استخراج کنید. بخش **set** ، که با کلمه **set**نشان داده شده است برای مقدار دهی به متغیرهای **private** به کار می رود. برای دسترسی به یک خاصیت می توانید از علامت -> استفاده کنید.

```
$person1 -> set_name('Jack');
```

دستور بالا بخش **set** مربوط به **property** را فراخوانی کرده و مقادیری به متغیر اختصاص می دهد. استفاده از **property** ها کد نویسی را انعطاف پذیر می کند مخصوصا اگر بخواهید یک اعتبارسنجی برای اختصاص یک مقدار به متغیر یا استخراج یک مقدار از آن ایجاد کنید. مثلا در مثال زیر شما می توانید یک محدودیت ایجاد کنید که فقط اعداد مثبت به فیلد **age** (سن) اختصاص داده شود .می توانید با تغییر بخش **set** خاصیت **Age** این کار را انجام دهید:

```
1   <?php
2
3       class Person
4       {
5           private $age;
6
7           public function set_age($my_age)
8           {
9               if($my_age > 0)
10              {
11                  $this->age = $my_age;
12              }
13              else
14              {
15                  echo 'Age can not negative or zero!';
16              }
```

```
17          }
18
19          public function get_age()
20          {
21              return $this->age;
22          }
23      }
24
25      $person1 = new Person();
26
27      $person1 -> set_age(12);
28      echo $person1 -> get_age();
29
30  ?>
```

```
12
```

وراثت

وراثت به یک کلاس اجازه می دهد که خصوصیات یا متدهایی را از کلاس دیگر به ارث برد. وراثت مانند رابطه پدر و پسری می ماند به طوریکه فرزند خصوصیاتی از قبیل قیافه و رفتار را از پدر خود به ارث برده باشد.

- **کلاس پایه یا کلاس والد کلاسی است که بقیه کلاسها از آن ارث می برند.**
- **کلاس مشتق یا کلاس فرزند کلاسی است که از کلاس پایه ارث بری می کند.**

همه متد و خصوصیات کلاس پایه می توانند در کلاس مشتق مورد استفاده قرار بگیرند به استثنای اعضا و متدهای با سطح دسترسی . private مفهوم اصلی وراثت در مثال زیر نشان داده شده است:

```php
<?php

    class classParent
    {
        private function privateMessage()
        {
            echo 'This is private Message From Parent Class!';
        }

        public function publicMessage()
        {
            echo 'This is public Message From Parent Class!';
        }
    }

    class classChild extends classParent
    {

    }

    $child = new classChild();

    $child -> publicMessage();
    $child -> privateMessage ();

?>
```

```
This is public Message From Parent Class!
 Fatal error: Call to private method classParent::privateMessage()
```

همانطور که مشاهده می کنید در کد بالا دو کلاس تعریف کرده ایم : یکی کلاس classParent (خطوط 3-14) که دارای دو متد یکی با سطح دسترسی private و دیگری با سطح دسترسی public است و کلاس دیگر (خطوط 16-19) که در بدنه خود هیچ متد یا متغیری ندارد. نحوه ارث بری یک کلاس به صورت زیر است:

```
classChild extends Parent
```

که در خط 16 مشخص کرده ایم که کلاس classChild قرار است از کلاس classParent ارث بری کند:

```
class classChild extends classParent
```

در خط 21 یک نمونه از کلاس فرزند ایجاد می کنیم و در خطوط 23 و 24 دو متد کلاس پدر ر فراخوانی می کنیم. همانطور که در خروجی مشاهده می کنید متدی که دارای سطح دسترسی public است فراخوانی و اجرا شده ولی در فراخوانی متدی با سطح دسترسی private با خطا مواجه می شویم. در پایان یاد آور می شویم که کلاس فرزند (خطوط 19-16) هیچ متد یا متغیری در بدنه خود ندارد ولی چون از کلاس classParent ارث بری کرده است متد با سطح دسترسی public آن را می تواند برای خود داشته باشد .

سطح دسترسی Protect

سطح دسترسی **protect** اجازه می دهد که اعضای کلاس، فقط در کلاسهای مشتق شده از کلاس پایه قابل دسترسی باشند. بدیهی است که خود کلاس پایه هم می تواند به این اعضا دسترسی داشته باشد. کلاسهایی که از کلاس پایه ارث بری نکرده اند نمی توانند به اعضای با سطح دسترسی **protect** یابند. در مورد سطوح دسترسی **public** و **private** قبلا توضیح دادیم. در جدول زیر نحوه دسترسی به سه سطح ذکر شده نشان داده شده است:

قابل دسترسی در	public	private	protected
داخل کلاس	true	true	true
خارج از کلاس	true	false	false
کلاس مشتق	true	false	true

مشاهده می کنید که **public** بیشترین سطح دسترسی را داراست. صرف نظر از مکان، اعضای **public** در هر جا فراخوانی می شوند و قابل دسترسی هستند. اعضای **private** فقط در داخل کلاسی که به آن تعلق دارند قابل دسترسی هستند. کد زیر رفتار اعضای دارای این سه سطح دسترسی را نشان می دهد:

```php
<?php

    class classParent
    {
        protected $protectedMember = 10;
        private   $privateMember   = 20;
        public    $publicMember    = 30;
    }

    class classChild extends classParent
    {
        public function __construct ()
        {
            echo $this -> publicMember    .'<br/>';
            echo $this -> protectedMember .'<br/>';
            echo $this -> privateMember;
        }
    }

    $child = new classChild();

?>
```

```
10
2
(!) Notice: Undefined property: classChild::$privateMember
```

کدهایی که با خط قرمز نشان داده شده اند نشان دهنده وجود خطا است .همانطور که در خط 16 مشاهده می کنید کلاس classChild سعی می کند که به عضو private کلاس classParent دسترسی یابد. از آنجاییکه اعضای private در خارج از کلاس قابل دسترسی نیستند، حتی کلاس مشتق در خط 16 نیز ایجاد خطا می کند .اگر شما به خط ۱۴ توجه کنید کلاس classChild می تواند به عضو protect کلاس classParent دسترسی یابد چون کلاس classChild از کلاس classParent مشتق شده است. حال کد زیر را در نظر بگیرید:

```
1   <?php
2
3       class classParent
4       {
5           public    $publicMember    = 10;
6           protected $protectedMember = 20;
7           private   $privateMember   = 30;
8       }
9
10      $parent =new classParent();
11
12      echo $parent -> publicMember;
13      echo $parent -> protectedMember;
14      echo $parent -> privateMember;
15
16  ?>
```

```
10
(!) Fatal error: Cannot access protected property
classParent::$protectedMember
```

همانطور که در کد بالا مشاهده می کنید از آنجاییکه اعضای protected و private در خارج از کلاس قابل دسترسی نیستند خطوط 13 و 14 ایجاد خطا می کنند.

trait

یکی از مشکلات **PHP** این است که از وراثت چندگانه پشتیبانی نمی کند یعنی هر کلاس فقط می تواند از یک کلاس ارث بری کند. بعضی مواقع ارث بری از چند کلاس می تواند باعث کاهش کدنویسی شود. در **PHP 5.4** قابلیتی با نام **trait** اضافه شده است که این مشکل را برطرف می کند **Trait** .گروهی از متدها هستند که می توان در داخل کلاس ها از آنها استفاده کرد. یا به عبارت دیگر می توان چندین متد در که قرار است در داخل چند کلاس مورد استفاده قرار گیرند را در داخل یک **trait** قرار داد. این کار باعث کاهش کدنویسی شده و مشکل وراثت چندگانه را برطرف می کند. از **trait** ها همانند کلاس های انتزاعی نمی توان نمونه ایجاد کرد. برای ایجاد یک **trait** به صورت زیر عمل می شود:

```
trait traitname
{
    // code here...
}
```

همانگونه که در کد بالا مشاهده می کنید برای ایجاد **trait** از کلمه کلیدی **trait** استفاده می شود و سپس یک نام برای آن انتخاب می کنیم. برای روشن شدن نحوه استفاده از **trait** ها به مثال زیر توجه کنید:

```
1   <?php
2       trait PrintFunctionality
3       {
4           public function FirstPrint()
5           {
6               echo 'This is FristPrint !<br/>';
7           }
8
9           public function SecondPrint()
10          {
11              echo 'This is SecondPrint !<br/>';
12          }
13      }
14
15      class firstClass
16      {
17          use PrintFunctionality;
18      }
19
20      class secondClass
21      {
22          use PrintFunctionality;
23      }
24
25      $firstClass  = new firstClass();
26      $secondClass = new secondClass();
27
28      $firstClass -> FirstPrint();
```

```
29      echo '<br/>';
30      $secondClass -> FirstPrint();
31      $secondClass -> SecondPrint();
32  ?>
```

```
This is FristPrint !

This is FristPrint !
This is SecondPrint !
```

در کد بالا یک trait تعریف کرده ایم (خطوط 13-2). در داخل این trait دو متد با نام های FirstPrint() و SecondPrint() قرار داده ایم. حال می خواهیم از این متدها در داخل دو کلاس firstClass و secondClass استفاده کنیم .برای اینکار همانطور که مشاهده می کنید از کلمه کلیدی use و سپس نام trait در داخل کلاس بهره می بریم. سپس با ایجاد شیء از دو کلاس می توانیم از هر متدی که دوست داشتیم استفاده کنیم.

فضای نام

فضای نام راهی برای دسته بندی کدهای برنامه می باشد. هر چیز در **PHP** حداقل در یک فضای نام قرار دارد. وقتی برای یک کلاس اسمی انتخاب می کنید ممکن است برنامه نویسان دیگر به صورت اتفاقی اسمی شبیه به آن برای کلاسشان انتخاب کنند. وقتی شما از آن کلاسها در برنامه تان استفاده کنید از آنجاییکه از کلاسهای همنام استفاده می کنید در برنامه ممکن است خطا به وجود آید.

فضاهای نامی از وقوع این خطاها جلوگیری کرده یا آنها را کاهش می دهند .هنگامی که یک پروژه جدید ایجاد می کنید به صورت پیشفرض یک فضای نام برای شما ایجاد خواهد شد. این فضای نام ، فضای نام عمومی یا **global** است و نوشتن آن در صورتیکه نخواهید از فضای نام در برنامه استفاده کنید الزامی نیست .برای روشن شدن مطلب دو کد زیر شبیه به هم هستند:

```
<?php
    class Sample
    {
        public function sayHello()
        {
            echo ("Hello World!");
        }

    }
?>
-----------------------------------------------
<?php
    namespace
    {
        class Sample
        {
            public function sayHello()
            {
                echo ("Hello World!");
            }

        }
    }
?>
```

همانطور که در کد بالا مشاهده می کنید برای تعریف فضای نام از کلمه کلیدی **namespace** استفاده می شود و سپس یک نام به آن اختصاص می دهیم. به مثال زیر توجه کنید:

```
namespace Test
{
    ...
}
```

البته به این نکته توجه کنید که وقتی می خواهیم از فضای نام عمومی استفاده کنیم لازم نیست نامی به آن اختصاص دهیم. برای روشن شدن اینکه چرا استفاده از فضای نام مفید است به کد زیر توجه کنید:

```php
<?php
     namespace FirstNamespace
     {
         class Sample
         {
             public function ShowMessage()
             {
                 echo ("Hello World !");
             }

         }
     }

     namespace SecondNamespace
     {
         class Sample
         {
             public function ShowMessage()
             {
                 echo ("Hello IRAN !");
             }
         }
     }

     namespace
     {
         $Sample = new SecondNamespace\Sample();

         $Sample->ShowMessage();
     }
?>
```

```
Hello IRAN !
```

همانطور که در کد بالا مشاهده می کنید دو فضای نام به نام های **FirstNamespace**(خطوط 2-12) و **SecondNamespace** (خطوط 14-23) ایجاد کرده ایم. در داخل این دو فضای نام دو کلاس و دو متد با نام یکسان وجود دارد که تفاوت آنها فقط در نوع پیامی است که متد **ShowMessage()** چاپ می کند. حال فرض کنید که می خواهیم متد موجود در کلاس **Sample** که در فضای نام **SecondNamespace** را فراخوانی کنیم. برای این کار ابتدا یک فضای نام عمومی ایجاد کرده و در داخل آن یک نمونه از کلاس **Sample** ایجاد می کنیم. وای به یاد داشته باشید که قبل از نام کلاس باید نام فضای نام و سپس یک یک اسلش قرار دهید:

```php
$Sample = new SecondNamespace\Sample();
```

به همین راحتی می توانیم با استفاده از فضای نام متدها و کلاس های همنامی در برنامه داشته باشیم.

Overriding

متدهای مجازی متدهایی از کلاس پایه هستند که می توان به وسیله یک متد از کلاس مشتق آنها را **override** کرده و به صورت دلخواه پیاده سازی نمود. به عنوان مثال شما متد **A** را در کلاس **A** دارید و کلاس **B** از کلاس **A** ارث بری می کند، در این صورت متد **A** در کلاس **B** در دسترس خواهد بود. اما متد **A**دقیق همان متدی است که از کلاس **A** به ارث برده شده است. حال اگر بخواهید که این متد رفتار متفاوتی از خود نشان دهد چکار می کنید؟ **Overriding** یا بازنویسی این مشکل را برطرف می کند. به تکه کد زیر توجه کنید:

```php
<?php
    class Person
    {
        public function ShowMessage()
        {
            echo 'Message from Parent.';
        }
    }

    class Child extends Person
    {
        public function ShowMessage()
        {
            parent::ShowMessage();
            echo '<br/>ShowMessage method was overrided !' ;
        }
    }

    $myPerson = new Person();
    $myChild = new Child();

    $myPerson -> ShowMessage();
    echo '<br/><br/>';
    $myChild  -> ShowMessage();
?>
```

```
Message from Parent.

Message from Parent.
ShowMessage method was overrided !
```

همانطور که در کد بالا مشاهده می کنید یک متد به نام **ShowMessage()** (خطوط 4-7) در کلاس **Person** تعریف شده است که یک پیغام چاپ می کند. حال می خواهیم این متد در کلاس **Child** علاوه بر این پیغام پیغام **ShowMessage method!** **was overrided**را نیز چاپ کند. برای این کار همانطور که مشاهده می کنید همین متد را در خطوط (12-16) و در داخل کلاس **Child** می نویسیم و سپس با استفاده از کلمه کلیدی **parent** و سپس دو نقطه (::) در خط **14** به **PHP** اعلام می کنیم که قصد استفاده از تمام کدهای بدنه همین متد در کلاس مادر را داریم بعلاوه اینکه در خط بعد از این دستور یعنی خط **15** کدهای اضافی را که قرار است این متد در کلاس فرزند یعنی **Child** داشته باشد می نویسیم. شاید این کار برای متدی به این

سادگی زیاد کارا نباشد، اما اگر متد کلاس مادر دارای کدهای زیادی در بدنه خود باشد و شما بخواهید کد دیگری در کلاس فرزند به آن اضافه کنید استفاده از این روش کدنویسی را بهینه و ساده تر می کند.

کلاسهای انتزاعی

کلاسهای مجرد **(abstract)** کلاسهایی هستند که کلاس پایه سایر کلاسها هستند.این نوع کلاسها می توانند مانند کلاسهای عادی دارای سازنده باشند. شما نمی توانید از کلاسهای انتزاعی نمونه ایجاد کنید چون که هدف اصلی از به کار بردن کلاسهای انتزاعی استفاده از آنها به عنوان کلاس پایه برای کلاسهای مشتق است. برای تعریف یک کلاس انتزاعی از کلمه کلیدی **abstract** استفاده می شود.

```
abstract class className
{

}
```

به مثال زیر در مورد استفاده از کلاسهای انتزاعی توجه کنید:

```
1   <?php
2
3         abstract class Base
4         {
5               protected abstract function ShowMessage();
6         }
7
8         class firstChild extends Base
9         {
10              protected function ShowMessage()
11              {
12              echo 'Hello World!';
13              }
14        }
15
16        class secondChild extends Base
17        {
18              public function ShowMessage()
19              {
20              echo 'Hello IRAN!';
21              }
22        }
23
24  ?>
```

همانطور که در کد بالا مشاهده می کنید یک کلاس **abstract** (خطوط 6-3) که دارای یک مت **abstract** (خط 5) هم هست را تعریف کرده ایم. متد **abstract** فاقد بدنه است. این متدها توسط کلاس هایی که از کلاس **abstract** مشتق می شوند، پیاده سازی می شوند (اعضایی که با کلمه کلیدی **abstract** مشخص می‌شوند باید توسط کلاس های مشتق شده پیاده سازی شوند.). در کد بالا دو کلاس **firstChild** (خطوط 14-8) و **secondChild** (خطوط 22-16) تعریف کرده ایم که به وسیله کلمه کلیدی

extends از کلاس Base ارث بری کرده اند. هنگام ارث‌بری از یک کلاس انتزاعی تمام متدها انتزاعی باید در کلاس فرزند نیز تعریف شده باشند. علاوه بر این متدها باید دارای همان سطح دسترسی که در کلاس پایه تعریف شده است و یا سطح دسترسی بیشتری باشند. برای مثال کلاس پایه در کد بالا دارای متدی با سطح دسترسی protected است (خط 5)، حال ما این متد را یک بار در خط 10 به صورت protected و بار دیگر در خط 18 به صورت public تعریف کرده ایم و مشکلی پیش نیامده است. حال فرض کنید که سطح دسترسی این متد در کلاس پایه publicباشد. در این صورت در کلاس های فرزند این متد فقط باید به صورت public تعریف شود، چون سطح دسترسی protected و private نسبت به public کمتر است .نمی توان از یک کلاس abstract نمونه ایجاد کرد ولی از کلاس هایی که از این نوع کلاس ها مشتق می شوند، می توان نمونه ایجاد کرد.

کلاس final و متد final

کلاس **final** کلاس نهایی)، کلاسی است که دیگر کلاس ها نمی توانند از آن ارث بری کنند و چون قابلیت ارث بری ندارد نمی تواند مجرد **(abstract)**هم باشد. مثال زیر یک کلاس **final** را نشان می دهد:

```
<?php
    final class Base
    {
       public $someField;

       public function SomeMethod()
       {
          //Do something here
       }

    }

    class Derived extends Base
    {
       //This class cannot inherit the Base class
    }
?>
```

```
( ! ) Fatal error: Class Derived may not inherit from final class (Base)
```

برای تعریف این کلاس ها از کلمه کلیدی **final** استفاده می شود. مشاهده می کنید که کلاس نهایی مانند کلاس های عادی، دارای فیلد، خواص، و متد می باشند. کلاس مشتق **(Derived)** در مثال بالا نمی تواند از کلاس نهایی **(Base)** ارث بری کند. وقتی یک کلاس را نهایی می کنیم، تمام متدهای آن نیز نهایی می شوند. استفاده از این کلاسها همانطور که ذکر شد زمانی مفید است که بخواهید کلاسی ایجاد کنید که دیگر کلاسها نتوانند از آن ارث بری کنند.

متد Final

متد **final** به متدی گفته می شود که هیچ زیر کلاسی نتواند آن را بازنویسی یا **Override** کند. به مثال زیر توجه کنید:

```
<?php
    class Base
    {
           final function ShowMessage()
        {
            echo "This is a final method!" ;
        }
    }
```

```
    class Derived extends Base
    {
        function ShowMessage()
        {
            parent::ShowMessage();
        }
    }
?>
```

```
( ! ) Fatal error: Cannot override final method Base::ShowMessage()
```

اگر کدهای بالا را اجرا کنید، مشاهده می کنید که خطا به وجود می آید .چون کلاس Child از کلاس Parent ارث بری کرده است و زیر کلاس محسوب می شود و طبق تعریف هیچ زیر کلاسی نمی تواند متدهای final را بازنویسی کند.

اعضای Static

اگر بخواهیم عضو داده ای (فیلد) یا خاصیتی ایجاد کنیم که در همه نمونه های کلاس قابل دسترسی باشد از کلمه کلیدی static استفاده می کنیم. کلمه کلیدی static برای اعضای داده ای و خاصیت هایی به کار می رود که می خواهند در همه نمونه های کلاس تقسیم شوند. وقتی که یک متد یا خاصیت به صورت static تعریف شود، می توانید آنها را بدون ساختن نمونه ای از شی، فراخوانی کنید.برای فراخوانی یک عضو استاتیک ابتدا نام کلاس سپس علامت دو نقطه (::) و در آخر نام عضو استاتیک را می نویسید:

```
Class Name :: Static Member
```

به مثالی در مورد متدها و خاصیتهای static توجه کنید:

```
1   <?php
2
3       class SampleClass
4       {
5           static $number = 10;
6
7           static function PrintNumber()
8           {
9               echo self::$number;
10          }
11
12          static function ShowStaticMessage()
13          {
14              echo 'This is a Static Function!';
15          }
16      }
17
18      SampleClass::PrintNumber ();
19      echo '<br/>';
20      SampleClass::ShowStaticMessage ();
21
22  ?>
```

```
This is a Static Function!
10
```

همانگونه که در کد بالا مشاهده می کنید یک کلاس (خطوط 16-3) تعریف کرده ایم که دارای یک متغیر (خط 5) و دو متد (خطوط 10-7 و خطوط 15-12) از نوع استاتیک می باشد. برای دسترسی به متد یا متغیرهای استاتیک به دو صورت عمل می کنیم:

اگر بخواهیم به یک متغیر استاتیک در داخل یک متد استاتیک دسترسی داشته باشیم به جای استفاده از کلمه کلیدی this از کلمه کلیدی self و به جای علامت ->از علامت :: استفاده می کنیم. مثلا در خط 9 مثال بالا برای چاپ متغیر number که استاتیک است در داخل متد استاتیک PrintNumber به روشی که ذکر شد عمل کرده ایم.

اما اگر بخواهیم به یکی از اعضای استایک کلاس در خارج از کلاس دسترسی داشته باشیم لازم نیست که از کلاس نمونه ایجاد کنیم و فقط کافیست که نام کلاس را نوشته و بعد از آن علامت دو نقطه و در آخر نام عضو استاتیک را بنویسید مانند خطوط 18 و 20 مثل بالا.

ثابت های کلاس

همانطور که قبلا اشاره شد، ثابت ها انواعی از متغیرها هستند که مقدار آنها در طول برنامه تغییر نمی کند. ثابت ها حتما باید مقدار دهی اولیه شوند و اگر مقدار دهی آنها فراموش شود در برنامه خطا به وجود می آید. بعد از این که به ثابت ها مقدار اولیه اختصاص داده شد هرگز در زمان اجرای برنامه نمی توان آن را تغییر داد. ثابت های کلاس هم همانند ثابت های معمولی هستند با این تفاوت که در داخل کلاس تعریف می شوند و فقط در داخل آن کلاس، نمونه های ایجاد شده از کلاس و یا کلاس های مشتق قابل دسترسی هستند .برای دسترسی به ثابت های کلاس، همانند اعضای استاتیک، از عملگر دو نقطه (::) استفاده می شود. به مثال زیر توجه کنید:

```
<?php
    class Person
    {
        const name = 'Jack';
    }

    echo Person::name;
?>
```

```
Jack
```

عملگر instanceof

عملگر **instanceof** در **PHP** به شما اجازه می دهد که تست کنید که آیا یک شی یک نمونه از یک نوع خاص (کلاس، زیر کلاس، ...) است یا نه. عملگر **instanceof**به دو عملوند نیاز دارد و یک مقدار بولی را برمی گرداند. به عنوان مثال، فرض کنید یک کلاس به نام **Animel** داریم، سپس یک نمونه از آن ایجاد می کنیم:

```
<?php
    class Animal
    {

    }

    $myAnimal = new Animal();

    if ($myAnimal instanceof Animal)
    {
        echo "myAnimal is an Animal!" ;
    }
?>
```

```
myAnimal is an Animal!
```

رفتار عملگر **instanceof** را در این مثال مشاهده کردید. همانطور که می بینید از آن به عنوان شرط در عبارت **if** استفاده شده است. کاربرد آن در مثال بالا این است که چک می کند که آیا شی **myAnimal** یک نمونه از **Animal** است و چون نتیجه درست است کدهای داخل دستور **if** اجرا می شود. این عملگر همچنین می تواند چک کند که آیا یک شی خاص در سلسله مراتب وراثت یک نوع خاص است. به این مثال توجه کنید:

```
<?php
    class Animal
    {

    }

    class Dog extends Animal
    {

    }

    $mydog = new Dog();

    if ($mydog instanceof Animal)
    {
        echo "myAnimal is an Animal!" ;
    }
?>
```

```
myDog is an Animal!
```

همانطور که در مثال بالا می بینید ما یک کلاس به نام **Dog** ایجاد کرده ایم که از کلاس **Animal** ارث می برد. سپس یک نمونه از این کلاس **(Dog)** ایجاد می کنیم و سپس با استفاده از عملگر **instanceof** تست می کنیم که آیا نمونه ایجاد شده جز کلاس **Animal**است یا یک کلاس مشتق شده از کلاس **Animal** می باشد. از آنجاییکه کلاس **Dog** از کلاس **Animal** ارث می برد (سگ من یک حیوان است.)، نتیجه عبارت درست **(true)** است. حال جمله بالا را تغییر دهیم : " حیوان من یک سگ است" . وقتی جمله برعکس می شود چه اتفاقی می افتد؟

```php
$myAnimal = new Animal();

if ($myAnimal instanceof Dog)
{
    echo "myAnimal is a Dog!" ;
}
```

این باعث خطا نمی شود و عبارت فقط نتیجه **false** را بر می گرداند. می توان از کد بالا این را درک کرد که همه حیوانات سگ نیستند ولی همه سگها حیوان هستند. یکی دیگر از روش های تشخیص نام کلاسی که یک شیء از آن مشتق شده است استفاده از متد **get_Class()** است. به کد زیر توجه کنید:

```php
<?php
    class Animal
    {

    }

    $myAnimal = new Animal();
    echo get_Class($myAnimal);
?>
```

```
Animal
```

چند ریختی

چند ریختی (Polymorphism) به کلاسهایی که در یک سلسله مراتب وراثتی مشابه هستند اجازه تغییر شکل و سازگاری مناسب می دهد و همچنین به برنامه نویس این امکان را می دهد که به جای ایجاد برنامه های خاص، برنامه های کلی و عمومی تری ایجاد کند. . به عنوان مثال در دنیای واقعی همه حیوانات غذا می خورند، اما روش های غذا خوردن آنها متفاوت است. در یک برنامه برای مثال ، یک کلاس به نام Animal ایجاد می کنید. بعد از ایجاد این کلاس می توانید آن را چند ریخت (تبدیل) به کلاس Bird کنید و متد Fly() را فراخوانی کنید. به مثالی در باره چند ریختی توجه کنید:

```php
<?php

    class Animal
    {
        public function Eat()
        {
            echo 'The animal ate!' .'<br/>';
        }
    }

    class Dog extends Animal
    {
        function Eat()
        {
            echo 'The dog ate!' .'<br/>';
        }
    }


    class Bird extends Animal
    {
        function Eat()
        {
            echo 'The bird ate!' .'<br/>';
        }
    }

    class Fish extends Animal
    {
        function Eat()
        {
            echo 'The fish ate!' .'<br/>';
        }
    }


    $myDog    = new Dog();
    $myBird   = new Bird();
```

```
39      $myFish   = new Fish();
40      $myAnimal = new Animal();
41
42      $myAnimal ->Eat();
43
44      $myAnimal = $myDog;
45      $myAnimal ->Eat();
46
47      $myAnimal = $myBird;
48      $myAnimal ->Eat();
49
50      $myAnimal = $myFish;
51      $myAnimal ->Eat();
52
53  ?>
```

```
The animal ate!
The dog ate!
The bird ate!
The fish ate!
```

همانطور که مشاهده می کنید ۴ کلاس مختلف تعریف کرده ایم **Animal** .کلاس پایه است و سه کلاس دیگر از آن مشتق می شوند. هر کلاس متد **()Eat** مربوط به خود را دارد. نمونه ای از هر کلاس ایجاد کرده ایم (**40-37**). حال متد **()Eat** را به وسیله نمونه ایجاد شده از کلاس **Animal** به صورت زیر فراخوانی می کنیم:

```
$myAnimal = new Animal();
$myAnimal ->Eat();
```

در مرحله بعد چندریختی روی می دهد. همانطور که در مثال بالا مشاهده می کنید شی **Dog** را برابر نمونه ایجاد شده از کلاس **Animal**قرار می دهیم (خط **44**) و متد **()Eat** را بار دیگر فراخوانی می کنیم. حال با وجود اینکه ما از نمونه کلاس **Animal** استفاده کرده ایم ولی متد **()Eat** کلاس **Dog** فراخوانی می شود. این به دلیل تاثیر چند ریختی است.

سپس دو شی دیگر (**Bird** و **Fish**) را برابر نمونه ایجاد شده از کلاس **Animal** قرار می دهیم و متد **()Eat** مربوط به هر یک را فراخوانی می کنیم.(خطوط **51-47**) به این نکته توجه کنید که وقتی در مثال بالا اشیاء را برابر نمونه کلاس **Animal** قرار می دهیم ازعمل **Cast** استفاده نکرده ایم چون این کار **(cast)** وقتی که بخواهیم یک شی از کلاس مشتق (مثلا**Dog**) را در شیی از کلاس پایه **(Animal)** ذخیره کنیم لازم نیست.

ثابت های جادویی

همانطور که در درس های قبلی گفتیم، ثابت ها در زبان php با استفاده از const و define() ایجاد می شوند. به صورت پیشفرض یک سری ثابت ها در php تعریف شده که به نام ثابت های جادویی (Magic Constant) معروف هستند و ما می توانیم در هر جای پروژه از مقادیر آنها استفاده کنیم.مشخصه ای که این ثابت ها دارند این است که قبل و بعد از نام آنها دو علامت زیر خط (__)وجود دارد. در جدول زیر لیست این ثابت های جادویی آورده شده است:

ثابت جادویی	کاربرد
__LINE__	شماره ی خط جاری رو برمی گرداند
__FILE__	آدرس فایل جاری به همراه نام فایل را برمی گرداند
__DIR__	آدرس فایل جاری را بدون نام فایل برمی گرداند
__FUNCTION__	نام تابعی که در آن حضور دارد را برمی گرداند
__CLASS__	نام کلاس جاری را برمی گرداند
__TRAIT__	نام trait جاری را برمی گرداند
__METHOD__	نام متد جاری در داخل کلاس را برمی گرداند
__NAMESPACE__	نام namespace جاری را برمی گرداند

مثال زیر نحوه عملکرد این ثابت ها را نشان می دهد:

```php
<?php
    namespace MagicNamespace
    {
        class MagicClass
        {
            public function __construct()
            {
                echo  __LINE__      .'<br/>';
                echo  __FILE__      .'<br/>';
                echo  __DIR__       .'<br/>';
                echo  __FUNCTION__  .'<br/>';
                echo  __CLASS__     .'<br/>';
                echo  __METHOD__    .'<br/>';
                echo  __NAMESPACE__ .'<br/>';
            }
        }

        $magicClass = new MagicClass();
    }
```

```
20 ?>
```

```
8
C:\wamp\www\Tutorials\index.php
C:\wamp\www\Tutorials
__construct
MagicNamespace\MagicClass
MagicNamespace\MagicClass::__construct
MagicNamespace
```

همانطور که در کد بالا مشاهده می کنید وجود مقلا ثابت جادویی __LINE__ در خط 8 باعث چاپ عدد 8 و وجود ثابت جادویی __CLASS__ در خط 21 و در داخل کلاس MagicClass (خط 17) باعث چاپ نام این کلاس می شود.البته اگر کلاس خود در داخل یک فضای نام باشد ابتدا نام فضای نام و سپس کلاس را چاپ می کند. پس وجود این ثابت ها در اسکریپت های مختلف و بسته به مکانی که در آن قرار دارند باعث چاپ نتایج مختلفی می شود.

متدهای جادویی (Magic Methods)

در PHP تعدادی متد با عنوان متدهای جادویی (Magic Method) وجود دارد که نام آنها با دو زیرخط آغاز می شود و در هنگام وقوع رخداد های خاصی بصورت اتوماتیک فراخوانی و اجرا می شوند. در زیر نام این متدها ذکر شده است:

- __construct()
- __destruct()
- __call()
- __callStatic()
- __get()
- __set()
- __isset()
- __unset()
- __toString()
- __invoke()
- __sleep()
- __wakeup()
- __set_state()
- __clone()
- __autoload()

__construct()

سازنده ها متدهای خاصی هستند که وجود آنها برای ساخت اشیا لازم است .آنها به شما اجازه می دهند که متغیرهای کلاس را مقداردهی اولیه کنید و کدهایی که را که می خواهید هنگام ایجاد یک شی اجرا شوند را به برنامه اضافه کنید. اگر از هیچ سازنده ای در کلاس تان استفاده نکنید، PHP از سازنده پیشفرض که یک متد بدون پارامتر است استفاده می کند. قبلا ذکر شد که نام کلاس و نام سازنده باید دقیقا عین هم باشد ولی با وجود این متد جادویی دیگر تشابه اسمی لازم نیست:

```php
<?php
    class Person
    {
        public function __construct()
        {
            echo 'Constructor was called.';
        }
    }

    $person = new Person();
?>
```

```
Constructor was called.
```

همانطور که در کد بالا مشاهده می کنید به محض ایجاد شی از کلاس سازنده فراخوانی شده و کد درون آن اجرا می شود.

__destruct()

مخرب ها نقطه مقابل سازنده ها هستند. مخرب ها متدهای خاصی هستند که هنگام تخریب یک شی فراخوانی می شوند. اشیا از حافظه کامپیوتر استفاده می کنند و اگر پاک نشوند ممکن است با کمبود حافظه مواجه شوید. می توان از مخرب ها برای پاک کردن منابعی که در برنامه مورد استفاده قرار نمی گیرند استفاده کرد. معمولا PHP به صورت اتوماتیک از زباله روب (garbage collection) برای پاک کردن حافظه استفاده می کند و لازم نیست شما به صورت دستی اشیا را از حافظه پاک کنید. به عنوان مثال وقتی کاربر یک فایل متنی را برای خواندن باز می کند و آن را نمی بندد، می توان عمل بستن فایل را با استفاده از مخرب انجام داد.

```php
<?php
    class Person
    {
        public function __destruct()
        {
            echo 'Destructor was called.';
        }
    }

    $person = new Person();
?>
```

```
Destructor was called.
```

__set() و __get()

برای درک بهتر عملکرد این دو متد توصیه می کنم که ابتدا درس (خواص) را بخوانید. در کل این دو متد باعث ایجاد و مقداردهی خاصیت های کلاس می شوند و کدنویسی را به شدت کاهش می دهند. به مثال زیر توجه کنید:

```php
<?php
    class Person
    {
        private $info = array() ;

        function __set($name , $value)
        {
            $this->info[$name] = $value ;
        }

        function __get($name)
```

```
        {
            return $this->info[$name];
        }
    }

    $person = new Person();

    //set values
    $person -> name   = 'Leo' ;
    $person -> family = 'Messi';
    $person -> Age    = 29;

    // Get values
    echo $person -> name ;
?>
```

```
Leo
```

همانطور که در کد بالا مشاهده می کنید به جای تعریف چند خاصیت برای کلاس یک آرایه به نام **info** تعریف کرده ایم و سپس خاصیت ها را با متد **set__** مقداردهی و سپس با متد **get__** چاپ کرده ایم.

__invoke()

این متد به یک شیء اجازه می دهد که مانند یک متد عمل کند. به مثال زیر توجه کنید:

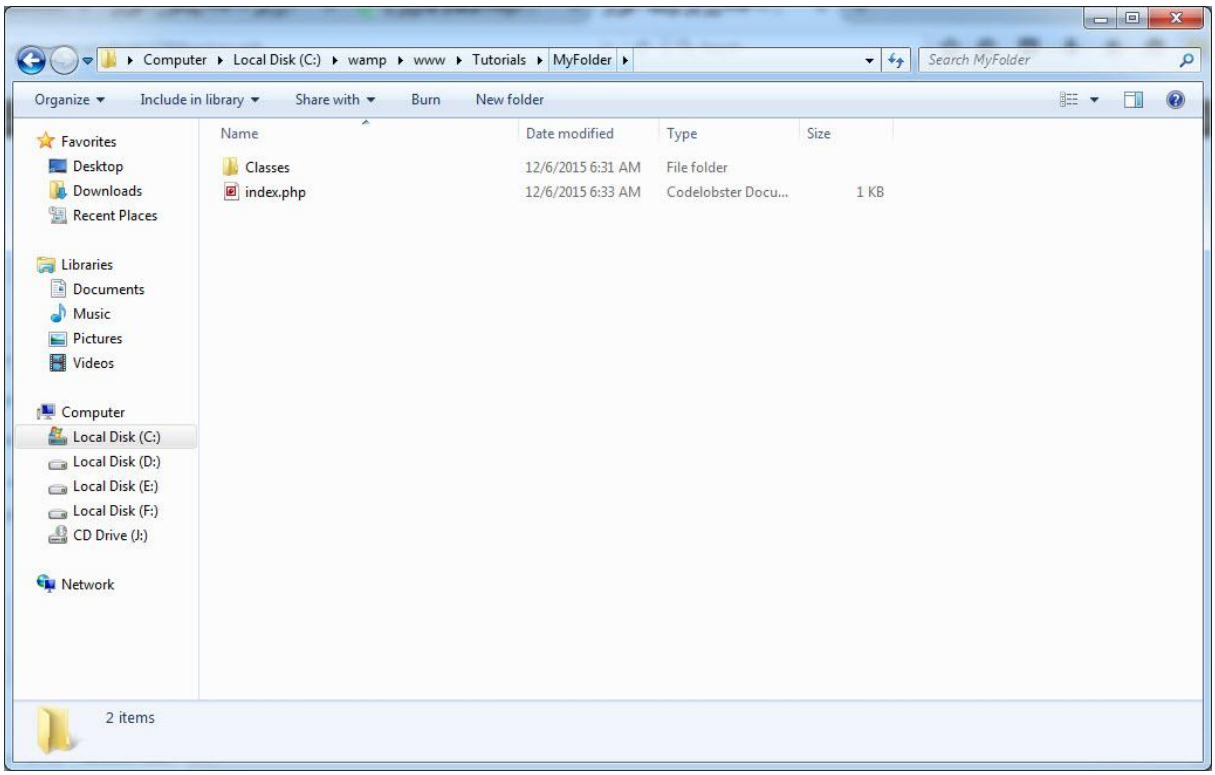
```
<?php
    class Person
    {
        public function __invoke($param)
        {
            echo 'Hello '.$param;
        }
    }

    $person = new Person();
    $person('World !');
?>
```

```
Hello World !
```

همانطور که در کد بالا مشاهده می کنید بعد از ایجاد شیء می توانیم با قرار دادن دو پرانتز در جلوی آن ، متد **invoke__** را فراخوانی کرد.

__autoload()

از این متد در فراخوانی خودکار کلاس هایی که از روی آنها شیء ایجاد کرده ایم به کار می رود. همانطور که می دانید برای استفاده از کلاس هایی که در فایل هستند ، در فایل دیگر ، باید آنها را **include** کرد. حال اگر تعداد کلاسها زیاد باشد ، **include**کردن آنها هم خسته کننده است و هم ممکن است باعث بروز خطا شود. برای جلوگیری از تکرار و خطا از متد جادویی **__autoload**می کنیم. فرض کنید که یک فایل به نام **index.php** و یک پوشه که در داخل آن دو کلاس به نام های **firstClass.php**و **secondClass.php** داریم.



محتویات دو کلاسی که در داخل پوشه **Classes** قرار دارند به صورت زیر می باشد:

```php
<?php
    class firstClass
    {
        public function ShowMessage()
        {
            echo 'Hello World!';
        }
    }
?>
```

```php
<?php
    class secondClass
    {
        public function ShowMessage()
        {
            echo 'Hello IRAN!';
        }
    }
?>
```

به یک نکته خیلی مهم توجه کنید و آن این است که نام کلاس باید با نام فایلی که در آن قرار دارد دقیقا مشابه هم باشد. مثلا اگر نام فایل **firstClass.php** است نام کلاسی که در داخل این فایل وجود دارد باید **firstClass** و اگر نام فایل **secondClass.php** است نام کلاس داخل آن باید **secondClass** باشد . حال برای فراخوانی این کلاس ها در فایل **index.php**، فایل مذکور را باز کرده و کدهای زیر را در داخل ان بنویسید:

```php
<?php
    function __autoload($classname)
    {
        require_once 'Classes/'.$classname.'.php';
    }


    $firstclass  = new firstClass();
    $firstclass -> ShowMessage();

    echo '<br />';

    $secondClass  = new secondClass();
    $secondClass -> ShowMessage();
?>
```

```
Hello World!
Hello IRAN!
```

در کد بالا بعد از اینکه یک شی از یک کلاس می سازیم ، نام کلاس جایگزین پارامتر تابع **autoload** می شود. مثلا با ایجاد شی از **firstClass** کد بالا به صورت زیر در می آید:

```php
<?php
    function __autoload(firstClass)
    {
        require_once 'Classes/'.firstClass.'.php';
    }
```

```
    $firstclass  = new firstClass();
    $firstclass -> ShowMessage();
?>
```

در نتیجه به راحتی متدهای این کلاس قابل دسترسی می شوند.در مورد سایر متدهای جادویی در درس های آینده توضیح می دهیم.

آرایه های فوق سراسری (super globals)

آرایه های فوق سراسری اولین بار در نسخه ۴,۱ پی اچ پی ارائه شد. این آرایه ها برای در دسترس بودن در تمام بخش ها ساخته شده اند .چندین آرایه superglobal از پیش تعریف شده در PHP وجود دارد. بدین معنی که بدون در نظر گرفتن بخش، تابع و کلاس و یا هر فایلی قابل استفاده هستند. این آرایه های فوق سراسری عبارتند از:

- **$GLOBALS**
- **$_SERVER**
- **$_REQUEST**
- **$_POST**
- **$_GET**
- **$_FILES**
- **$_COOKIE**
- **$_SESSION**

$GLOBALS

در PHP در php متغیرهای سراسری در آرایه ای با نام $GLOBALS ذخیره می شوند که به این صورت نیز می توانیم به مقادیر آنها دسترسی داشته باشیم:

```
<?php

    $firstNumber  = 10;
    $secondNember = 5;
    $Sum;

    function GlobalVariable()
    {
        $GLOBALS['Sum'] = $GLOBALS['firstNumber'] + $GLOBALS['secondNember'];
    }

    GlobalVariable ();
    echo $Sum;

?>
```

همانطور که در کد بالا مشاهده می کنید متغیر های $firstNumber و $secondNember که در خارج از متد تعریف شده اند در داخل متد و متغیر Sum هم که در داخل متد تعریف شده است در خارج از متد و با استفاده از آرایه $GLOBALS قابل دسترسی هستند.

$_SERVER

این آرایه در PHP اطلاعاتی درباره عنوان ها، مسیر ها و محل دستورها می‌دهد. این آرایه مقادیری دریافت می کند که لیست آنها در جدول زیر آمده است:

آرایه فوق سراسری	مقادیر دریافتی	کاربرد
$_SERVER	PHP_SELF	نام فایل در حال اجرا را (نسبت به ریشه سایت) بر می گرداند.
	SERVER_ADDR	با استفاده از این متغیر می توانیم آی پی سرور را به دست بیاوریم.
	REMOTE_ADDR	آی پی کاربر را بر می گرداند.
	REQUEST_URI	آدرس صفحه درخواستی را برمی گرداند
	HTTP_REFERRER	با استفاده از این مقدار می توانید بفهمید که کاربری که وارد سایت شما شده از چه سایتی آمده است
	DOCUMENT_ROOT	ریشه اصلی سایت رو بر می گرداند
	SCRIPT_FILENAME	مسیر مطلق فایل در حال اجرا رو بر می گرداند

ابته مقادیری که این آرایه دریافت می کند بیشتر از جدول بالاست که ما پرکاربردترین آنها را ذکر کرده ایم. برای آشنایی با کاربرد این آرایه به مثال زیر توجه کنید:

```php
<?php
    echo $_SERVER['PHP_SELF'];
    echo "<br>";
    echo $_SERVER['SERVER_NAME'];
    echo "<br>";
    echo $_SERVER['HTTP_HOST'];
    echo "<br>";
    echo $_SERVER['HTTP_USER_AGENT'];
    echo "<br>";
    echo $_SERVER['SCRIPT_NAME'];
?>
```

```
/Tuts/index.php
localhost
localhost
Mozilla/5.0 (Windows NT 6.1; WOW64; rv:42.0) Gecko/20100101 Firefox/42.0
/Tuts/index.php
```

در کد بالا با استفاده از آرایه $SERVER_ اطلاعی از قبیل نام سرور، نام دستورها، http مراجعه کنندگان، هاست و... را چاپ کرده ایم. البته این خروجی ممکن است در کامپیوتر شما متفاوت باشد. در باره سایر آرایه های فوق سراسری در درس های آینده توضیح می دهیم.

مدیریت استثناءها و خطایابی

بهترین برنامه نویسان در هنگام برنامه نویسی با خطاها و باگ ها در برنامه شان مواجه می شوند. درصد زیادی از برنامه ها هنگام تست برنامه با خطا مواجه می شوند. بهتر است برای از بین بردن یا به حداقل رساندن این خطاها، به کاربر در مورد دلایل به وجود آمدن آنها اخطار داده شود. خوشبختانه PHP دارای یک مکانیسم داخلی و راه هایی برای نشان دادن دلیل وقوع خطا در هنگام اجرای برنامه است. استثناءها توسط برنامه به وجود می آیند و شما لازم است که آنها را اداره کنید. به عنوان مثال در دنیای کامپیوتر یک عدد صحیح هرگز نمی تواند بر صفر تقسیم شود. اگر بخواهید این کار را انجام دهید (یک عدد صحیح را بر صفر تقسیم کنید)، با خطا مواجه می شوید .اگر یک برنامه در PHP با چنین خطایی مواجه شود پیغام خطای "DivideByZeroException"نشان داده می شود که بدین معنا است که عدد را نمی توان بر صفر تقسیم کرد. باگ (Bug)اصطلاحا خطا یا کدی است که رفتارهای ناخواسته ای در برنامه ایجاد می کند. خطایابی فرایند برطرف کردن باگها است ، بدین معنی که خطاها را از برنامه پاک کنیم PHP .دارای ابزارهایی برای خطایابی هستند ، که خطاها را یافته و به شما اجازه می دهند آنها را برطرف کنید. در درسهای آینده خواهید آموخت که چگونه از این ابزارهای کارامد جهت برطرف کردن باگها استفاده کنید. قبل از اینکه برنامه را به پایان برسانید لازم است که برنامه تان را اشکال زدایی کنید.

استثناءهای اداره نشده

استثناءهای اداره نشده، استثناءهایی هستند که به درستی توسط برنامه اداره نشده اند و باعث می شوند که برنامه به پایان برسد. در اینجا می خواهیم به شما نشان دهیم که وقتی یک برنامه در زمان اجرا با یک استثناء مواجه می شود و آن را اداره نمی کند چه اتفاقی می افتد. در آینده خواهید دید که یک استثناء چگونه به صورت بالقوه باعث نابودی جریان و اجرای برنامه شما می شود. به کدهای زیر توجه کنید:

```
<?php

    $five = 5;
    $zero = 0;

    $result = $five / $zero ;

    echo $result;

?>
```

```
( ! ) Warning: Division by zero in ...
```

همانطور که در کد بالا مشاهده می کنید، خط 6 دلیل به وجود آمدن خطاست، چون تقسیم عدد بر صفر غیر ممکن است. این خطا باعث می شود که کدهای بعد از ان اجرا نشوند.

دستورات try و catch

می توان خطاها را با استفاده از دستور **try…catch** اداره کرد. بدین صورت که کدی را که احتمال می دهید ایجاد خطا کند در داخل بلوک **try**قرار می دهید. بلوک **catch** هم شامل کدهایی است که وقتی اجرا می شوند که برنامه با خطا مواجه شود. تعریف ساده ی این دو بلوک به این صورت است که بلوک **try** سعی می کند که دستورات را اجرا کند و اگر در بین دستورات خطایی وجود داشته باشد برنامه دستورات مربوط به بخش **catch** را انجام می دهد .برنامه زیر نحوه استفاده از دستور **try…catch**را نمایش می دهد:

```
1   <?php
2
3       $five = 5;
4       $zero = 0;
5
6       try
7       {
8           if($zero > 0  ||  $zero < 0)
9           {
10              $result = $five / $zero ;  //Error
11              echo $result;
12          }
13          else
14          {
15              throw new Exception('An attempt to divide by 0 was
16  detected.');
17          }
18      }
19      catch(Exception $ex)
20      {
21          echo 'Error : '.$ex -> getMessage();
22      }
23
24  ?>
```

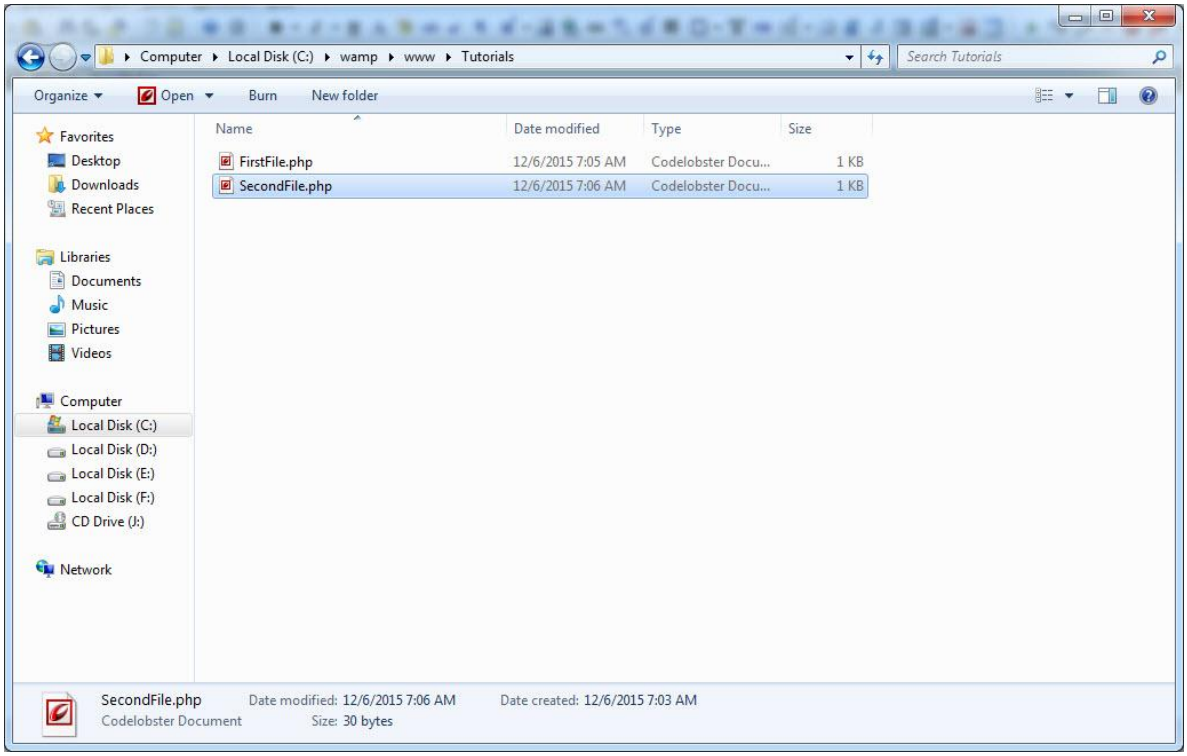
در کد بالا ما قصد داریم خطاهای احتمالی را که در عملیات تقسیم ممکن است به وجود آید را اداره کنیم. در ریاضیات عمل تقسیم عدد بر صفر ممکن نیست .در خطوط **3** و **4** دو متغیر تعریف کرده ایم که مقدار یکی از آنها **5** و دیگری **0** است. چون ممکن اس که عمل تقسیم بر صفر اتفاق افتد پس در قسمت **try** در خط **8** با استفاده از دستور **if** چک می کنیم که اگر مقدار متغیر **$zero**بزرگتر و یا کوچکتر از **0** باشد عملیات تقسیم انجام و نتیجه نمایش داده شود)خطوط **10** و **11**) در غیر اینصورت یک استثناء ایجاد کند. از آنجاییکه مقدار متغیر **$zero**عدد **0** است، یک استثناء رخ می دهد. این استثناء را در خط **15** و در قسمت **else** می نویسیم و یک پیغام دلخواه و قابل فهم برای نمایش به کاربر می نویسیم. برای چاپ پیغام خطا در دستور **catch** یک نمونه از کلاس **Exception** ایجاد کرده (خط **18**) و با فراخوانی متد (**()getMessage**خط **20**) این کلاس آن را چاپ و به کاربر نمایش می دهیم.

دستورات include و require

در PHP چهار دستور برای وارد کردن محتویات یک فایل در داخل فایل دیگر وجود دارد که در زیر نام آنها آمده است:

- **require**
- **require_once**
- **include**
- **include_once**

برای درک عملکرد این دو دستورات دو فایل به نام های FirstFile.php و SecondFile.php ایجاد کرده:



و در داخل FirstFile کد زیر را:

```
<p>This is First File !</p>
```

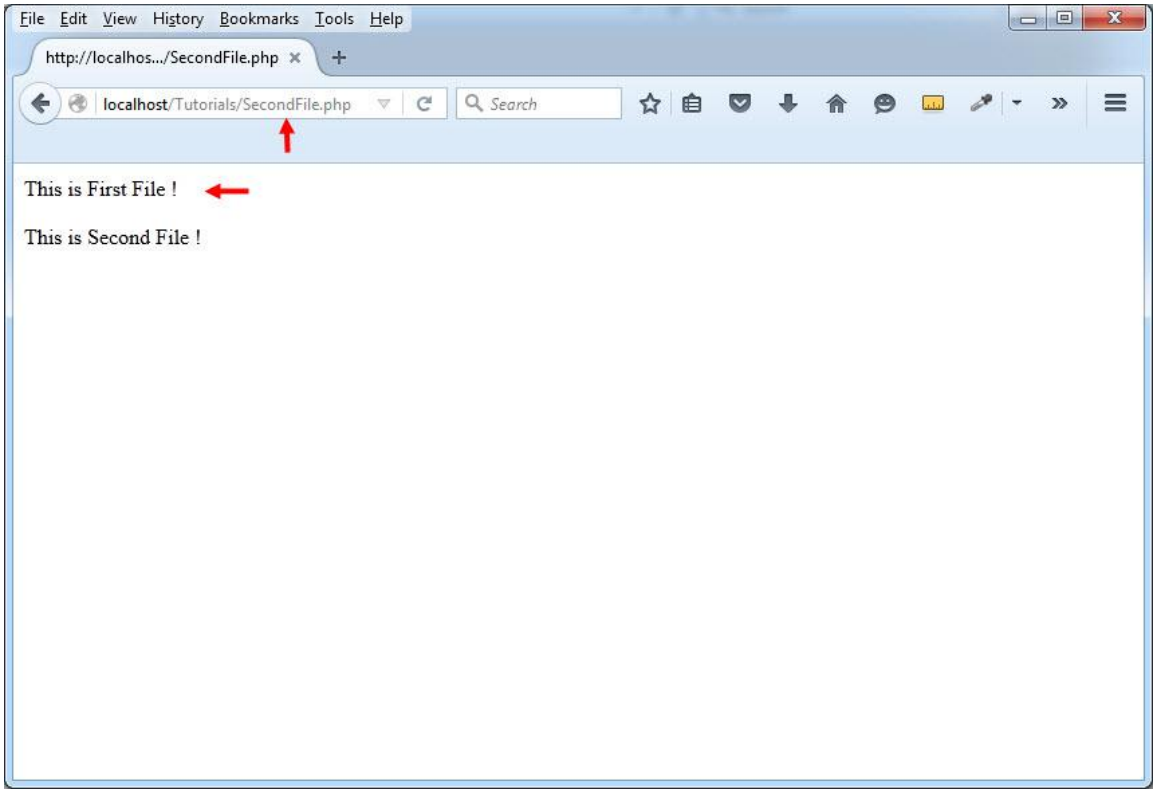
و در داخل SecondFile هم کد زیر را بنویسید:

```
<p>This is Second File !</p>
```

حال فرض کنید که می خواهیم کدهای فایل FirstFile را در داخل SecondFile یک بار با include و بار دیگر با require وارد کنیم. برای این کار فایل SecondFileرا باز کرده و کدهای آن را به صورت زیر تغییر دهید:

```
<?php require 'FirstFile.php'; ?>

<p>This is Second File !</p>
```

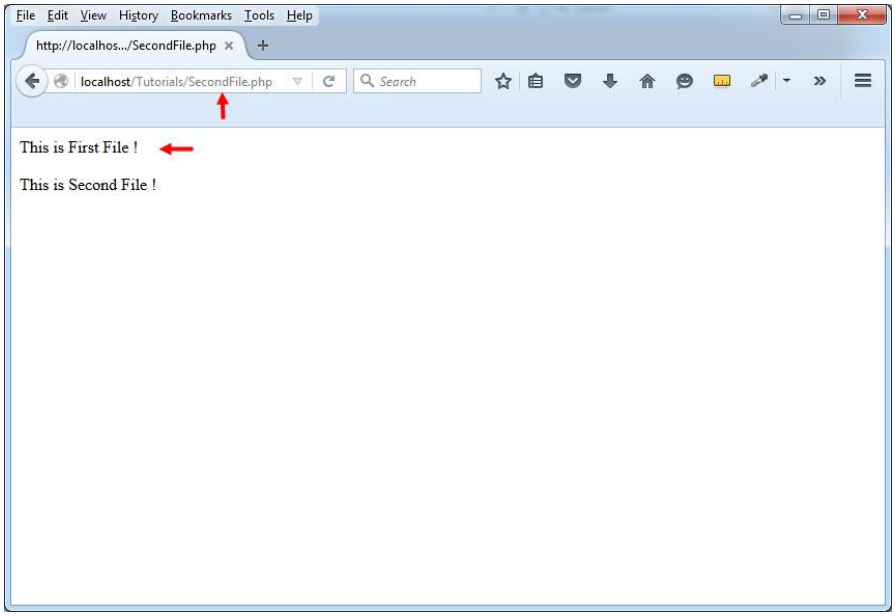
کد را اجرا کنید:



همانطور که در شکل بالا مشاهده می کنید با اجرای فایل SecondFile محتویات FirstFile هم که به وسیله دستور require به آن اضافه شده اند اجرا می شوند. در کل کد بالا با کد زیر هیچ فرقی ندارد:

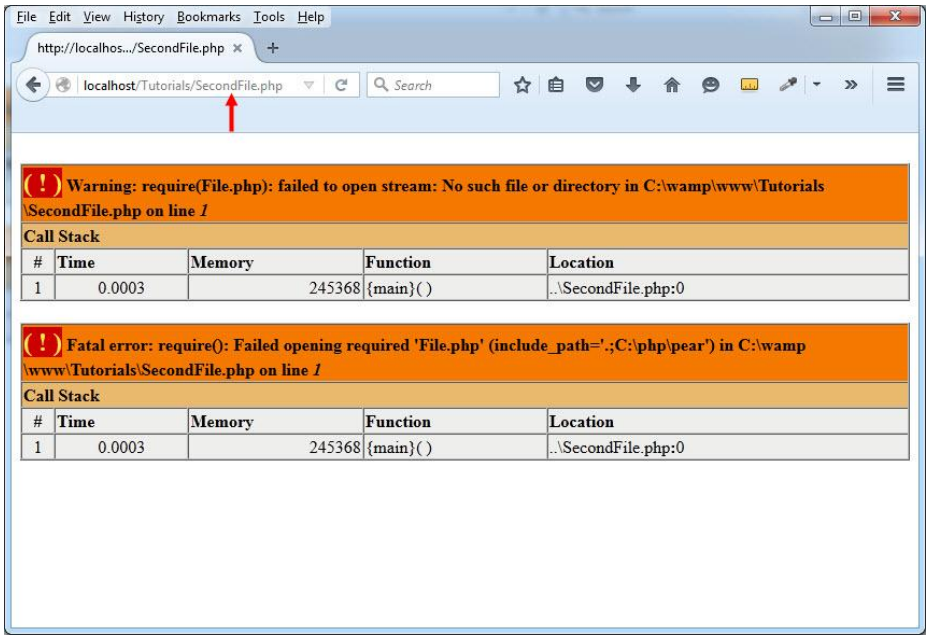
```
<p>This is First File !</p>

<p>This is Second File !</p>
```

حال به جای کلمه require از کلمه include استفاده و فایل را اجرا کنید:



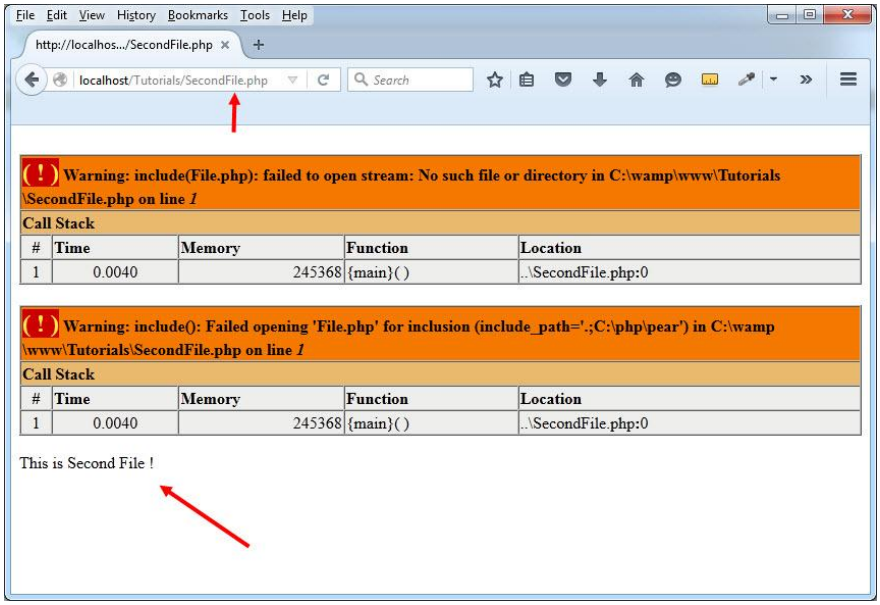
همانطور که در شکل بالا مشاهده می کنید هیچ تفاوتی در خروجی وجود ندارد .تفاوت این دو دستور در این است که اگر مثلا اشتباهی در دادن مسیر FirstFileو یا غلط املایی در نوشتن نام آن وجود داشته باشد، رفتار این دو دستور متفاوت است. برای روشن شدن این موضوع نام FirstFile را به File تغییر داده و کد را اجرا کنید:

با استفاده از require



در شکل بالا همانطور که می بینید اگر فایل File.php پیدا نشود، هیچ کدی اجرا نمی شود.

با استفاده از include



اما در دستور inculde ، اگر فایل File.php پیدا نشود، بقیه کدها اجرا می شوند .require_once و include_once هم برای اطمینان از عدم ورود چند باره یک فایل به داخل فایل دیگر به کار می شوند. برای روشن شدن مطلب به کد زیر توجه کنید :

```
<?php include 'FirstFile.php'; ?>

<p>This is Second File !</p>

<?php include 'FirstFile.php'; ?>
```

```
This is First File !
This is Second File !
This is First File !
```

با اجرای کد بالا فایل FirstFile دو بار در داخل فایل SecondFile وارد می شود. برای جلوگیری از این کار کد بالا را به صورت زیر تغییر داده و آن را اجرا کنید:

```
<?php include 'FirstFile.php'; ?>
```

```
<p>This is Second File !</p>

<?php include_once 'FirstFile.php'; ?>
```

```
This is First File !
This is Second File !
```

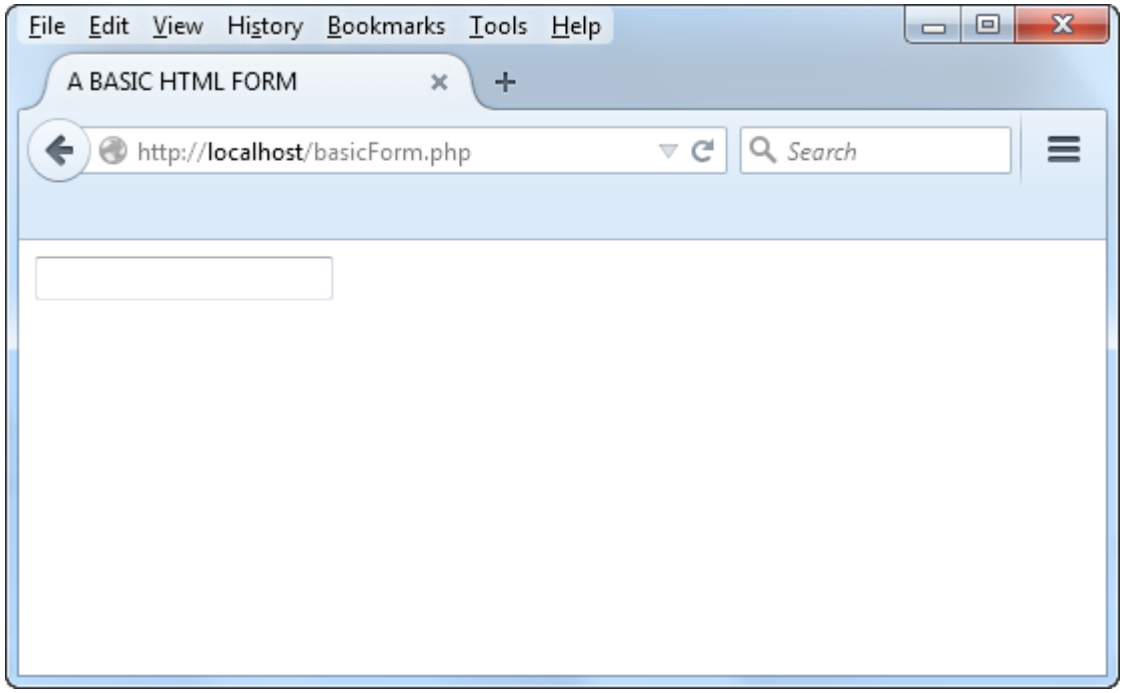
حال همین دو کد بالا را با require_once چک کرده و نتیجه را مشاهده کنید.

تگ input

از آنجاییکه از تگ input در فرم های HTML و همچنین گرفتن ورودی و ارسال اطلاعات زیاد استفاده می شود. ابتدا لازم است در مورد این تگ مهم مختصری توضیح دهیم. تگ input برای ساختن انواع کنترل های HTML از قبیل کادر های متن ، لیست های بازشو ، دکمه های فرمان و انتخابی و ... درون فرم ها استفاده می شود . نوع کنترل توسط خاصیت Type در درون تگ input مشخص می شود. مثلا برای ایجاد یک جعبه متن به وسیله این تگ به صورت زیر عمل می کنیم :

```
<INPUT TYPE ="TEXT" />
```

خروجی کد بالا به صورت زیر است:



مقادیری که خاصیت type تگ input می تواند دریافت کند در جدول زیر آمده است:

مقدار	کاربرد
Button	دکمه فرمان.
Checkbox	کادر یا دکمه گزینشی
File	یک فایل خارجی مثل یک صفحه اینترنتی و. ...
Hidden	یک کنترل مخفی دلخواه را بر روی صفحه ایجاد می کند.

Image	یک کنترل برای نمایش عکس یا تصویر ایجاد می کند.
Password	یک کادر متن است که به جای نمایش اعداد یا حروف وارد شده ، ستاره یا دایره نمایش می دهد تا اطلاعات ورودی توسط کاربر مخفی بماند.
Radio	دکمه انتخابی.
Reset	یک دکمه فرمان است که با کلیک بر روی آن ، محتویات درون کلیه کنترل های فرم پاک می شود.
Submit	یک دکمه فرمان است که با کلیک بر روی آن ، اطلاعات درون کنترل های فرم به مقصد ارسال می شود.
text	یک کادر متن ایجاد می کند.

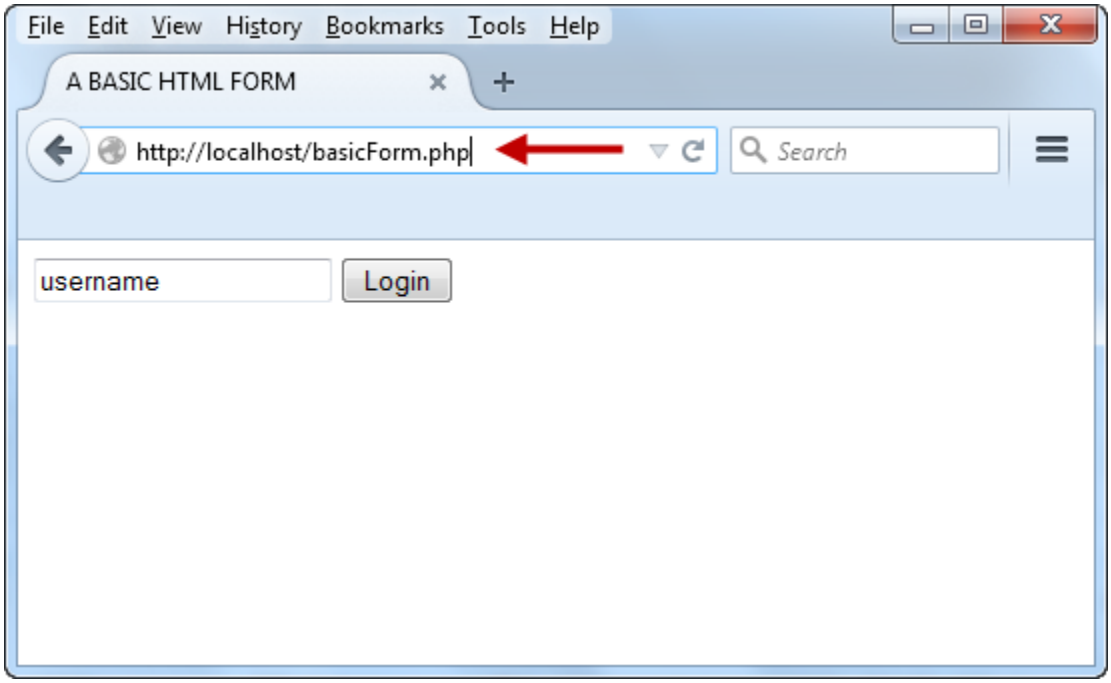
در درس های بعدی در مورد هر کدام از مقادیر بالا و کاربرد آنها در انتقال اطلاعات و فرم ها بیشتر توضیح می دهیم.

تگ Form

اگر کمی با HTML آشنایی داشته باشید می دانید که از تگ FORM برای تعامل با کاربر استفاده می شود.چیزهایی که می توانید به فرمها اضافه کنید عبارتند از جعبه های متن ، دکمه های رادیویی، چک باکس ها، لیست های بازشونده و دکمه های ثبت. به یک فرم ساده HTML توجه کنید. این فرم شامل یک جعبه متن، و یک دکمه ثبت می باشد:

```
1  <html>
2  <head>
3        <title>A BASIC HTML FORM</title>
4  </head>
5  <body>
6
7        <FORM  NAME ="form1" METHOD ="" ACTION = "">
8              <INPUT  TYPE = "TEXT"    VALUE = "username"/>
9              <INPUT  TYPE = "Submit"  VALUE = "Login"  Name = "Submit1"
10 />
11       </FORM>
12
13 </body>
14 </html>
```

ما در این آموزش قصد توضیح کدهای HTML را نداریم و فرض را بر این می گذاریم که شما با HTML آشنایی دارید.اما در مورد صفات METHOD ، ACTIONو SUBMIT در فرم بالا توضیح می دهیم چون مهم هستند. بعد از اینکه فرم بالا را ایجاد کردید آن را با عنوان basicForm.php ذخیره کنید(این نام بسیار مهم است). سرورتان را استارت کنید و از بارگذاری فرم در مرورگرتان مطمئن شوید (بر روی basicForm.php کلیک کنید تا توسط مرورگر نمایش داده شود). بعد از اجرای فایل شکلی شبیه به شکل زیر مشاهده می کنید:



اگر کاربری وارد سایت شما شده و مثلا بخواهد در سایت شما ثبت نام کند لازم است که اطلاعات خود را وارد جعبه متن کند و دکمه **Login** را بزند .اطلاعات وارد شده توسط کاربر با اطلاعات کاربرانی که قبلا در سایت شما ثبت نام کرده اند مقایسه می شود(این اطلاعات معمولا در دیتابیس ذخیره می شوند که در بخش های آینده در مورد آنها توضیح می دهیم). حال لازم است که در مورد عملکرد صفات **METHOD** ، **ACTION** و **SUBMIT** آشنا شوید.

صفت METHOD

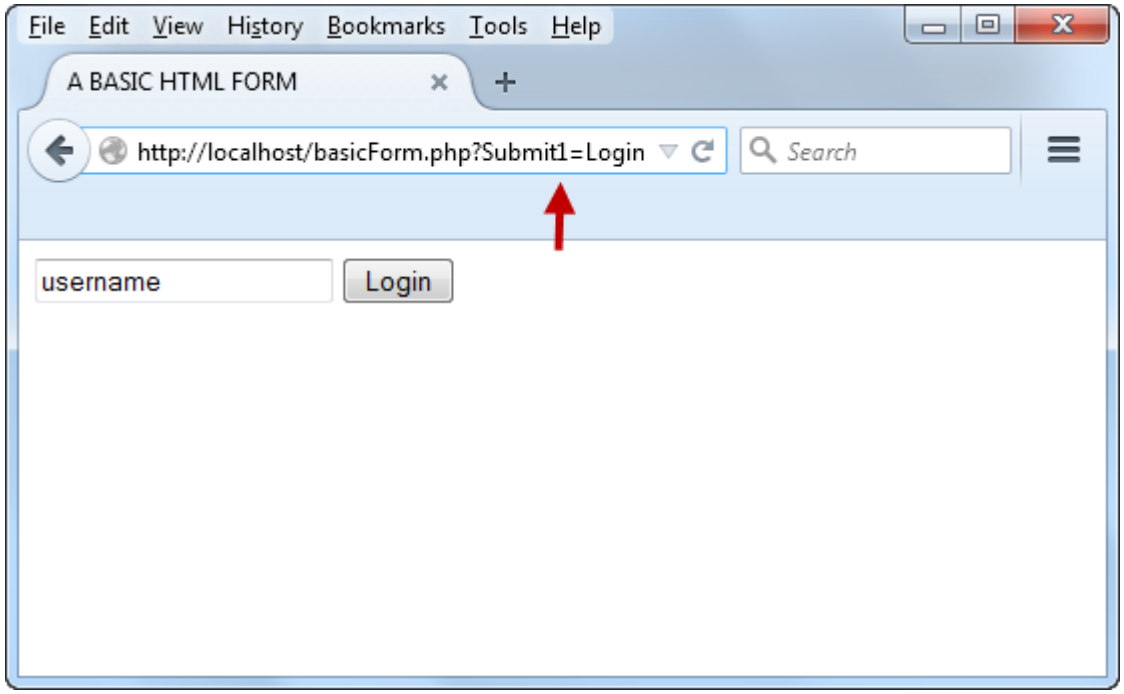
اگر به خط اول فرم مان درس درس قبل نگاهی بیندازید متوجه صفت **Method** خواهید شد:

```
<FORM  NAME ="form1" METHOD =" "ACTION = "">
```

این صفت به مرورگر می گوید که اطلاعات را چگونه ارسال کند و دو مقدار می گیرد یکی **GET** و دیگری . **POST** ابتدا از مقدار **GET** استفاده می کنیم. فایل **basicForm.php** را باز کرده و تغییر زیر را اعمال می کنیم:

```
<FORM  NAME ="form1" METHOD ="GET" ACTION = "">
```

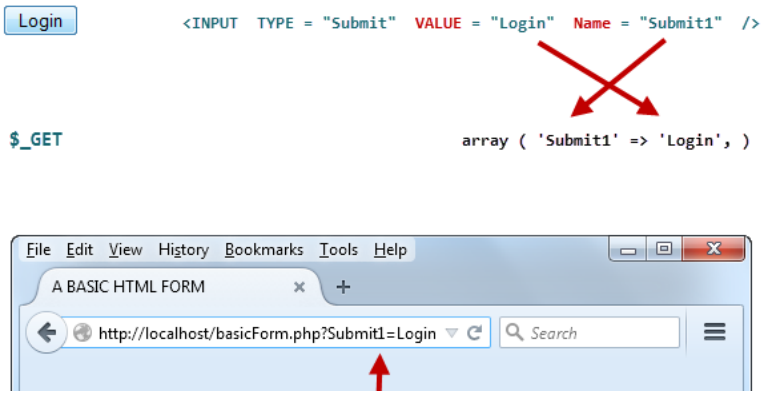
برای مشاهده تاثیر این متد کد را ذخیره کرده و آنرا توسط مرورگر اجرا کرده و دکمه **Login** را فشار دهید. چیزی که مشاهده می کنید به صورت زیر است:



به نوار آدرس توجه کنید.بعد از basicForm.php?Submit1=Login، را مشاهده می کنید. این به دلیل استفاده از مقدار GETمی باشد. داده های ارسالی از فرم در نوار آدرس نمایش داده می شوند.در عکس بالا ، Submit1نام دکمه و Login مقدار آن می باشد. به زبان ساده تر کد basicForm.php?Submit1=Login نشان می دهد که دکمه ای با نام Submit1 و مقدار Login فشرده شده است. به کد دکمه Login در فایل basicForm.php توجه کنید:

```
<INPUT TYPE = "Submit" Name = "Submit1" VALUE = "Login">
```

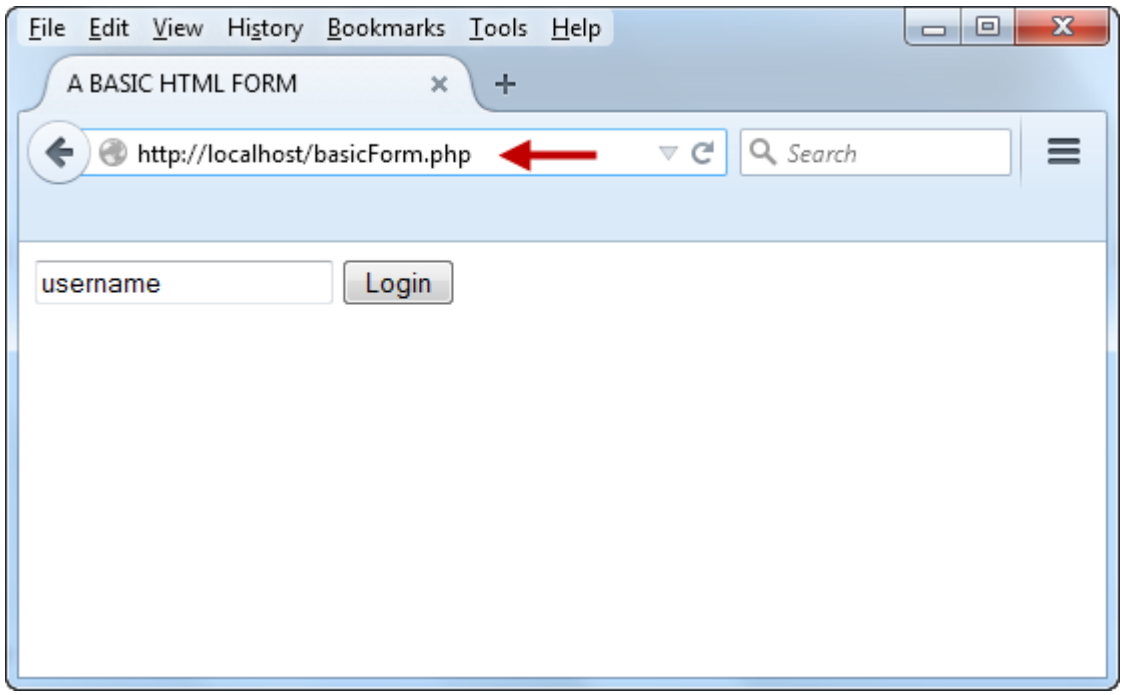
به این نکته توجه کنید که در HTML برای دسترسی به عناصر با خصوصیت id و classکار داشتیم اما در PHP با خصوصیت Nameعناصر سر و کار داریم. یعنی برای انجام اعمال خاص بر روی یک عنصر HTML باید از خاصیت Name آن استفاده کنیم. همانطور که در درس محدوده متغیرها توضیح دادیم، متغیرهای سراسری در PHP وجود دارند که در هر جای اسکریپت قابل دسترسی هستند یکی از این متغیرها $_GETاست. این متغیر یا بهتر بگوییم آرایه بعد از کلیک بر روی دکمه ارسال (submit)، مقادیر ارسالی را در خود ذخیره می کند. حال ما می توانیم با استفاده از قوانین آرایه به عناصر آن دسترسی داشته باشیم برای درک بهتر کاربرد این آرایه به شکل زیر توجه کنید:



همانطور که در شکل بالا مشاهده می کنید $_GET آرایه ای متشکل از مقدار خاصیت **Name** و مقدار خاصیت **Value** عناصر HTMLاست. از متد **GET** زمانی استفاده می شود که داده هایی که می خواهید ارسال کنید زیاد مهم نیستند. در درس قبل مشاهده کردید که هنگام استفاده از متد **GET** چه اتفاقی در نوار آدرس مرورگر رخ داد. حال به جای متد **GET** ا ز متد **POST**استفاده می کنیم. خط اول تگ فرم را به صورت زیر تغییر دهید:

```
<FORM  NAME ="form1" METHOD ="POST" ACTION = "">
```

اگر مرورگرتان باز است ان را بسته، فایل **basicForm.php** را اجرا و سپس بر روی دکمه **Login** کلیک کنید. به نوار آدرس نگاه کنید:



همانطور که مشاهده می کنید قسمت **Submit1=Login?** وجود ندارد که این به دلیل استفاده از متد **POST** می باشد. استفاده از متد **POST** بدین معناست که اطلاعات ارسالی درنوار آدرس مرورگر نشان داده نشود. پس به یاد داشته باشید که ما بسته به مهم و غیر مهم بودن اطلاعات از هر دو متد **POST** و **GET** در آموزش ها استفاده می کنیم. اگر اطلاعات مهم بودند از **POST**در غیر اینصورت از **GET** استفاده می کنیم.

صفت ACTION

یکی دیگر از صفات مهم تگ **FORM** صفت **Action** می باشد که بدون آن فرم نمی داند اطلاعات را به کجا ارسال کند. صفت **Action**در فرم های **HTML** از اهمیت بخصوصی برخوردار است. از این صفت از شما سوال می کند که : "می خواهید فرمتان را به کجا ارسال کنید؟". اگر این صفت را فراموش کنید فرم و به تبع آن اطلاعات وارد شده در فرم نمی دانند به کجا بروند! شما می توانید فرم را به یک اسکریپت **PHP** دیگر، اسکریپتی مشابه، آدرس ایمیل، یا هر شکل دیگر از اسکریپت ارسال کنید. یک تکنیک عمومی در **PHP** ارسال اطلاعات فرم به همان صفحه ای است که خود فرم در آن قرار دارد. قسمت **ACTION** فرم ایجاد شده در درس های قبل را به صورت زیر تغییر دهید:

```
<Form Name ="form1" Method ="POST" ACTION = "basicForm.php">
```

در کد بالا ما اطلاعات فرم را به خود صفحه ای که فرم در آن قرار دارد **(basicForm.php)**ارسال کرده ایم. برای کنترل اطلاعات ارسال شده لازم است کدهای دیگری را به صفحه **basicForm.php** اضافه کنیم، اما فعلا این کار را انجام نمی دهیم. کد را ذخیره کرده و دوباره بر روی دکمه کلیک کنید. همانطور که می بینید هیچ پیغام خطایی نمایش داده نمی شود. برای اراسل اطلاعات در یک فرم لازم است که نام صفحه ای که قرار است اطلاعات به انجا ارسال شوند در قسمت **Action** ذکر شود. در مورد دکمه **Submit** در درس آینده توضیح می دهیم.

دکمه ارسال (submit)

همانطور که در آموزش تگ **input** گفتیم، از دکمه **submit** در فرم های**HTML** ، برای ارسال داده های فرم به اسکریپت موجود در صفت **ACTION** استفاده می شود. به کد زیر توجه کنید:

```
<Form Name ="form1" Method ="POST" ACTION = "basicForm.php">
```

همانطور که مشاهده می کنید در جلوی صفت **ACTION** در کد بالا نام **basicForm.php**نوشته شده است که بدین معناست که اطلاعات فرم باید به این صفحه ارسال شوند. برای ارسال این اطلاعات شما فقط به یک دکمه **submit** نیاز دارید:

```
<INPUT TYPE = "Submit"Name = "Submit1" VALUE = "Login">
```

لازم نیست که شما کار اضافی با این دکمه انجام دهید.تمام کارها در پس زمینه انجام می شوند. زمانی که دکمه **submit** را فشار می دهید اطلاعات وارد شده در جعبه متن فرم به صفحه مشخص شده در صفت **ACTION** ارسال می شوند. صفت **NAME** دکمه های **submit** نیز اختیاری است و هر چه می تواند باشد. ولی بهتر است که این نام را با دقت انتخاب کنید. استفاده از نام زمانی مهم است که شما بخواهید چک کنید که آیا کاربر واقعا دکمه **submit** را فشار داده است یا نه؟ در مورد این صفت در درس های آینده توضیح می دهیم. در کد بالا نام دکمه را **submit1**انتخاب کرده ایم که شما می توانید آن را به دلخواه تغییر دهید.

حال که با عملگرد صفت های**METHOD** ، **ACTION**و **SUBMIT** آشنا شدید می توانید داده های وارد شده توسط کاربر را پردازش کنید. در درس بعد چگونگی دریافت مقادیر از جعبه متن را مورد بررسی قرار می دهیم.

جعبه متن (text)

همانطور که در درس های قبل دیدید ما در صفحه **basicForm.php** مقادیری به خاصیت های **METHOD** و **ACTION** اختصاص دادیم. حال می خواهیم نحوه پردازش اطلاعات وارد شده توسط کاربر در فرم را مورد بررسی قرار دهیم. خاصیت **METHOD**به شما می گوید که می خواهید چگونه اطلاعات را ارسال کنید و خاصیت **ACTION**می گوید که اطلاعات را می خواهید به کجا ارسال کنید.

برای دریافت اطلاعات وارد شده توسط کاربر در جعبه های متن ، لازم است که جعبه متن دارای خاصیت **Name** باشد. شما با استفاده از این خاصیت به **PHP** می گویید که می خواهید با جعبه متنی با این نام کار کنید. جعبه متن ما دارای خاصیت **Name** نیست. کدهای جعبه متن موجود در فایل **basicForm.php** را به صورت زیر تغییر می دهیم و به آن صفت **Name** را اضافه می کنیم:

```
<INPUT TYPE = "Text"VALUE ="username" NAME = "username">
```

همانطور که در کد بالا می بینید، مقدارخاصیت **Name** جعبه متن **username** است. در اصل **PHP** با خاصیت **name** عناصر **HTML**سرو کار دارد. برای دریافت مقادیر ارسال شده از عناصر **HTML** از دو آرایه فوق سراسری **$_POST** و **$_GET** استفاده می شود:

```
$_POST['formElement_name'];
```

یا

```
$_GET['formElement_name'];
```

البته استفاده از این دو آرایه بستگی به این دارد که شما در خاصیت **method**فرمتان از کدام یک استفاده کرده باشید. مقدار دریافت شده از عنصر را می توان به صورت زیر در داخل یک متغیر قرار داد:

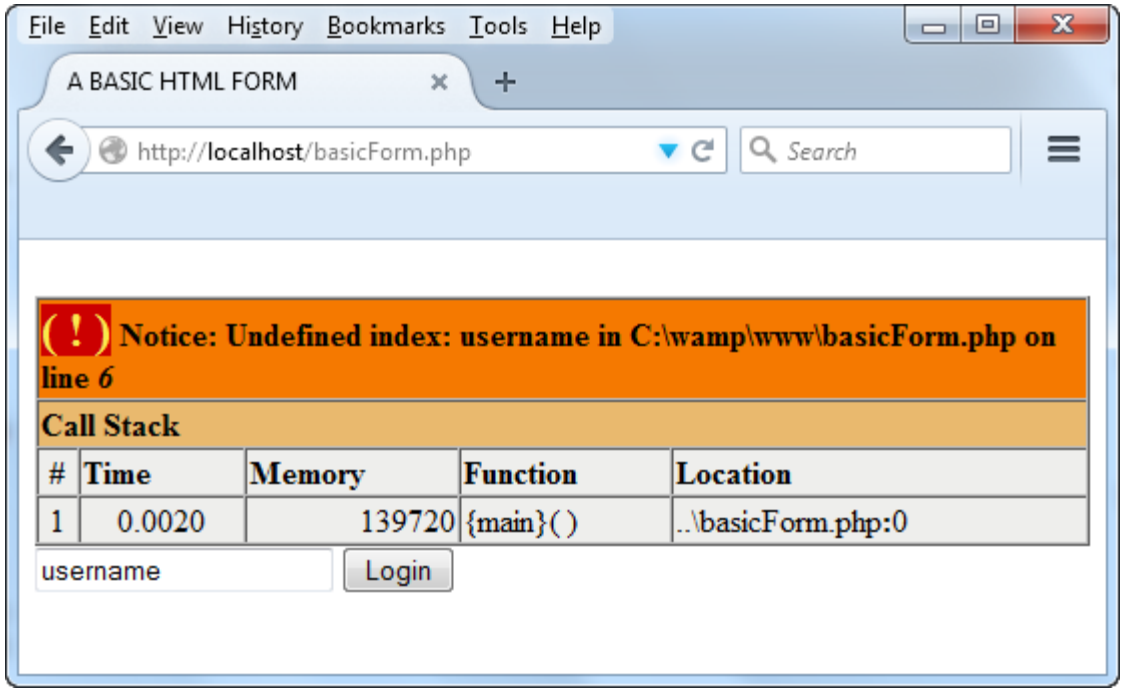
```
$Your_Variable = $_POST['formElement_name'];
```

قبل از توضیح دستور بالا، کد **PHP** زیر را به بخش **HEAD** فایل **HTML** خود اضافه کنید:

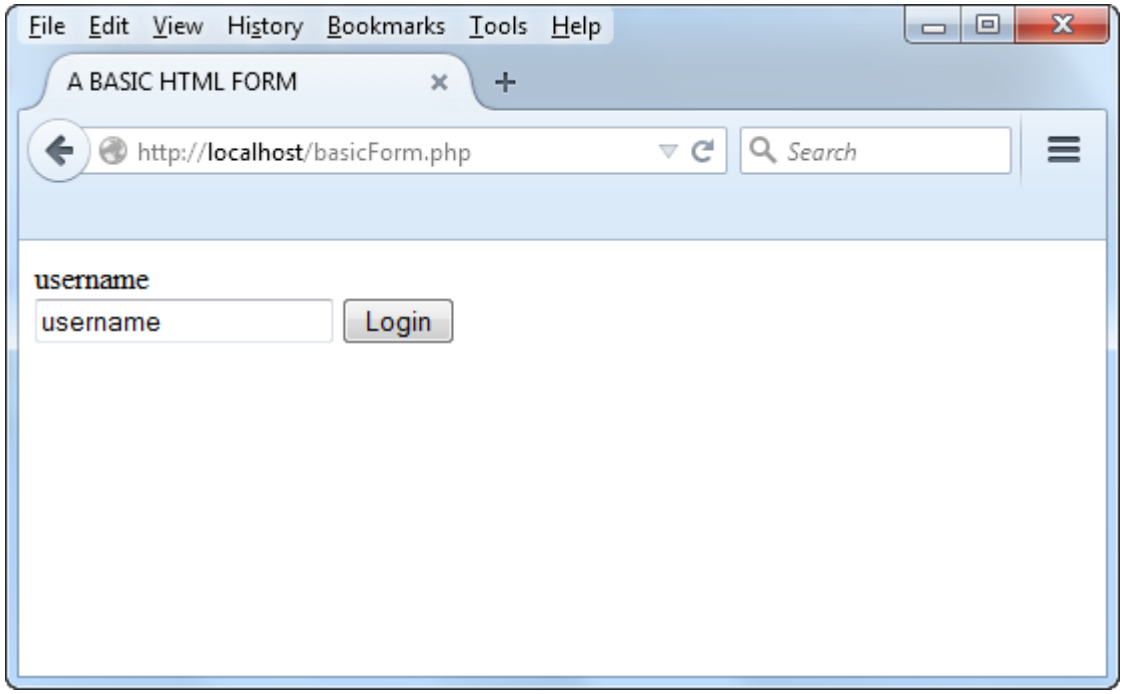
```
1  <html>
2  <head>
3  <title>A BASIC HTML FORM</title>
4
5  <?PHP
6    $username = $_POST['username'];
7    print ($username);
```

```
8   ?>
9
10  </head>
11  <body>
12
13        <FORM  NAME ="form1" METHOD ="POST" ACTION = "basicForm.php">
14              <INPUT  TYPE = "TEXT"    VALUE = "username" Name =
15  "username"/>
16              <INPUT  TYPE = "Submit"  VALUE = "Login"    Name = "Submit1"
17  />
18        </FORM>
19
20  </body>
21  </html>
```

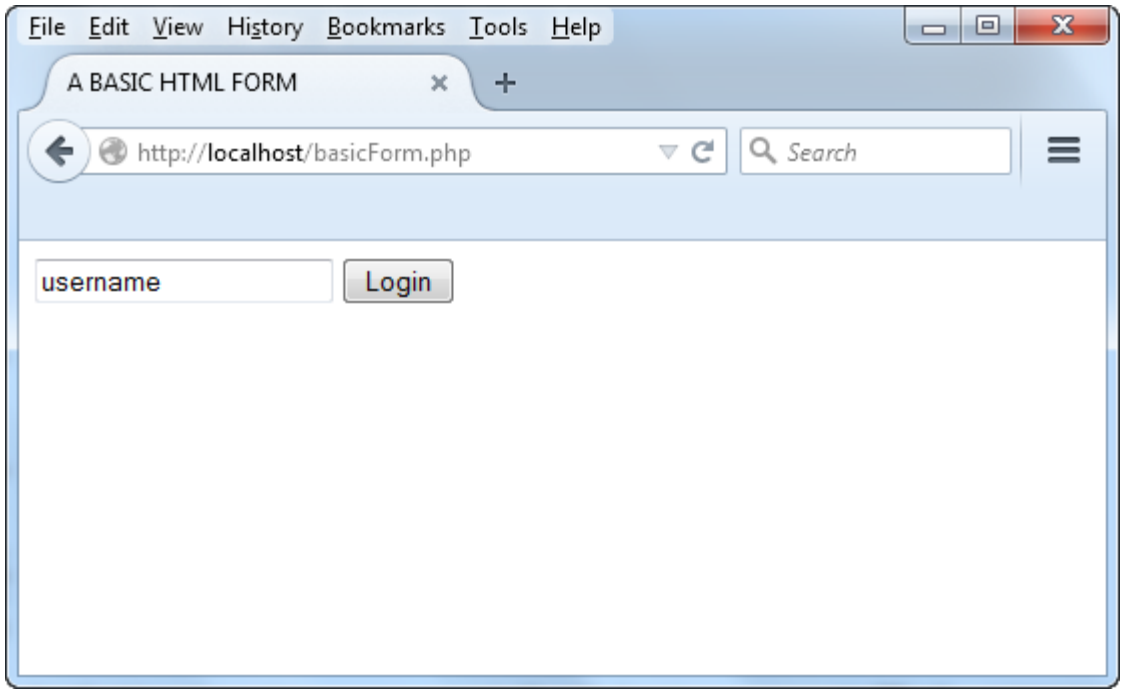
کد را ذخیره و اجرا کنید. بعد از اجرای کد پیغام خطایی به صورت زیر نمایش داده می شود:



نگران پیغام خطای "Undefined index" نباشید، در ادامه درباره علت به وجود آمدن این خطا توضیح می دهیم. بر روی دکمه کلیک کنید. بعد از کلیک بر روی دکمه صفحه شما به شکل زیر در می آید:



متن "username" که در داخل جعبه متن است را پاک کرده و دوباره بر روی دکمه کلیک کنید. همانطور که مشاهده می کنید متن بالای جعبه متن از بین می رود و فرم به شکل زیر در می آید:



دلیل اینکه متن داخل جعبه متن از بین نرفته است این است که ما آن را در داخل خاصیت VALUE جعبه متن نوشته ایم .
حال به توضیح کد می پردازیم $_POST[] .یک تابع درونی PHP است که با استفاده از آن می توانید مقادیر ارسالی از فرم توسط متد POST را دریافت کنید. اگر از متد GET برای ارسال اطلاعات استفاده می کنید باید به صورت زیر عمل کنید:

```
$username =$_GET['username'];
```

ابتدا یک علامت $، سپس علامت _ و درنهایت نام متد POST و یا GET را می نویسید. بعد از نام متد علامت [] در داخل علامت کروشه نام عنصری HTML ی را که می خواهید با آن کار کنید بنویسید که در مثال بالا نام عنصر مورد نظر username است.

```
$_POST['username'];
```

مقداری را که در خاصیت VALUE عنصر HTML قرار دارد می توانید در داخل یک متغیر قرار دهید:

```
$username = $_POST['username'];
```

در کد بالا PHP به عنصر HTML ی با نام username مراجعه و مقداری را که در خاصیت VALUE آن قرار دارد را بر می دارد. تمام کاری را که ما در کدهای قبلی انجام داده ایم این است که مقدار VALUE عنصری با نام username را در مرورگر چاپ کرده ایم. حال به همین کد می توانیم یک شرط هم به صورت زیر اضافه کنیم:

```
<?PHP

	$username = $_POST['username'];

	if ($username == "jack")
	{
		print ("Welcome back, friend!");
	}
	else
	{
		print ("You're not a member of this site");
	}

?>
```

ما در دستور بالا به PHP گفته ایم که اگر مقداری که توسط کاربر وارد شده است برابر jack بود، متن "Welcome back, friend!" و در غیر اینصورت متن "You're not a member of this site" را در خروجی نمایش بده.

چک فشرده شدن دکمه submit

در کد بالا چگونگی دریافت متن از جعبه متن را مشاهده کردید و دیدید که قبل از فشردن دکمه پیغامی در بالای جعبه متن به نمایش در می آمد. دلیل نمایش این پیغام این بود که ما در دستور شرطی چک نکردیم که آیا دکمه توسط کاربر فشرده شده است یا نه؟ برای حل این مشکل (نمایش خودکار پیغام) باید از یک دستور شرطی مثلا IF استفاده کنید و با استفاده از آن چک کنید که آیا دکمه فشرده شده است یا نه و پیغام متناسب با آن نمایش داده شود. برای این کار از دستور زیر استفاده می کنیم:

```
if ( isset( $_POST['Submit1'] ) ) { }
```

کد بالا کمی شلوغ به نظر می رسد. این کد از سه بخش تشکیل شده است:

```
if ( ) { }
isset( )
$_POST['Submit1']
```

شما با دستور IF آشنا هستید ولی دستور isset نه. این کد یک تابع از پیش ساخته PHP می باشد که مقداردهی شدن یا نشدن یک متغیر را چک می کند. که در مورد مثال ما باید متغیر $_POST['Submit']چک شود. حال اگر کاربر صفحه را از نو بارگذاری کند (refresh) هیچ مقداری به این متغیر اختصاص داده نمی شود در غیر اینصورت اگر بر روی دکمه submit کلیک کند، PHPبه طور خودکار یک مقدار را بر می گرداند. اسکریپت بالا را به شکل زیر تغییر دهید:

```
<?PHP

    if (isset($_POST['Submit1']))
    {
        $username = $_POST['username'];
        if ($username == "jack")
        {
            print ("Welcome back, friend!");
        }
        else
        {
            print ("You're not a member of this site");
        }
    }

?>
```

همانطور که در دستور if اول کد بالا مشاهده می کنید، ابتدا با استفاده از تابع isset تست می کنیم که آیا دکمه ارسال که نام آن submit1 است فشرده شده است یا نه؟ با این کار از بروز خطا در هنگام اجرای اسکریپت جلوگیری می شود. کاملا حواستان به کروشه ها ، براکت ها و آکولادها باشد، از قلم انداختن یکی از آنها باعث بروز خطا می شود. در درس بعد با نحوه ارسال اطلاعات فرم از یک اسکریپت PHP به صفحه دیگر آشنا می شوید.

دکمه رادیویی (Radio)

یک **Radio** دکمه ای است که دارای دو حالت خاموش و روشن می باشد. دکمه ی **Radio** یک دکمه ی دایره ای شکل به همراه یک برچسب است. شما با کلیک کردن بر روی دکمه ی **Radio** می توانید آنرا از حالت خاموش به روشن ویا بلعکس تغییر دهید. وقتی که یک دکمه ی **Radio** روشن باشد ، یک نقطه درر وسط آن قرار می گیرد ، و زمانی که خاموش باشد ، دایره ی آن خالی است.

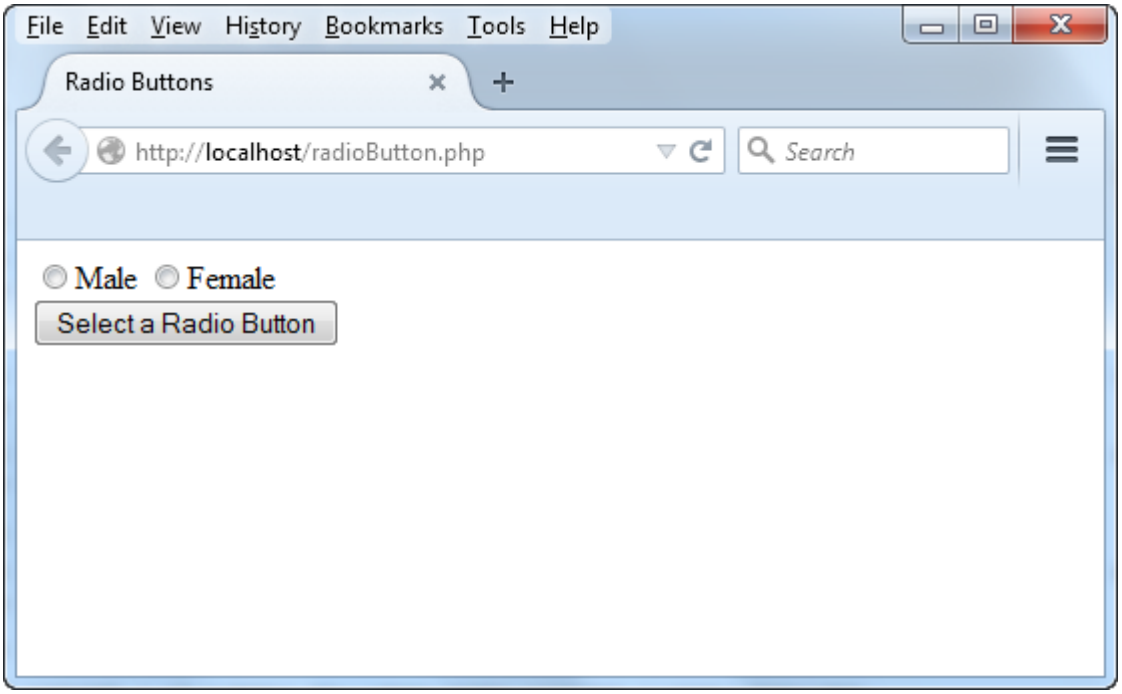
دکمه های **Radio** معمولا زمانی استفاده می شوند که یک کاربر می بایست از بین چند گزینه یکی از آنها را انتخاب کند. برای مثال: زمانی که شما بخواهید جنسیت کاربر را مشخص کنید ، می توانید از دو دکمه ی **Radio** با نام های مرد و زن استفاده کنید. وقتی که شما از دکمه های **Radio** استفاده کردید ، فقط می توانید یکی از آن دو را انتخاب کنید. برای ایجاد دکمه رادیویی در **HTML** به صورت زیر عمل کنید:

```
<INPUT TYPE ="radio" />
```

حال می خواهیم که شما را با نحوه استاده و دریافت مقادیر از دکمه رادیویی آشنا کنیم.کدهای زیر یک فرم که دارای دو دکمه رادیویی و یک دکمه ارسال است، را ایجاد می کنند:

```
1  <html>
2  <head>
3      <title>Radio Buttons</title>
4  </head>
5  <body>
6
7  <Form name ="form1" Method ="Post" ACTION ="radioButton.php">
8
9      <Input type = 'Radio' Name ='gender' value= 'male'>Male
10     <Input type = 'Radio' Name ='gender' value= 'female'>Female
11     <br/>
12     <Input type = "Submit" Name = "Submit1" Value = "Select a Radio
13 Button">
14
15 </FORM>
16
17 </body>
18 </html>
```

کدهای بالا را با نام **radioButton.php** در پوشه **www** ذخیره و اجرا کنید. فرمی به صورت زیر در مرورگر مشاهده می کنید :



برای دریافت مقدار از دکمه رادیویی در **PHP** باید خاصیت **Name** آن را در اختیار داشته باشیم. سپس با استفاده از خاصیت **Name**به مقدار خاصیت **Value** آن دست یابیم. همانطور که در خطوط **9** و **10** کد بالا مشاهده می کنید دو دکمه رادیویی با مقدار خاصیت **Name** یکسان **gender** در اختیار داریم. مقدار خاصیت **Value** یکی از آنها **male** و دیگری **female** است. حال قرار است ما این دو مقدار را به دست آوریم. پس کد زیر را در بخش **Head** کد بالا بنویسید:

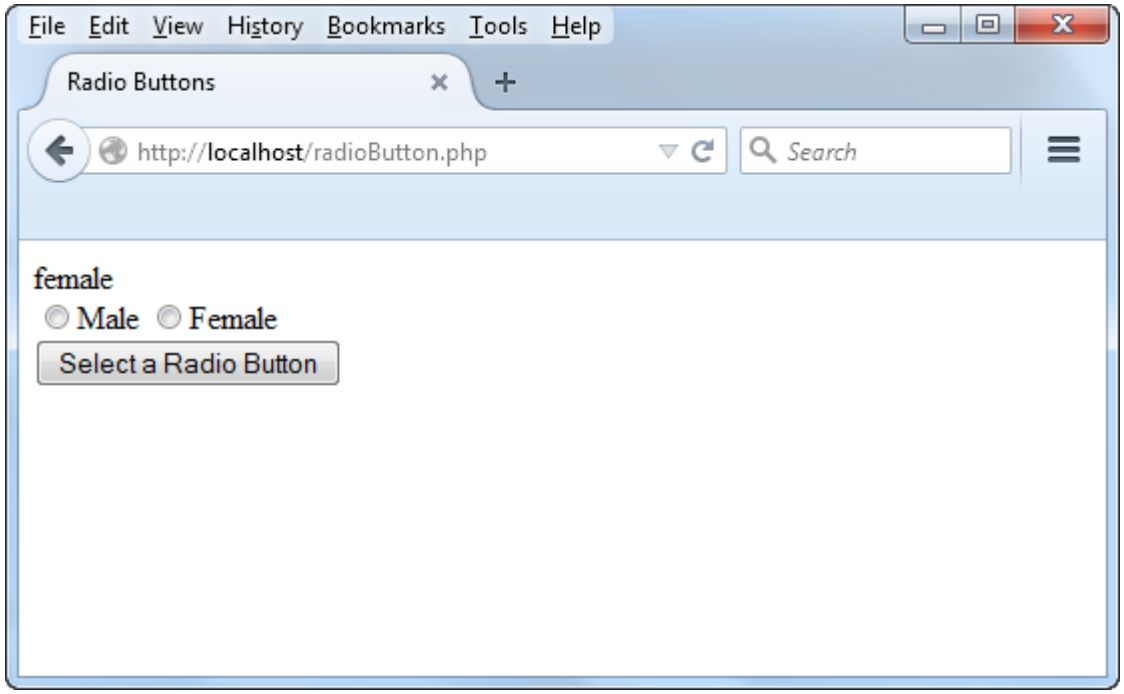
```
<?php
    if (isset($_POST['Submit1']))
    {
        $selected_radio = $_POST['gender'];
        print $selected_radio;
    }
?>
```

تا کد نهایی به صورت زیر در آید. برنامه را اجرا کنید:

```
1  <html>
2  <head>
3  <title>Radio Buttons</title>
4
5  <?php
6      if(isset($_POST['Submit1']))
7      {
8          $selected_radio = $_POST['gender'];
```

```
9           print $selected_radio;
10      }
11  ?>
12
13  </head>
14  <body>
15
16      <Form name ="form1" Method ="Post" ACTION ="radioButton.php">
17
18          <Input type = 'Radio' Name ='gender' value= 'male'>Male
19          <Input type = 'Radio' Name ='gender' value= 'female'>Female
20          <br/>
21          <Input type = "Submit" Name = "Submit1" Value = "Select a Radio
22  Button">
23
24      </FORM>
25
26  </body>
27  </html>
```

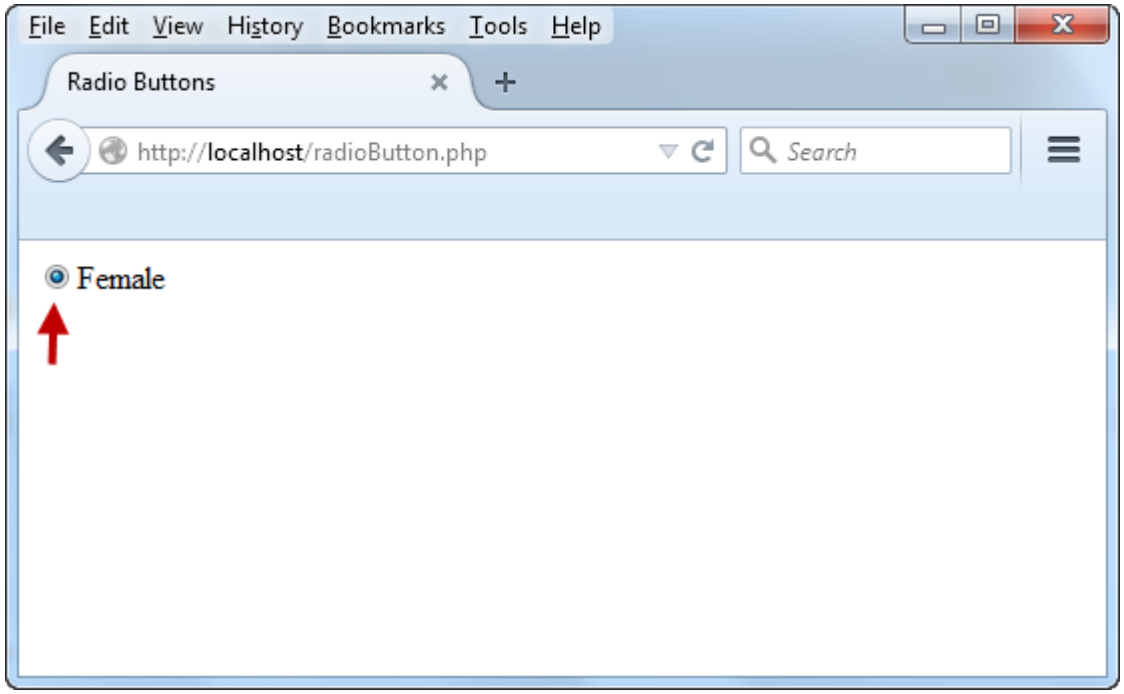
همانطور که در کد بالا مشاهده می کنید ابتدا در خط 6 چک می کنیم که آیا دکمه ای با نام Submit1 که همان دکمه ارسال است فشرده شده است یا نه؟ اگر به یاد داشته باشید گفتیم که آرایه های $_POST و $_GET تلفیقی از خاصیت Name و Value عناصر HTML هستند. به طوریکه اگر خاصیت Name یک عنصر را به آنها بدهید مقدار خاصیت Value عنصر را به شما می دهند. پس در خط 8 مقدار خاصیت Name دکمه را به $_POST می دهیم و مقدار خاصیت Value آن را در یک متغیر قرار می دهیم و در خط 9 چاپ می کنیم. حال برنامه را اجرا کرده و با انتخاب دکمه رادیوی و زدن بر روی دکمه ارسال نتیجه را مشاهده کنید:



در تصویر بالا مشاهده می کنید که با وجودیکه دکمه رادیویی **female** را انتخاب و بر روی دکمه ارسال کلیک کرده ایم ولی تیک دکمه رادیویی از بین رفته است و این یک مشکل است. دکمه های رادیویی برای اینکه دارای تیک باشند دارای خاصیتی به نام **checked** هستند که وقتی این خاصیت را می نویسیم نقطه ای در وسط دکمه ظاهر می شود که به معنای فعال بودن آن است:

```
<Input type = 'Radio' Name ='gender' value= 'female' checked >Female
```

خروجی کد بالا به شکل زیر است:



برای اضافه کردن این خاصیت کد اصلی را به صورت زیر تغییر دهید:

```
<html>
<head>
<title>Radio Buttons</title>
</head>

<?PHP

    $male_status = 'unchecked';
    $female_status = 'unchecked';

    if (isset($_POST['Submit1']))
    {
          $selected_radio = $_POST['gender'];

          if ($selected_radio == 'male')
            {
                $male_status = 'checked';
          }
          else if ($selected_radio == 'female')
            {
                $female_status = 'checked';
          }
    }

    ?>
```

```
26
27  <body>
28
29      <FORM NAME ="form1" METHOD ="POST" ACTION ="radioButton.php">
30      <INPUT TYPE = 'Radio' Name ='gender'  value= 'male' <?PHP print
31  $male_status;?>>Male
32      <INPUT TYPE = 'Radio' Name ='gender'  value= 'female'<?PHP print
33  $female_status; ?>>Female
34      <br />
35      <INPUT TYPE = "Submit" Name = "Submit1"  VALUE = "Select a Radio
36  Button">
37      </FORM>

    </body>
    </html>
```

در کد بالا و در خطوط 8 و 9 دو متغیر تعریف و آنها را با مقدار **unchecked**مقداردهی کرده ایم. سپس با استفاده از دستور **print**در خطوط 30 و 31مقدار این دو متغیر را چاپ می کنیم. این کار باعث می شود که دو دکمه رادیویی دارای خاصیت **unchecked** (تیک نخورده) شوند. حال قبل از دستورات **if** باید یک نکته را بار دیگر توضیح دهیم. همانطور که گفتیم دو آرایه **$_POST** و **$_GET** حاوی خاصیت **Name** و **Value** عناصر **HTML** هستند یعنی مانند آرایه های **Assosiative** عمل می کنند و اگر مقدار خاصیت **Value** یک عنصر را بخواهیم باید مقدار خاصیت **Name** عنصر را در داخل کروشه بنویسیم. این کار را در خط 13 کد بالا انجام داده ایم. خاصیت **Name** دکمه های رادیویی **gender** می باشد. مقدار خاصیت **Value** دکمه ای که کاربر کلیک کرده است را با استفاده از آرایه **$_POST**به دست می آوریم و در یک متغیر **($selected_radio)** قرار می دهیم. سپس در دستور **if** (خط 15) این مقدار را با مقدار **male** چک می کنیم. اگر برابر بود مقدار متغیر **$male_status** را به **check** تغییر می دهیم. این کار باعث می شود که مقدار **checked** در خط 30 چاپ شود و دکمه رادیوی تیک بخورد. همین کار را در قسمت **elseif** (خط 19) هم انجام می دهیم. حال که با نحوه کار ب دکمه های رادیویی آشنا شدید در درس بعد در مورد چک باکس ها توضیح می دهیم.

چک باکس (checkbox)

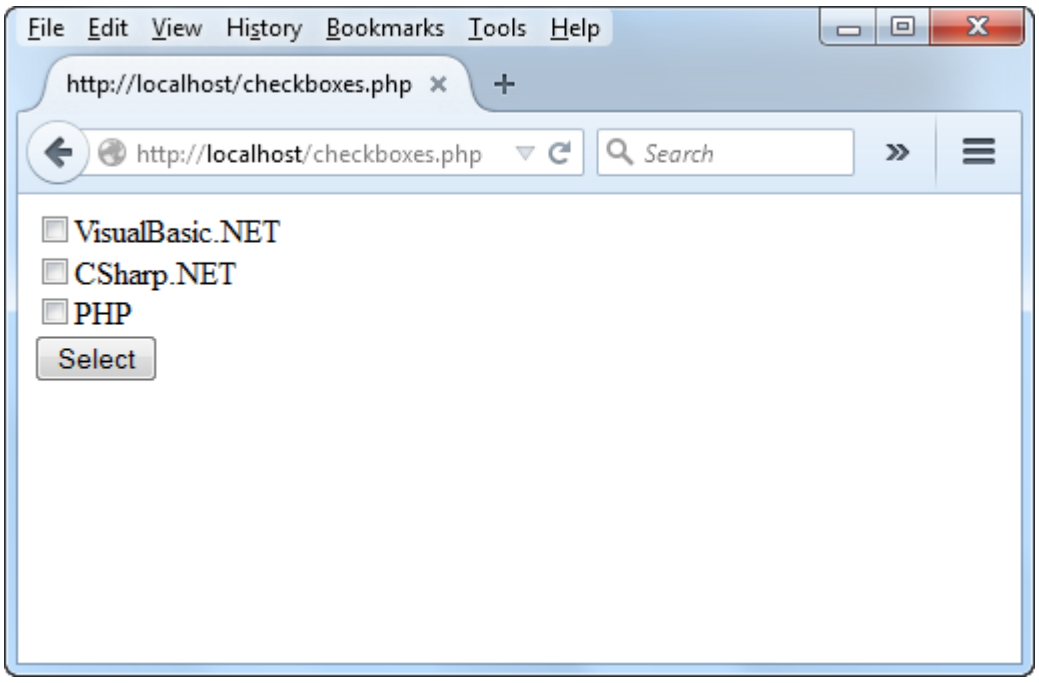
CheckBox یک دکمه است و به شکل یک جعبه ی خالی به همراه یک برچسب در کنار آن نمایش داده می شود. در حالت عادی ،زمانی که بر روی جعبه ی خالی کلیک شود ، یک تیک در داخل جعبه نمایان می شود که به ما می گوید کنترل **CheckBox** در حالت **Checked** قرار دارد. برخلاف دکمه ی **Radio** که فقط اجازه ی انتخاب یکی از **Radio** های فرم را به ما می داد، شما می توانید چند عدد **CheckBox** و یا همه ی آنها را تیک بزنید. برای ایجاد چک باکس به صورت زیر عمل می شود :

```
<input type="checkbox" >
```

برای آشنایی کار با چک باکس ها به کد زیر توجه کنید:

```
<form action="" method="POST">
    <input type="checkbox" name="Programs[]"
value="VB.NET">VisualBasic.NET<br />
    <input type="checkbox" name="Programs[]" value="C#.NET">CSharp.NET<br />
    <input type="checkbox" name="Programs[]" value="PHP">PHP<br />
    <input type="submit" name="submit" value="Select">
</form>
```

در کد بالا ما یک فرم به همراه سه چک باکس و یک دکمه ارسال مانند شکل زیر ایجاد کرده ایم:



برای کار با چک باکس ها همانطور که در کد بالا مشاهده می کنید بهتر است که نام آنها را به صورت آرایه تعریف کنید تا راحت تر بتوانید با آنها کار کنید. حال در پایین کدهای بالا کدهای زیر را بنویسید:

```
1  <?php
2
3        if (isset($_POST['submit']))
4        {
5              if(isset($_POST['Programs']))
6              {
7                    echo'You selected: <br />';
8                    foreach ($_POST['Programs'] as $a)
9                    {
10                   echo"<i>$a</i><br />";
11                   }
12             }
13             else
14             {
15             echo'Nothing selected';
16             }
17       }
18
19 ?>
```

همانطور که در کد بالا مشاهده می کنید از آنجاییکه چک باکس ها را به صورت آرایه تعریف کرده ایم می توانیم با استفاده از دستور foreach (خط 8) مقادیری را که تیک خورده اند را چاپ کنیم.

لیست کشویی (Select)

برای ساختن لیستهای کشویی که به کاربران اجازه می دهد یک یا چند گزینه موجود در لیستی را در یک فرم انتخاب کند باید از تگ <select>استفاده کنیم. گزینه های موجود در این تگ بین <option> و <option/> قرار می گردند و پس از گزینه ها تگ پایانی لیست به صورت <select/> نوشته می شود.

```
<select>
    <option>Jack  </option>
    <option>Rose </option>
    <option>John </option>
</select>
```

همانطور که اشاره شد از آنجاییکه در PHP با خاصیت های name و value سر و کار داریم کد بالا را به صورت زیر تغییر می دهیم:

```
<select name = "person" >
    <option value ="jack">  Jack </option>
    <option value = "rose">Rose </option>
    <option value ="john">  John </option>
</select>
```

در لیست های کشویی باید یک نام به تگ select اختصاص داده و مقادیر value را هم برای option ها مشخص کنید. حال کد بالا را به صورت زیر و در داخل تگ form بنویسید:

```
1  <form action = "" method="post">
2
3      <select name = "person" >
4          <option value = "jack" > Jack </option>
5          <option value = "rose" > Rose </ option>
6          <option value = "john" > John </option>
7      </select>
8
9      <input type = "submit" name="submit" value="Click Me !"/>
10
11 </form>
12
13 <?php
14     if(isset($_POST['submit']))
15     {
16         echo $_POST['person'];
17     }
18 ?>
```

در کد بالا یک فرم (خطوط 11-1) و در داخل آن یک لیست کشویی (خطوط 7-3) و یک دکمه ارسال (خط 9) تعریف کرده ایم. در قسمت شرطی (خط 14) می نویسیم زمانی که دکمه **submit** توسط کاربر کلیک شد مقدار گزینه ای از لیست کشویی که توسط کاربر انتخاب شده است، چاپ شود(خط 16). تگ **select** یک خاصیت دیگر به نام **multiple** دارد که به شما اجازه انتخاب چند گزینه را می دهد. کد بالا را به صورت زیر تغییر دهید:

```
1  <form action = "" method="post">
2
3      <select name = "person[]" multiple="multiple">
4      <option value = "jack" > Jack </option>
5      <option value = "rose" > Rose </option>
6      <option value = "john" > John </option>
7      </select>
8
9      <input type = "submit" name="submit" value="Click Me !"/>
10
11 </form>
12
13 <?php
14     if(isset($_POST['submit']))
15     {
16        foreach($_POST['person'] as $value)
17        {
18           echo $value . '<br>';
19        }
20     }
21 ?>
```

به خاصیت **name** تگ **select** توجه کنید (خط 3). این خاصیت را به صورت آرایه در می آوریم و خاصیت **multiple** را به آن اضافه می کنیم. با این کار اگر چند گزینه توسط کاربر با استفاده از دکمه کنترل **(ctrl)** کیبورد انتخاب شود، در آرایه **person** ذخیره می شود که در این صورت باید با استفاده از یک حلقه **foreach** مقادیر آن را چاپ کنیم (خطوط 19-16).

لیست کشویی (Select)

برای ساختن لیستهای کشویی که به کاربران اجازه می دهد یک یا چند گزینه موجود در لیستی را در یک فرم انتخاب کند باید از تگ <select>استفاده کنیم. گزینه های موجود در این تگ بین <option>و </option>قرار می گردند و پس از گزینه ها تگ پایانی لیست به صورت </select>نوشته می شود.

```
<select>
    <option>Jack  </option>
    <option>Rose </option>
    <option>John </option>
</select>
```

همانطور که اشاره شد از آنجاییکه در PHP با خاصیت های name و value سر و کار داریم کد بالا را به صورت زیر تغییر می دهیم:

```
<select name = "person" >
    <option value ="jack">  Jack </option>
    <option value = "rose">Rose </option>
    <option value ="john">  John </option>
</select>
```

در لیست های کشویی باید یک نام به تگ select اختصاص داده و مقادیر value را هم برای option ها مشخص کنید. حال کد بالا را به صورت زیر و در داخل تگ formبنویسید:

```
1  <form action = "" method="post">
2
3      <select name = "person" >
4      <option value = "jack" > Jack </option>
5      <option value = "rose" > Rose </ option>
6      <option value = "john" > John </option>
7      </select>
8
9      <input type = "submit" name="submit" value="Click Me !"/>
10
11 </form>
12
13 <?php
14     if(isset($_POST['submit']))
15     {
16         echo $_POST['person'];
17     }
18 ?>
```

در کد بالا یک فرم (خطوط 1-11) و در داخل آن یک لیست کشویی (خطوط 3-7) و یک دکمه ارسال (خط9) تعریف کرده ایم. در قسمت شرطی (خط 14) می نویسیم زمانی که دکمه **submit** توسط کاربر کلیک شد مقدار گزینه ای از لیست کشویی که توسط کاربر انتخاب شده است، چاپ شود(خط 16). تگ **select** یک خاصیت دیگر به نام **multiple** دارد که به شما اجازه انتخاب چند گزینه را می دهد. کد بالا را به صورت زیر تغییر دهید:

```
1   <form action = "" method="post">
2
3       <select name = "person[]" multiple="multiple">
4       <option value = "jack" > Jack </option>
5       <option value = "rose" > Rose </option>
6       <option value = "john" > John </option>
7       </select>
8
9       <input type = "submit" name="submit" value="Click Me !"/>
10
11  </form>
12
13  <?php
14      if(isset($_POST['submit']))
15      {
16         foreach($_POST['person'] as $value)
17         {
18            echo $value . '<br>';
19         }
20      }
21  ?>
```

به خاصیت **name** تگ **select** توجه کنید (خط 3). این خاصیت را به صورت آرایه در می آوریم و خاصیت **multiple** را به آن اضافه می کنیم. با این کار اگر چند گزینه توسط کاربر با استفاده از دکمه کنترل **(ctrl)** کیبورد انتخاب شود، در آرایه **person** ذخیره می شود که در این صورت باید با استفاده از یک حلقه **foreach**مقادیر آن را چاپ کنیم (خطوط 16-19).

تابع date

از تابع date() در PHP ، برای نمایش و یا دستکاری ساعت و تاریخ استفاده می شود . از تابع date() می توانید برای انجام اموری مثل نمایش تاریخ جاری سرور ، نمایش یک تاریخ خاص ، محاسبه زمان ، ایجاد یک برچسب زمانی و ... استفاده نمایید . این تابع یکی از توابع پیش ساخته و اصلی زبان PHP است . شکل کلی تعریف و استفاده از تابع date() در PHP به صورت زیر است:

```
date(format, timestamp);
```

پارامتر format تعیین کننده نحوه نمایش تاریخ توسط تابع date() است . در این پارامتر کاراکترهای زیر را می توانید به کار ببرید:

کاراکتر	توضیحات
d	روز از ماه را به شکل عدد دورقمی نشان می دهد (از 01 تا 31)
D	روز هفته را به صورت سه حرفی نشان می دهد.
j	روز را به صورت عدد 1 یا دوحرفی بدون صفر پیش از آن نشان می دهد (از 1 تا 31)
l	روز هفته را به صورت کامل نشان می دهد.
N	روز هفته به شکل عددی نشان می دهد (از 1 تا 7)
S	پسوند تاریخ را به شکل انگلیسی نشان می دهد (st، nd، rd، th)
w	روز هفته را به شکل عددی نشان می دهد (از 0 تا 6)
z	روز از سال را نشان می دهد (از 0 تا 365)
W	شماره هفته از سال را نشان می دهد.
F	نام کامل ماه را نشان می دهد.
m	شماره ماه به شکل عددی را همراه با صفر پیش از آن نشان می دهد (از 01 تا 12)
M	نام ماه را به صورت سه حرفی نشان می دهد.
n	شماره ماه بدون صفر پیش از آن را به شکل عددی نشان می دهد (از 1 تا 12)
t	تعداد روزهای ماه را نشان می دهد.
L	سال کبیسه را نشان می دهد (اگر سال کبیسه باشد 1 درغیر این صورت 0).
o	شماره سال را به شکل عددی نشان می دهد.
Y	شماره سال را به صورت 4 حرفی نشان می دهد.

y	شماره سال را به صورت 2 حرفی نشان می دهد.
a	Am (قبل از ظهر) یا pm (بعد از ظهر) را با حروف کوچک نشان می دهد.
A	Amیا PM را با حروف بزرگ نشان می دهد.
g	ساعت را به صورت 12 ساعته و بدون صفر پیش از آن نشان می دهد (از 1 تا 12)
G	ساعت را به صورت 24 ساعته و بدون صفر پیشین نشان می دهد (از 0 تا 23)
h	ساعت را به صورت 12 ساعته و همراه با صفر پیشین نشان می دهد (از 01 تا 12)
H	ساعت را به صورت 24 ساعته و همراه با صفر پیشین نشان می دهد (از 00 تا 23)
i	دقیقه را همراه با صفر پیشین نشان می دهد (از 00 تا 59)
s	ثانیه را همراه صفر پیشین نشان می دهد (از 00 تا 59)
u	میکروثانیه را نشان می دهد.
e	منطقه زمانی را نشان می دهد (مانند gmt یاutc)
O	تفاوت زمانی با زمان گرینویچ را به ساعت نشان می دهد.
P	تفاوت زمانی با گرینویچ را به ساعت و دقیقه نشان می دهد.

از کاراکترهایی مثل / ، – ، . و ... می توان برای جدا نمودن عددهای سال و ماه و روز در تابع date است نمود. در مثال زیر نحوه به کار بردن تابع date در نمایش تاریخ و حالت های مختلف پارامتر format در آن را مشاهده می کنید:

```
<?php
    echo date("Y/m/d") . "<br />";
    echo date("g:i:s a");
?>
```

```
 2015/01/18
7:50:55 pm
```

این تاریخ و ساعت مربوط به سرور است و برای به دست آوردن تاریخ و ساعت منطقه ای (برای ما، تهران) باید از تابع () date_default_timezone_set استفاده کنیم. پس برای این منظور کد بالا را به صورت زیر تغییر دهید:

```
<?php
    date_default_timezone_set("Asia/Tehran");
    echo date("Y/m/d") . "<br />";
    echo date("g:i:s a");
?>
```

```
 2015/01/18
11:20:55 pm
```

با اجرای کد بالا، زمان 3 ساعت و نیم به جلو کشیده می شود .و اما پارامتر دوم تابع () date، مهر زمانی یا timestamp می باشد. منظور از timestamp یک سری اعداد ساده به صورت integer است .در یک توضیح جزئی تر ، این اعداد تعداد ثانیه هایی هستند که از نیمه شب ۱ ژوئن ۱۹۷۰ تا به این لحظه گذشته است (مبدا گرینویچ) . در درس بعد با timestamp بیشتر آشنا می شوید.

مهر زمانی و توابع time() و mktime()

منظور از مهر زمانی یا timestamp ، یک سری اعداد ساده به صورت integer است .در یک توضیح جزئی تر ، این اعداد تعداد ثانیه هایی هستند که از نیمه شب ۱ ژوئن ۱۹۷۰ تا به این لحظه گذشته است . برای به دست آوردن تعداد این ثانیه ها در PHP توابعی وجود دارند که در زیر به آنها اشاره شده است.

تابع time()

تابع time() تعداد ثانیه هایی هستند که از نیمه شب ۱ ژوئن ۱۹۷۰ تا به این لحظه گذشته است را نشان می دهد. برای روشن شدن مطلب کد زیر را در یک فایل نوشته واجرا کنید:

```
<?php
    echo time();
?>
```

بعد از اجرای کد بالا عددی به صورت 1423333653 (البته ممکن است برای شما متفاوت باشد) نشان داده می شود که نشان دهنده تعداد ثانیه های گذشته شده از تاریخ مذکور تا الان است. حال چندین بار مرورگر را Refresh کنید. مشاهده می کنید که با هر بار Refresh مرورگر این رقم تغییر کرده و بیشتر می شود .با به کار بردن خروجی این تابع به عنوان پارامتر دوم در تابع date() می توان یک تاریخ تولید کرد. به مثال زیر توجه کنید:

```
<?php
    $timeStamp = time();
    echo date("Y/m/d",$timeStamp);
?>
```

```
2015/02/07
```

تابع date() تعداد ثانیه ها را به یک تاریخ خوانا تبدیل می کند.

تابع MKtime()

تابع MKtime() تعداد ثانیه های گذشته تا یک تاریخ خاص را بر می گرداند.دستور استفاده از این تابع به صورت زیر است:

```
mktime(hour,minute,second,month,day,year);
```

همانطور که در بالا مشاهده می کنید پارامترهای این تابع ساعت، دقیق، ثانیه، ماه، روز و سال است. مثلا برای به دست آوردن ثانیه های یک سال بعد از تاریخ مبداء (01/01/1970) کد زیر را نوشته و اجرا کنید:

```
<?php
    echo mktime(0,0,0,1,1,1971);
?>
```

```
31536000
```

چون اختلاف تاریخ بالا با تاریخ مبدا یک سال است در نتیجه عدد 31536000 به ما نشان داده می شود.

کار با فایل ها

بیشتر اوقات لازم است که داده هایتان را در یک فایل دائمی ذخیره کنید. همچنین لازم است که به درایورها یا پوشه های کامپیوتر دسترسی داشته باشید به طوری که بتوانید فایل ها را در آنها قرار داده و ذخیره کنید، مکانی را که می خواهید فایل درآن ذخیره شود را نشان داده، فایلی که در یک پوشه ذخیره شده را مشاهده کرده و به طور خلاصه جزییاتی در مورد یک پوشه خاص را به دست آورید.

شما حتی می توانید چک کنید که اندازه فایل چقدر است و آیا فایل از نوع فقط خواندنی است یا نه؟ اگر چه استفاده از پایگاه داده امروزه بیشتر ترجیح داده می شود، اما گاهی اوقات برای برانامه های کوچک استفاده از یک فایل متنی کارا تر است. همچنین برخی از برنامه های کاربردی هنوز وجود دارند که داده هایشان را در یک فایل متنی ذخیره می کنند و بنابراین لازم است که اطلاعات در این فایل نوشته و از آن خوانده شود.

PHP دارای توابعی است که شما با استفاده از آنها می توانید محتویات فایلها را بخوانید، در داخل آنها بنویسید و صفات فایلها و پوشه ها را چک کنید. در درس های آینده در مورد این توابع مطالب بیشتری را خواهید آموخت.

به دست آوردن اطلاعات در مورد فایل

در **PHP** توابعی وجود دارند که در مورد فایل ها اطلاعات مفیدی در اختیار ما قرار می دهند. مثلا برای اینکه بفهمیم که یک فایل وجود دارد یا نه از تابع **file_exsits()** به صورت زیر استفاده می کنیم:

```
 <?php
    echo file_exists('D:/MyFolder/MyFile.txt');
?>
```

```
true
```

کافیست که مانند کد بالا مسیر فایل مورد نظر را به این تابع بدهیم، تا با برگرداندن مقدار **true** یا **false** بفهمیم که تابع یا پوشه وجود دارد یا نه؟ یا برای به دست آوردن سایز یک فایل مسیر فایل را به تابع **filesize()** می دهیم تا اندازه را بر حسب بایت به ما بدهد:

```
<?php
    echo filesize('D:/MyFolder/MyFile.txt');
?>
```

```
12
```

مقدار بازگشتی **12** یعنی اینکه فایل **12** بایت می باشد.

خاصیت مربوط به زمان

منظور از خاصیت های مربوط به زمان ، خواصی مربوط به زمان دستکاری و ایجاد فایل می باشد. در **PHP** توابعی برای به دست اوردن آخرین دستکاری و یا خوانده شدن فایل وجود دارد که کافیست همانند توابع بالا، مسیر یک فایل را به آنها بدهیم تا اطلاعات مفیدی در اختیار ما قرار دهند. در زیر به برخی از آنها شاره شده است:

- **fileatime()** : زمان آخرین دسترسی به فایل مثلا خواندن آن را به صورت مهر زمانی نمایش می دهد.
- **filectime()** : زمان آخرین تغییر در فایل مثلا زمان ایجاد و نوشتن در آن را به صورت مهر زمانی نمایش می دهد.
- **filemtime()** : زمان آخرین تغییر در فایل مثلا زمان ایجاد و نوشتن در آن را به صورت مهر زمانی نمایش می دهد.

به مثال های زیر توجه کنید:

```
<?php
    $time = fileatime('D:/MyFolder/MyFile.txt');
    echo date('F d Y H:i:s' , $time);
?>
```

```
December 01 2015 04:08:51
```

```php
<?php
    $time = filectime('D:/MyFolder/MyFile.txt');
    echo date('F d Y H:i:s' , $time);
?>
```

```
December 01 2015 04:08:51
```

```php
<?php
    $time = filemtime('D:/MyFolder/MyFile.txt');
    echo date('F d Y H:i:s' , $time);
?>
```

```
December 01 2015 03:54:26
```

به دست آوردن نام فایل و پوشه

برای استخراج نام یک فایل یا پوشه از متد basename() استفاده می کنیم:

```
<?php
    echo basename('D:/MyFolder/MyFile.txt');
?>
```

```
MyFile.txt
```

```
<?php
    echo basename('D:/MyFolder');
?>
```

```
MyFolder
```

کد اولی نام فایل و کد دوم نام پوشه را بر می گرداند.

باز و بسته کردن یک فایل

به وسیله تابع fopen()در PHP ، می توانید یک فایل را باز کنید . این تابع چهار پارامتر می گیرد. دو پارامتر اول اجباری و دو پارامتر آخر اختیاری هستند. پارامتر اول تعیین کننده نام و آدرس فایل و پارامتر دوم تعیین کننده نحوه باز کردن فایل است . در ادامه به توضیح این پارامترها خواهیم پرداخت . برای کار با این تابع ، باید خروجی آن را در یک متغیر ذخیره نمود و سپس آن متغیر را در سطح برنامه استفاده نمود . شکل کلی استفاده از این متد به صورت زیر است:

```
fopen(filename,mode,include_path,context)
```

گفتیم که پارامتر اول تعیین کننده نام و یا آدرس فایل مورد نظر است . مثلا برای باز کردن یک فایل مثلا Test.txt که در درایو C:قرار دارد، به صورت زیر عمل می کنیم:

```
$fileName = fopen("C:\Test.txt");
```

اما پارامتر دوم تعیین کننده نحوه باز شدن فایل است . این پارامتر می تواند یکی از مقادیر جدول زیر را داشته باشد . ضمنا عملکرد هر مقدار نیز توضیح داده شده است:

مقدار	نحوه باز کردن فایل
r	در این حالت فایل به صورت فقط خواندنی و از ابتدای آن باز می شود . در این حالت امکان تغییر فایل برای کاربر وجود ندارد.
+r	در این حالت فایل به صورت خواندنی ، قابل ویرایش و از ابتدا باز می شود . در این حالت امکان تغییر فایل برای کاربر وجود دارد.
w	در این حالت فایل به صورت فقط نوشتنی و قابل تغییر باز می شود . در این حالت چنان فایل از قبل وجود داشته باشد ، محتویات آن پاک می شود . و اگر وجود نداشته باشد ، یک فایل جدید ایجاد می شود.
+w	در این حالت فایل به صورت خواندنی و نوشتنی و قابل تغییر باز می شود . در این حالت چنان فایل از قبل وجود داشته باشد ، محتویات آن پاک می شود . و اگر وجود نداشته باشد ، یک فایل جدید ایجاد می شود.
a	در این حالت فایل باز شده و متن تعیین شده به انتهای آن اضافه می شود.
x	در این حالت یک فایل جدید با نام تعیین شده ایجاد و به صورت فقط نوشتنی باز می شود . چنانچه فایل از قبل وجود داشته باشد ، برنامه پیام error صادر کرده و مقدار false را بر می گرداند.
+x	در این حالت یک فایل جدید با نام تعیین شده ایجاد و به صورت خواندنی و نوشتنی باز می شود . چنانچه فایل از قبل وجود داشته باشد ، برنامه پیام errorصادر کرده و مقدار false را بر می گرداند.

پس برای باز کردن یک فایل مثلا Test.txt که در درایو C: قرار دارد، اگر بخواهیم فقط آن را بخوانیم باید به صورت زیر عمل کنیم:

```
$fileName = fopen("C:\Test.txt","r");
```

بعد از اینکه فایل را باز کرده و کارهایی را که لازم داریم بر روی آن انجام دادیم لازم است آن را ببندیم. این کار را با استفاده از تابع fclose()انجام می دهیم. نحوه استفاده از این تابع به صورت زیر است:

```
fclose(file)
```

این تابع یک آرگومان می گیرد که نام فایل است. البته می توانیم نام متغیری که در هنگام باز کردن فایل به کار بردیم و محتویات فایل را درون آن ذخیره نمودیم. مثلا فایل باز شده مثال بالا را می توان به صورت زیر بست:

```
<?php
    $fileName  = fopen("C:\Test.txt","r");

    fclose($fileName);
?>
```

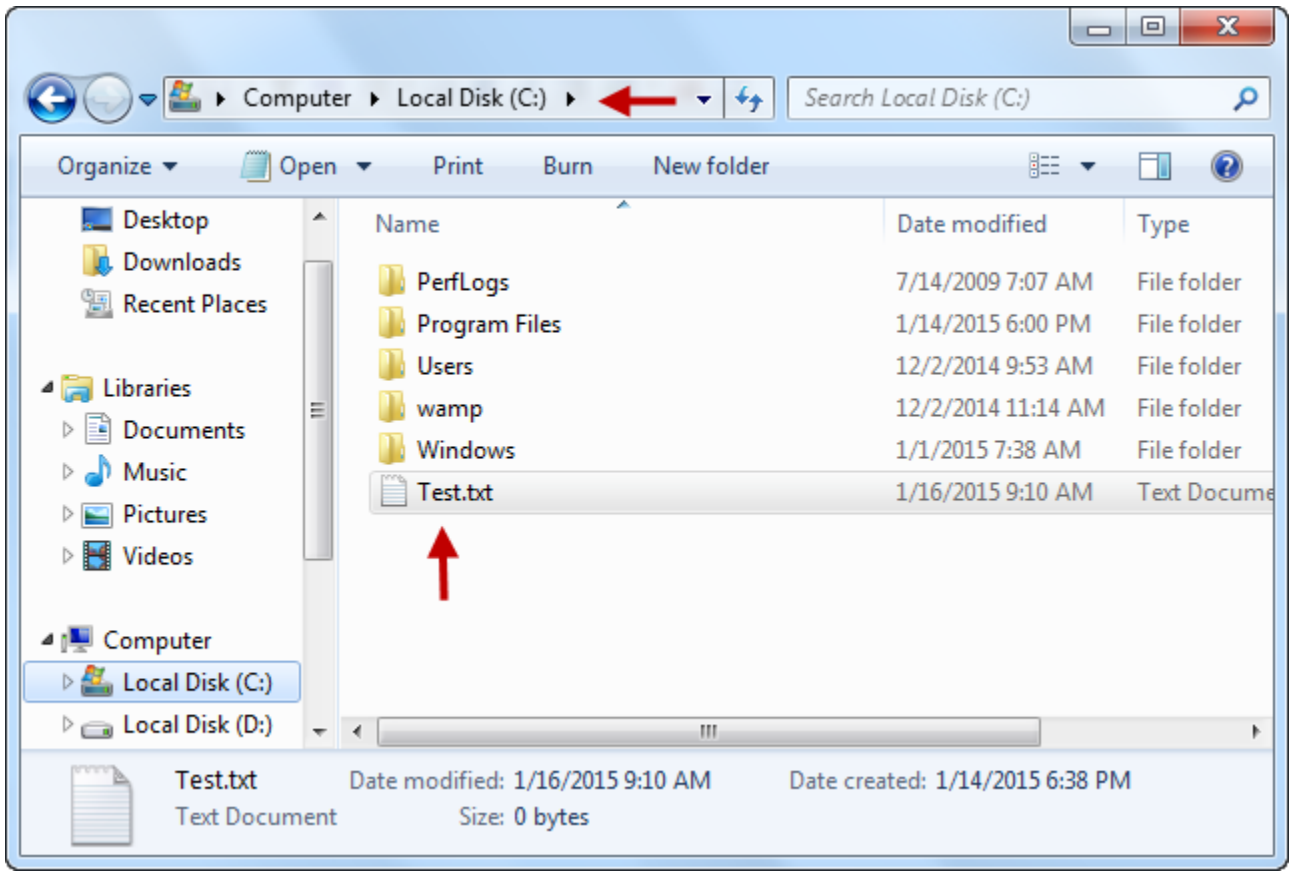
در این درس شما با نحوه و حالت های مختلف باز کردن یک فایل آشنا شدید، اما فعلا درک کاملی از کاربرد موارد ذکر شده ندارید. در درس های آینده با نحوه استفاده از آنها آشنا می شوید.

نوشتن در فایل

برای نوشتن در یک فایل از تابع **fwrite()** استفاده می شود. نحوه استفاده از این تابع به صورت زیر است:

```
fwrite(file,string,length)
```

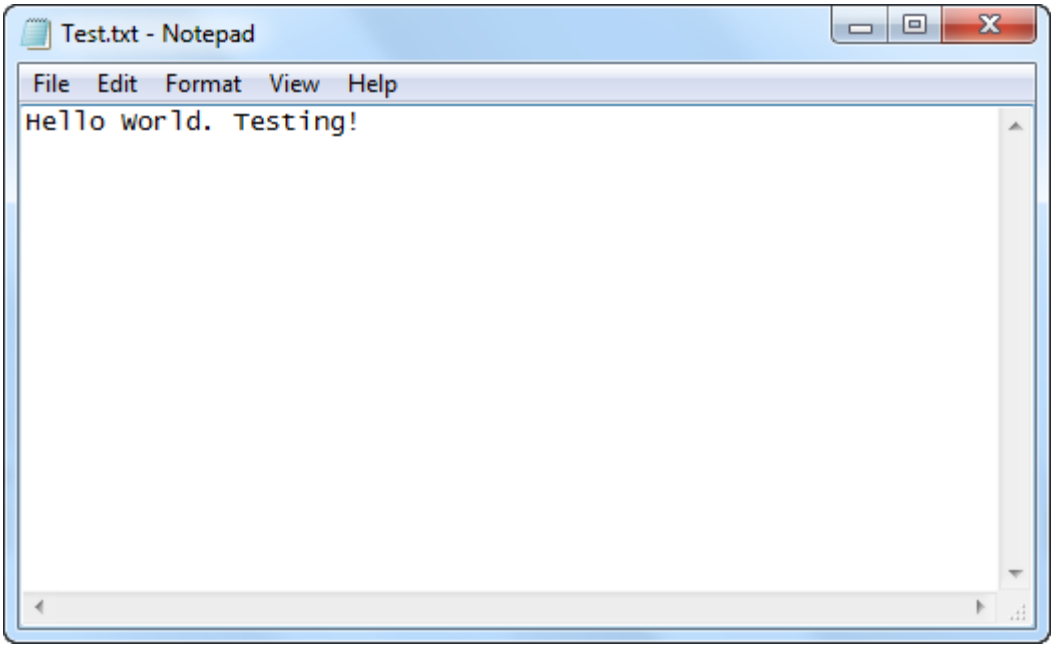
این تابع سه آرگومان می گیرد. که دو آرگومان اولی اجباری و آخرین آرگومان اختیاری است. اولین آرگومان نام فایلی که قرار است در آن بنویسیم، دومین آرگومان متنی که قرار است در فایل نوشته شود و سومین آرگومان هم تعداد کاراکترهایی که می خواهیم در فایل نوشته شوند.خروجی این تابع از نوع عدد است و نشان دهنده تعداد کاراکترهایی است که در فایل نوشته شده اند. فرض کنید که یک فایل با نام **Test.txt** در داخل درایو **C:** داریم:



و می خواهیم متن **Hello World. Testing!** را در داخل آن بنویسیم. برای این کار به صورت زیر عمل می کنیم:

```
1  <?php
2      $fileName = fopen("c:\Test.txt","w");
3      fwrite($fileName,"Hello World. Testing!");
4      fclose($file);
5  ?>
```

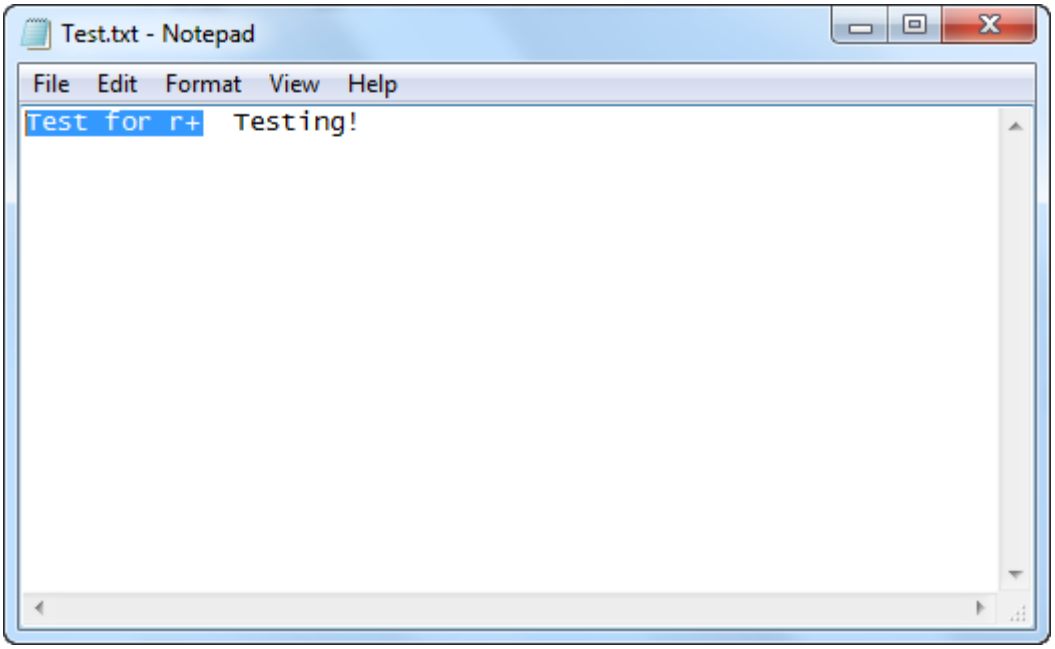
همانگونه که در کد بالا مشاهده می کنید ابتدا فایل را در حالت w (حالت نوشتنی) باز می کنیم و در خط 3 نام متغیری که در خط 2 تعریف کرده ایم که همان نام فایل است و همچنین متنی که قرار است در داخل فایل نوشته شود را به عنوان آرگومان به تابع fwrite() می دهیم و در آخر بعد از اتمام کار فایل ر می بندیم. با اجرای کد بالا متن Hello World. Testing! در داخل فایل Test.txt نوشته می شود.



حال نحوه باز کردن فایل به صورت r+ و نوشتن در آن را تست می کنیم. خطوط 2 و 3 کد بالا را به صورت زیر تغییر دهید:

```
$fileName = fopen("c:\Test.txt","r+");
fwrite($fileName,"Test for r+ ");
```

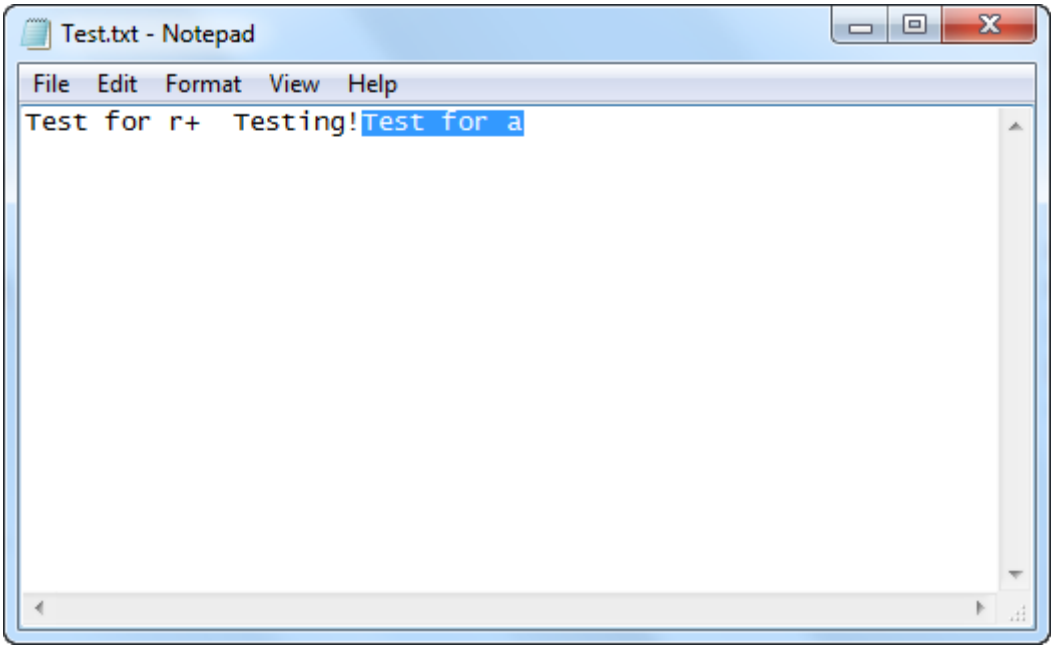
با اجرای کد بالا تابع fwrite() شروع به نوشتن رشته فعلی بر روی رشته های قبلی موجود در فایل می کند. یا به طور واضحتر به تعداد کاراکترهای رشته جدید از رشته قبلی حذف و رشته جدید در جایگزین می کند. برای روشن شدن مطلب به شکل زیر توجه کنید:



روش دیگر باز کردن فایل ها استفاده از a به عنوان آرگومان دوم است. خطوط 2 و 3 کد اصلی را به صورت زیر تغییر دهید:

```
$fileName = fopen("c:\Test.txt","a");
fwrite($fileName,"Test for a ");
```

با اجرای کد بالا متن Test for a به آخر متن موجود در فایل Test.txt اضافه می شود و شکل نهایی محتویات فایل به صورت زیر در می آید:



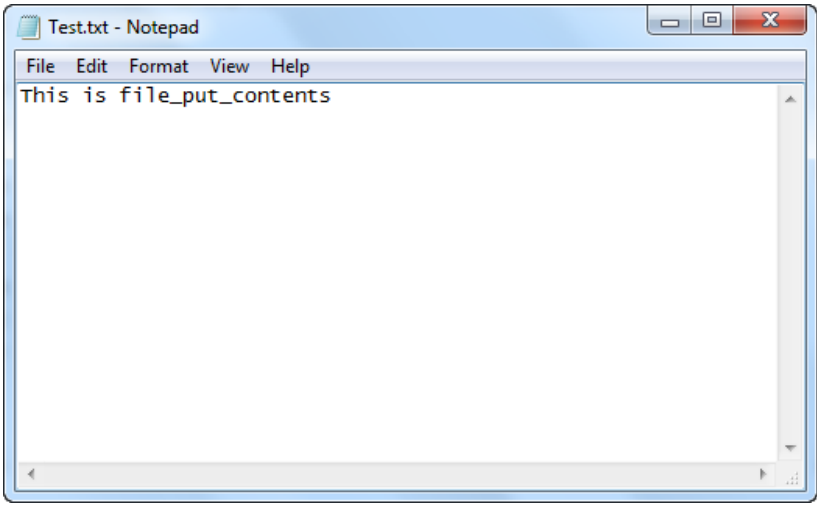
یکی دیگر از روش های نوشتن در یک فایل استفاده از تابع **file_put_contents()** است. و دستور استفاده از آن به صورت زیر می باشد:

```
file_put_contents(file,data,mode,context)
```

file در کد بالا همان فایلی است که می خواهیم چیزی در ان بنویسیم و **data** هم داده هایی هستند که می خواهیم در فایل نوشته شوند. مثلا در مثال همین درس فرض کنید اگر بخواهیم به فایل **Test.txt** به وسیله ی این تابع جمله ی **"This is file_put_contents"** را اضافه کنیم، باید به صورت زیر عمل کنیم:

```
<?php
    file_put_contents("c:\Test.txt","This is file_put_contents");
?>
```

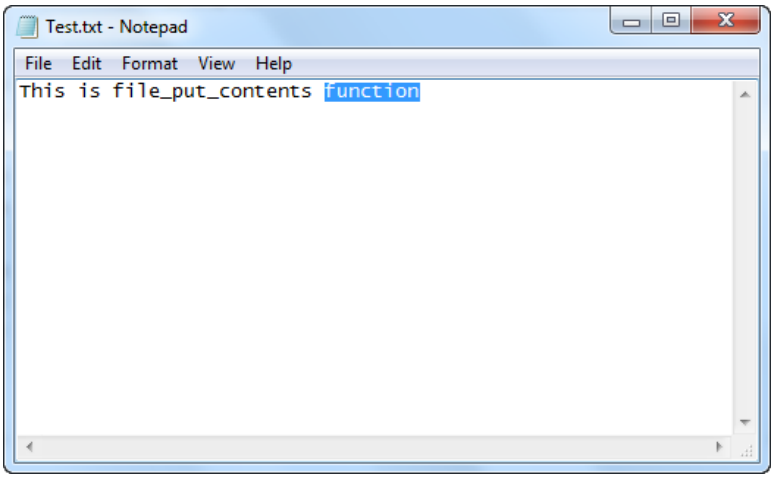
اگر کد بالا را ذخیره و اجرا کنید، خروجی به صورت زیر خواهد بود:



همانطور که در شکل بالا مشاهده می کنید این تابع در حالت عادی محتویات فایل را حذف و داده های جدید را اضافه می کند. برای جلوگیری از حذف شدن اطلاعات قبلی این تابع آرگومان سومی را هم قبول می کند. این آرگومان **FILE_APPEND** می باشد که اگر آن را هم به کد بالا اضافه کنیم داده های جدید به انتهای داده های قبلی اضافه می شوند:

```
<?php
    file_put_contents("c:\Test.txt"," function",FILE_APPEND);
?>
```

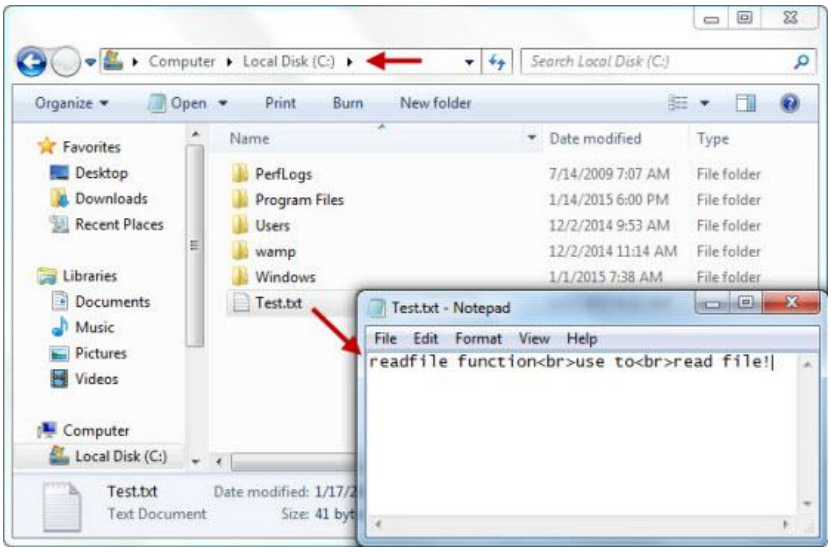
اگر کد بالا را ذخیره و اجرا کنید کلمه **function** به انتهای فایل اضافه می شود:



اکنون که با روش های نوشتن در فایل آشنا شدید در درس بعد شما را با نحوه خواندن محتویات فایل آشنا می کنم.

خواندن از فایل

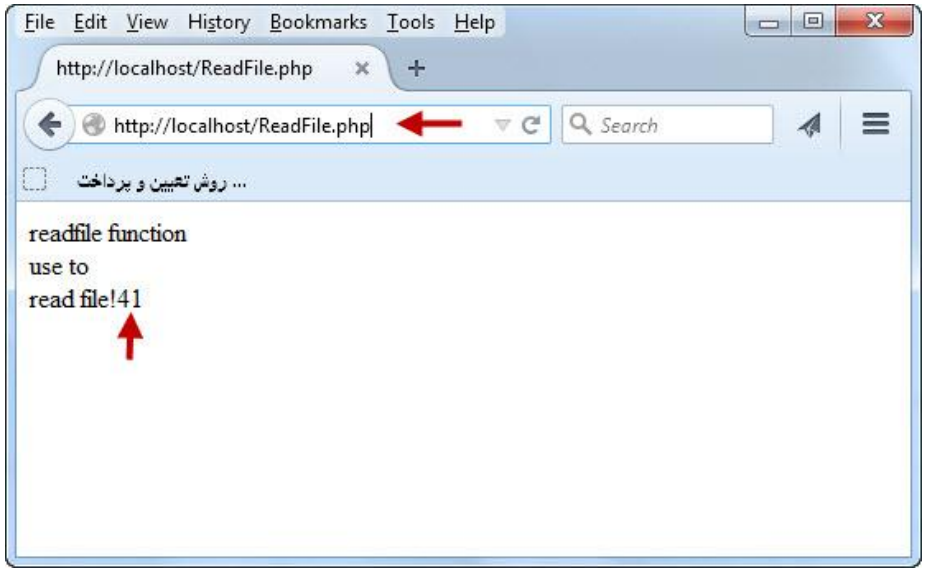
در **PHP** توابعی وجود دارند که از انها برای خواندن محتویات یک فایل استفاده می شود. برخی از این توابع محتویات را به صورت یکجا، برخی کاراکتر به کاراکتر و بعضی دیگر خط به خط می خوانند. حال به توضیح هر یک از آنها می پردازیم. فرض کنید که یک فایل با نام **Test.txt** در درایو **C:** که متن زیر در داخل آن نوشته شده است دارید:



ساده ترین روش برای خواندن محتویات فایل بالا استفاده از دو تابع **readfile()**و **file_get_contents()** می باشد. ابتدا می خواهیم محتویات فایل را با استفاده از تابع **readfile()** بخوانیم. به کد زیر توجه کنید:

```
<?php
    $fileContent = readfile("c:\Test.txt");
    echo $fileContent;
?>
```

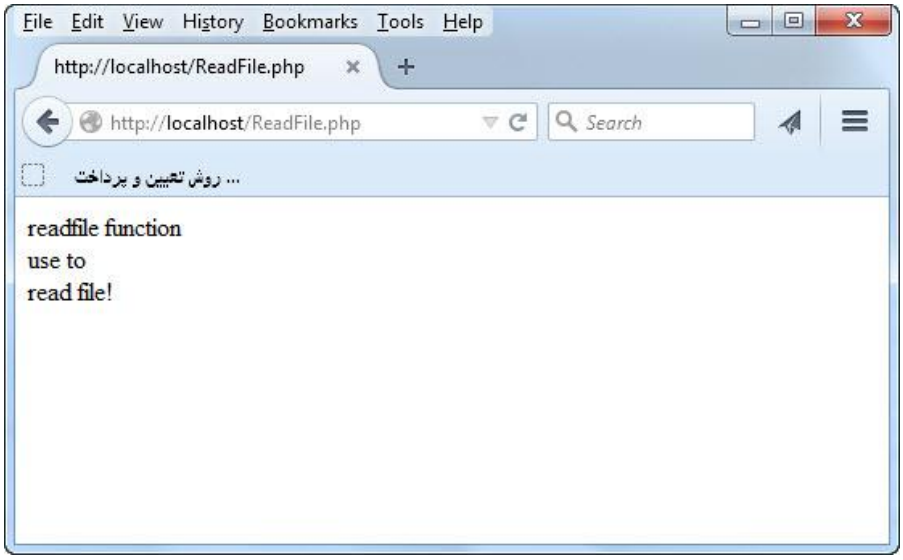
کد بالا را با نام **ReadFile** در پوشه **www** ذخیره و اجرا کنید:



**به عدد 41 در شکل بالا توجه کنید این عدد تعداد کاراکترهایی است که توسط این تابع خوانده شده است، که تعداد کل کاراکترها با احتساب فضاهای خالی می باشد. دلیل در سه خط نوشته شدن فایل اصلی هم ، وجود تگ تگ
می باشد. اکنون با استفاده از تابع file_get_contents() این کار را انجام می دهیم. کد بالا را به صورت زیر تغییر دهید:**

```
<?php
    $fileContent = file_get_contents("c:\Test.txt");
    echo $fileContent;
?>
```

فایل را ذخیره و اجرا کنید:



همانطور که در شکل بالا مشاهده می کنید این تابع فقط محتویات فایل را می خواند و تعداد کاراکترهای خوانده شده را نمایش نمی دهد. لازم به ذکر است که هر دو کد بالا را می توان به صورت خلاصه نیز نوشت:

```
<?php
    echo readfile("c:\Test.txt");
?>
```

```
<?php
    echo file_get_contents("c:\Test.txt");
?>
```

همانطور که اشاره شد، توابعی هستند که کار خواندن فایل را به صورت کاراکتر به کاراکتر و یا خط به خط انجام می دهند. برای استفاده از این توابع ابتدا باید فایل را باز کنید و سپس با استفاده از یک حلقه و این توابع کار خواندن فایل را انجام دهید. به کد زیر توجه کنید:

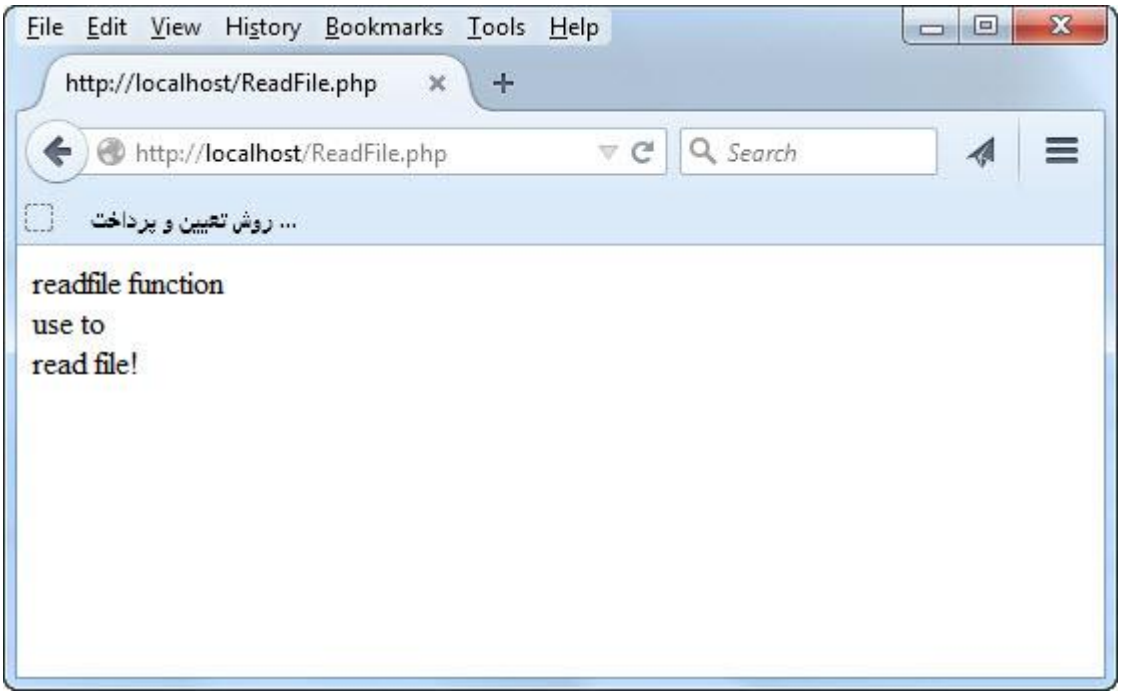
```
1  <?php
2       $fileName = fopen("c:\Test.txt","r");
3       while(!feof($fileName))
4       {
5            $readFile = fgetc($fileName);
6            echo $readFile;
7       }
8  ?>
```

در کد بالا و در خط 2 ابتدا فایل را باز کرده ایم. در خط 3 و در شرط حلقه تابع **feof()** را به کار برده ایم. کار این تابع این است که چک می کند که آیا برنامه به آخر یک فایل رسیده است یا خیر؟ در کل در خط 3 می گوییم که تا زمانی که برنامه به انتهای فایل نرسیده است عمل تکرار را انجام بده .عمل تکراری در برنامه هم در خط 5 آماده است. در این خط با استفاده از تابع **fgetc()** تک تک کاراکترهای داخل فایل را می خوانیم و در داخل متغیری قرار می دهیم و در خط 6 مقدار این متغیر را چاپ می کنیم. برای خواندن یک فایل به صورت خط به خط هم کافیست که به جای تابع **fgetc()** از تابع **fgets()** به صورت زیر اسفاده کنید:

```
1  <?php
2       $fileName = fopen("c:\Test.txt","r");
3       while(!feof($fileName))
4       {
5            $readFile = fgets($fileName);
6            echo $readFile;
```

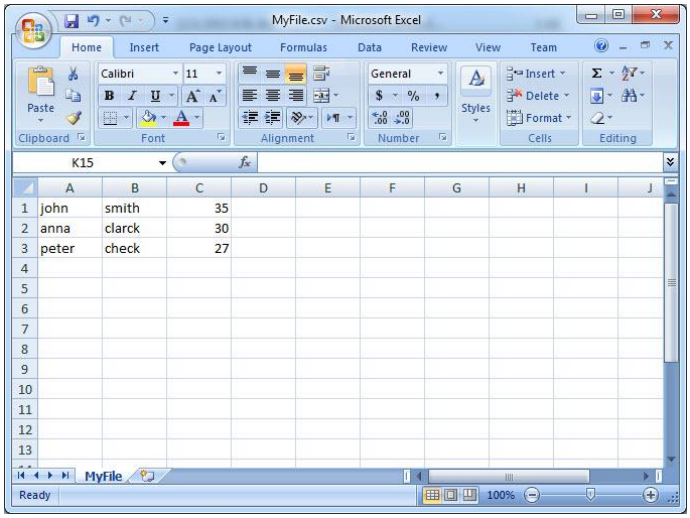
```
7       }
8   ?>
```

خروجی هر دو کد بالا به صورت زیر است:

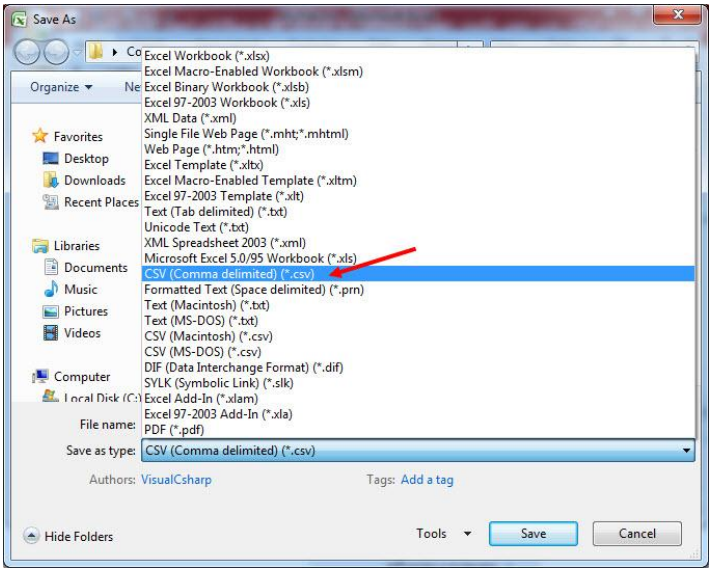


خواندن فایل CSV

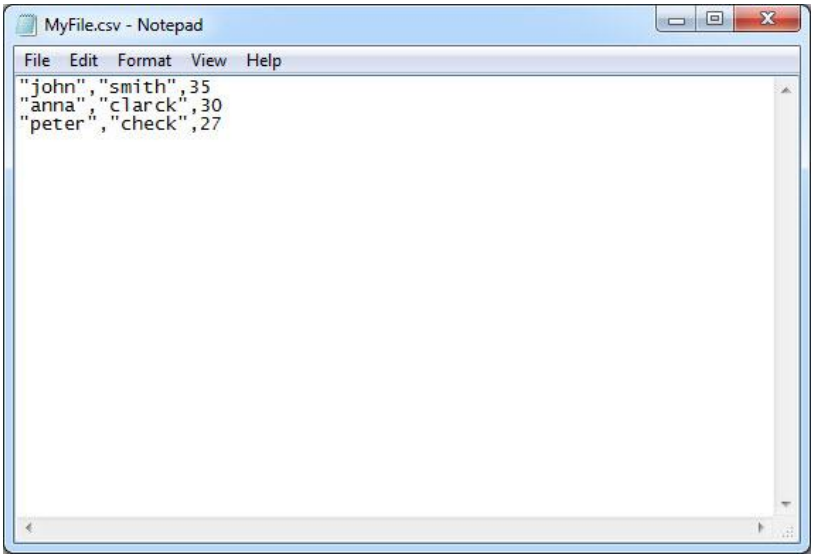
CSVیک نوع فایل بوده که مقادیر آن به وسیله کاما از هم جدا شده اند .CSV مخفف کلمات Comma Separated Valuesمی باشد. برخی نرم افزارها مانند Excel به شما اجازه ذخیره فایل هایی با این پسوند را می دهند. در این درس می خواهیم در مورد خواندن این فایل ها توضیح دهیم. عکس زیر سطرها و ستونهای نرم افزار اکسل را نشان می دهد:



در عکس زیر هم نحوه ذخیره فایل بالا با پسوند CSV نمایش داده شده است:



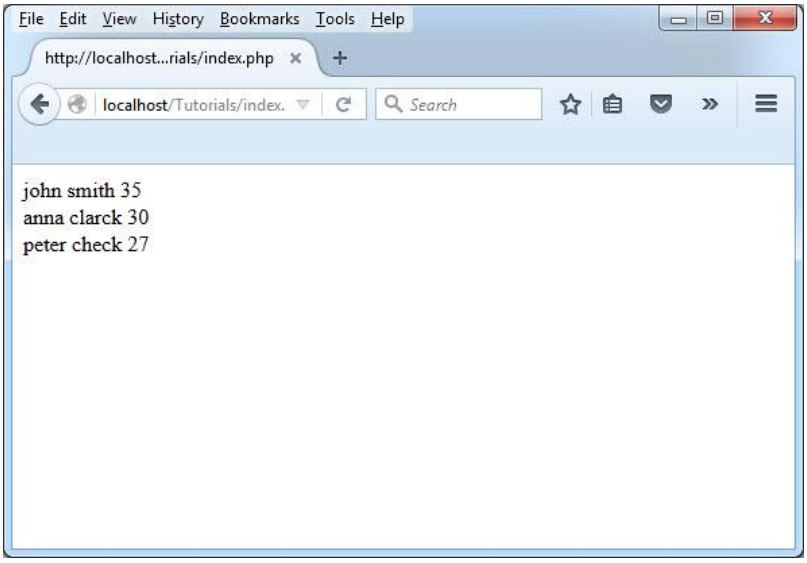
حال اگر با یک ویرایشگر متن فایل بالا را باز کنیم، چیزی شبیه به این را مشاهده خواهیم کرد:



همانطور که مشاهده می کنید مقادیر در فایل **CSV** بالا با کاما از هم جدا شده اند. تابعی در **PHP** با نام **()fgetcsv** وجود دارد که به شما اجازه می دهد با این نوع فایل ها کار کنید. عملکرد این تابع شبیه به تابع **()fgetc** بوده با این تفاوت که مقادیر موجود در هر خط را بر اساس کاما از هم جدا کرده و در داخل یک آرایه قرار می دهد. برای خواندن مقادیر موجود در فایل **CSV** بالا ابتدا فایل را باز کرده و سپس با استفاده از یک حلقه و تابع **()fgetcsv** مقادیر را چاپ می کنیم:

```
<?php
    $MyCSVFile = fopen('D:/MyFolder/MyFile.csv','r');

    while($LINE = fgetcsv($MyCSVFile,0,','))
    {
        echo "{$LINE[0]} {$LINE[1]} {$LINE[2]} <br/>";
    }

    fclose($MyCSVFile);
?>
```

در کد بالا ابتدا فایل **MyFile.csv** با تابع **()fopen** باز می کنیم. سپس در قسمت شرط حلقه هر تابع **()fgetcsv** هر خط از فایل را به یک آرایه تبدیل می کند. و سپس در داخل حلقه مقادیر آرایه را چاپ می کنیم.

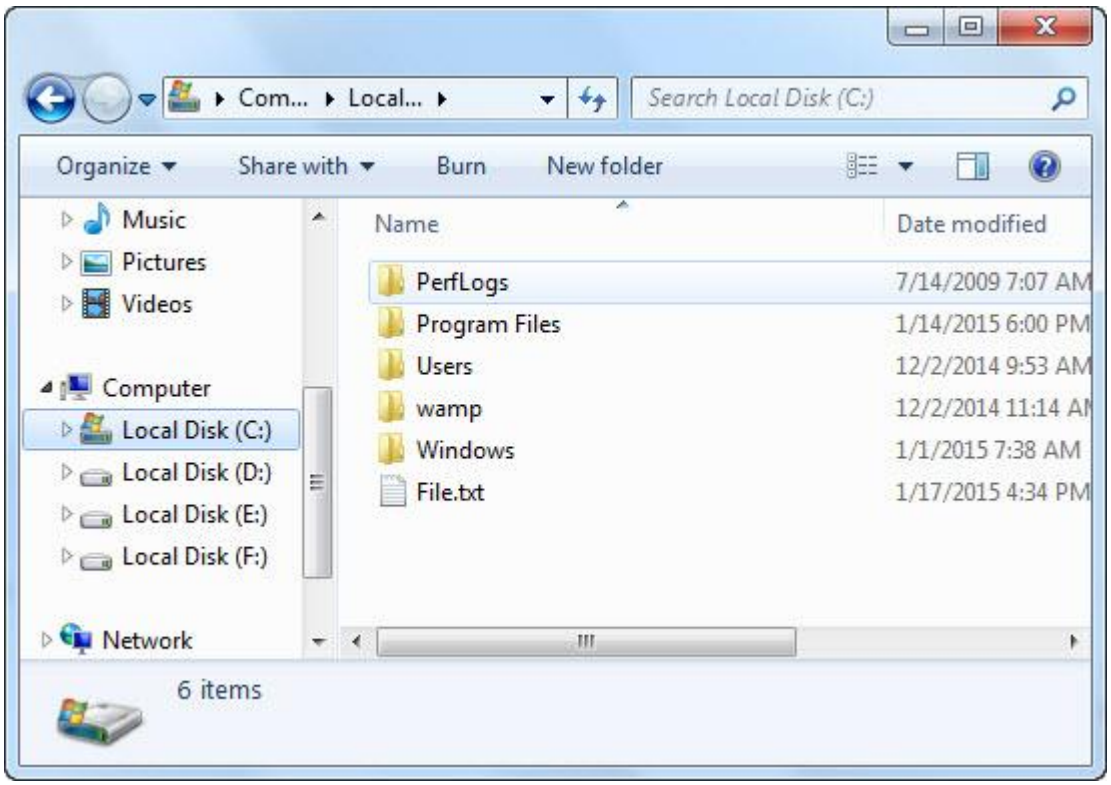

File Edit View History Bookmarks Tools Help
http://localhost...rials/index.php
localhost/Tutorials/index.
Search
john smith 35
anna clarck 30
peter check 27

ایجاد، حذف، کپی، برش و تغییر نام فایل ها

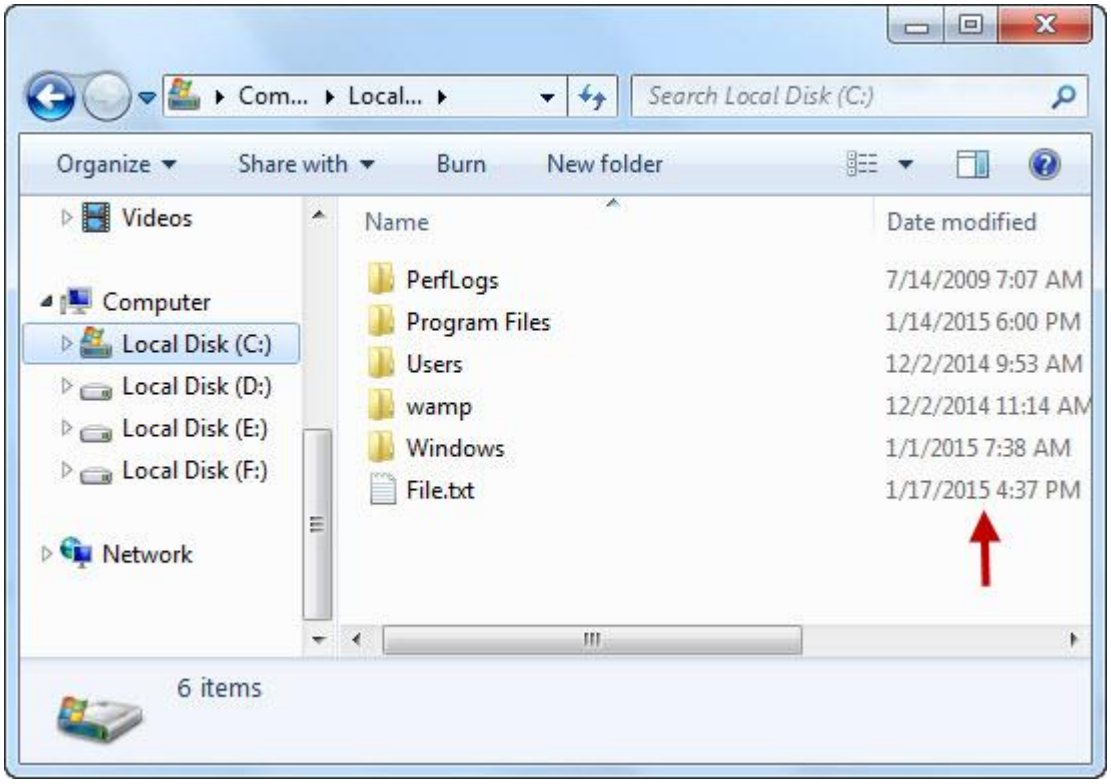
برای ایجاد یک فایل از تابع **touch()** استفاده می شود. این تابع یک مسیر و نام از شما دریافت و فایل را ایجاد می کند. مثلا برای ایجاد فایلی با نام **File.txt** در درایو **C:** به صورت زیر عمل کنید:

```
<?php
    touch("C:\File.txt");
?>
```

کد بالا را با نام **CreateFile.php** در پوشه **www** ذخیره و اجرا کنید.بعد از اجرای کد بالا اگر به درایو **C** بروید مشاهده می کنید که فایلی با همین نام ایجاد شده است:



اگر فایلی با همین نام از قبل وجود داشته و دارای محتویاتی نیز باشد آن را حذف نمی کند و فقط تاریخ ویرایش آن را تغییر می دهد:



برای حذف یک فایل از تابع unlink() استفاده می شود. برای آشنایی با کاربرد این تابع کد بالا را به صورت زیر تغییر دهید:

```
<?php
    unlink("C:\File.txt");
?>
```

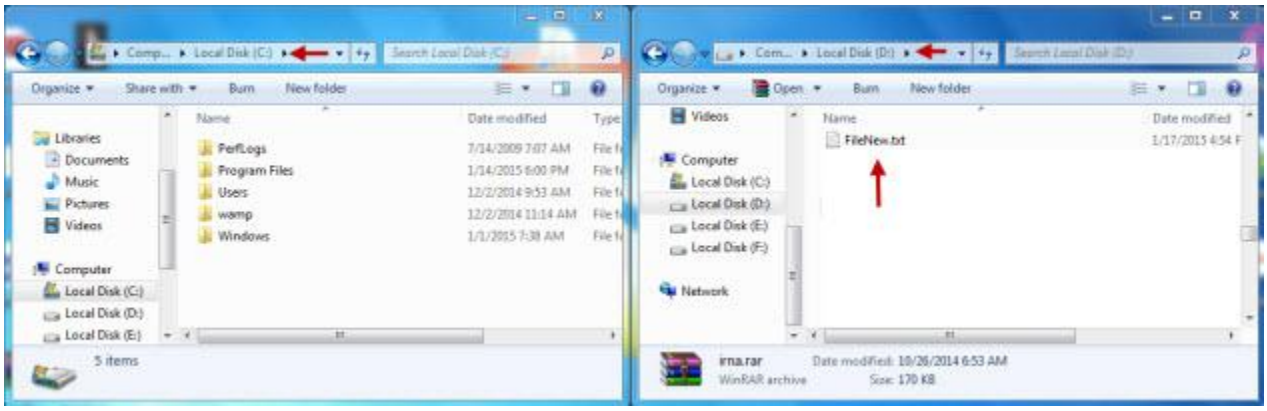
همانطور که در کد بالا مشاهده می کنید این تابع نیز یک آرگومان می گیرد که همان مسیر و نام فایل است. با ذخیره و اجرای کد فایلی که در بالا با نام File.txt در درایو C ایجاد کردیم حذف می شود. برای تغییر نام یک فایل در PHP از تابع rename استفاده می شود. این تابع دو آرگومان می گیرد. اولی مسیر و نام اصلی فایل و دومین آرگومان مسیر و نام جدید فایل. فرض کنید همین فایل File.txt را در درایو C داریم و می خواهیم آن را به FileNew.txt تغییر نام دهیم. برای این کار به صورت زیر عمل می کنیم:

```
<?php
    rename("C:\File.txt","C:\FileNew.txt");
?>
```

از همین تابع (rename) برای برش (cut) فایل هم می توان استفاده کرد. برای این کار کافیست نام درایو مقصد را به صورت زیر بنویسید:

```
<?php
    rename("C:\FileNew.txt","D:\FileNew.txt");
?>
```

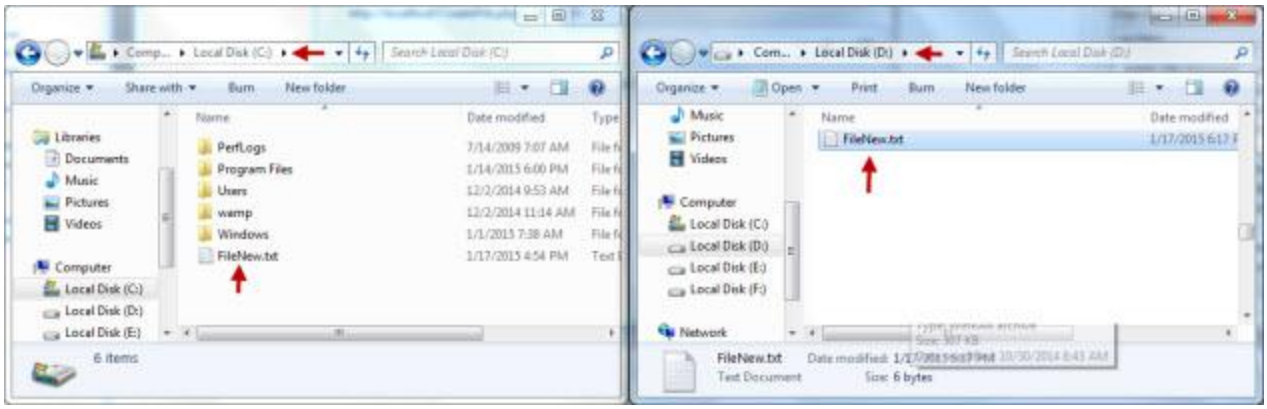
با اجرای کد بالا فایل FileNew.txt از درایو C به درایو D منتقل می شود:



برای کپی فایل هم از تابع copy استفاده می شود. فایل FileNew.txt را به درایو C برگردانید و کد بالا را به صورت زیر تغییر دهید:

```
<?php
    copy("C:\FileNew.txt","D:\FileNew.txt");
?>
```

با اجرای کد بالا یک کپی از فایل FileNew.txt در درایو D ایجاد می شود:



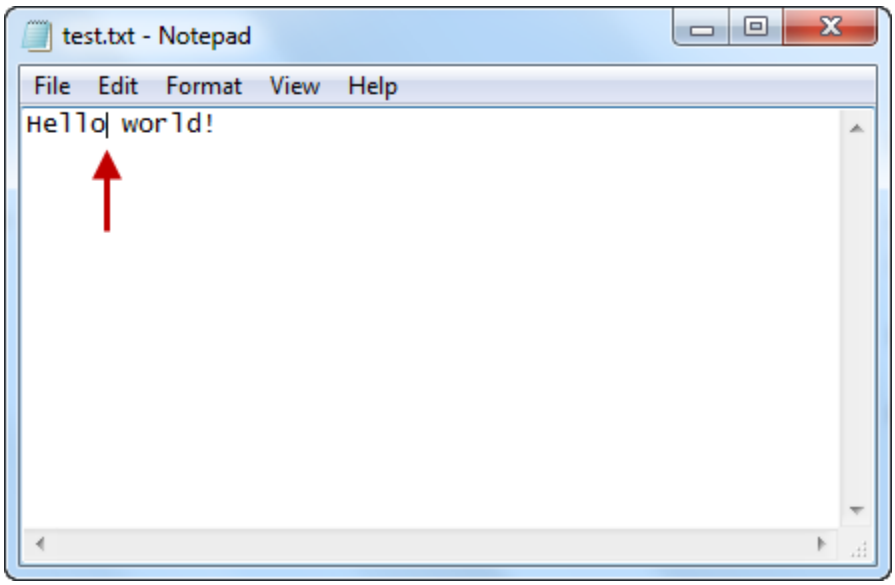
به درست آوردن موقعیت و انتقال اشاره گر به مکانی دیگر

فرض کنید که یک فایل با نام **text.txt** در درایو **C:** دارید و در آن جمله **Hello World!** نوشته شده است و می خواهیم بعد از خوانده شدن تعداد خاصی از کاراکترها موقعیت اشاره گر را به دست بیاوریم که در کجای فایل است. برای این کار از تابع **ftell()**به صورت زیر استفاده می کنیم:

```
<?php
    $file = fopen('c:\test.txt','r');
    fgets($file,6);
    echo ftell($file);
?>
```

```
5
```

همانطور که در کد بالا مشاهده می کنید با استفاده از تابع **fopen()** فایل را باز کرده،سپس با تابع **fgets()** شش کاراکتر اول را می خوانیم. از آنجاییکه این تابع یک کاراکتر کمتر از آن چیزی که برایش تعریف کرده ایم می خواند،یعنی تا پایان کلمه **Hello.** در نتیجه نشانگر بعد از حرف **o** قرار می گیرد و اگر با استفاده از تابع **ftell()** بخواهیم که بدانیم نشانگر کجاست عدد **5**را به ما نشان می دهد:



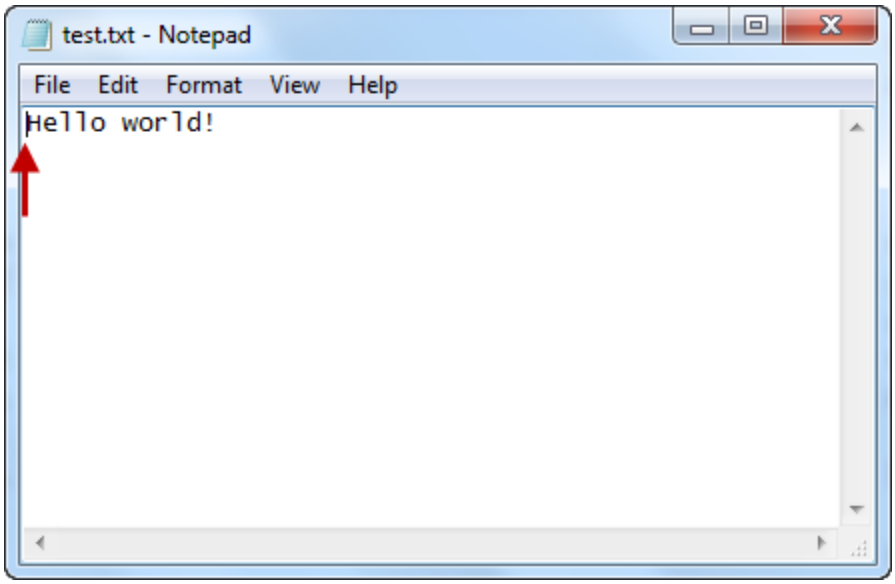
حال فرض کنید می خواهیم موقعیت کنونی اشاره گر را به ابتدای فایل بیاوریم برای این کار از تابع **rewind()** به صورت زیر استفاده می کنیم:

```
<?php
    $file = fopen('c:\test.txt','r');
    fgets($file,6);
```

```
    rewind($file);
    echo ftell($file);
?>
```

```
5
```

همانطور که در خروجی مشاهده می کنید عدد **0** نمایش داده شده است چون بعد از خوانده شدن **5** کاراکتر به وسیله تابع **fgets()** در خط بعد از آن با از استفاده از تابع **rewind()** اشاره گر را به ابتدای فایل آورده ایم و در نهایت تابع **ftell()** به ما می گوید که اشاره گر در ابتدای فایل است:



گاهی اوقات نیر ممکن است که بخواهید اشاره گر را به نقطه ای خاص، جهت خواندن از یا نوشتن در فایل انتقال دهید. برای این کار می توانید از تابع **fseek()** استفاده کنید. به مثال زیر توجه کنید:

```
<?php
    $file = fopen('c:\test.txt','r');
    fseek($file,6,SEEK_SET);
    echo fgets($file,6);
?>
```

```
world
```

به آرگومان دوم این تابع یعنی عدد **6** توجه کنید. این عدد بدین معناست که چند کاراکتر نشانگر را به جلو ببرد. یعنی هر کاری که قرار است روی فایل انجام شود از کاراکتر **6** به بعد باشد. آرگومان سوم هم سه مقدار می گیرد:

مقدار	کاربرد
SEEK_SET	مبدا حرکت نشانگر ابتدای فایل در نظر گرفته می شود

SEEK_CUR	مبدا حرکت نشانگر موقعیت کنونی نشانگر در نظر گرفته می شود
SEEK_END	مبدا حرکت نشانگر انتهای فایل در نظر گرفته می شود

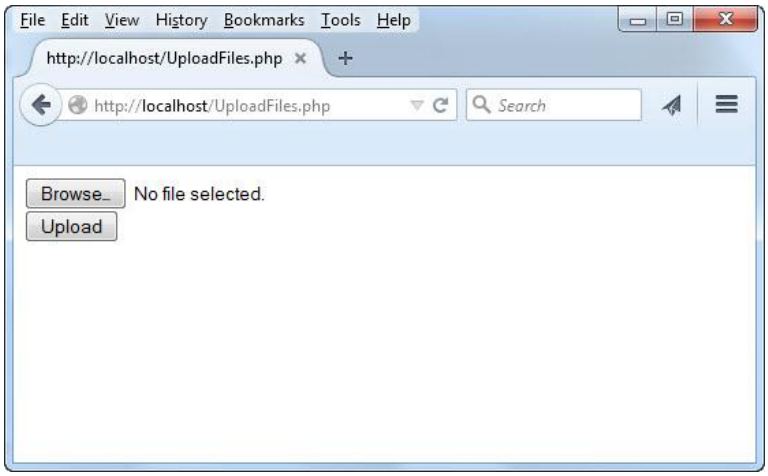
پس دلیل اینکه چرا در مثال بالا خروجی کلمه world است این است که تابع fseek() نشانگر را 6 کاراکتر به جلو می کشد و از آنجا به بعد 6 یا در اصل 5 کاراکتر توسط تابع fgets() خوانده و چاپ می شود.

آپلود فایل

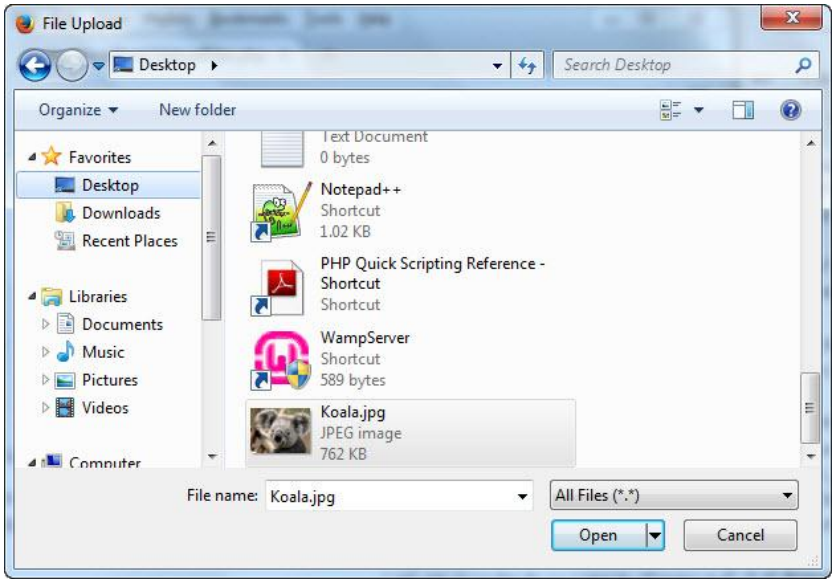
PHP این امکان را به شما می دهد که به کاربران اجازه دهید فایل های خودشان(عکس، فیلم و...) را در سرور شما آپلود کنند. البته این آپلود کردن فایل باید با اعتبار سنجی و ایجاد محدودیت همراه باشد تا کاربران نتوانند هر گونه فایلی را آپلود کنند. برای اضافه کردن این قابلیت (آپلود فایل) به سایت باید یک فرم به صورت زیر ایجاد کنید:

```
<form enctype="multipart/form-data" method="POST">

    <input type="file" name="file"/>
    <br>
    <input type="submit" name="submit" value="Upload">

</form>
```

همانطور که در کد بالا مشاهده می کنید فرم باید دارای خاصیت enctype=”multipart/form-data” و نوع دکمه هم باید file باشد،از این دکمه برای انتخاب فایل استفاده می شود. همچنین اطلاعات باید به روش POST ارسال شوند. خروجی کد بالا به صورت زیر است:



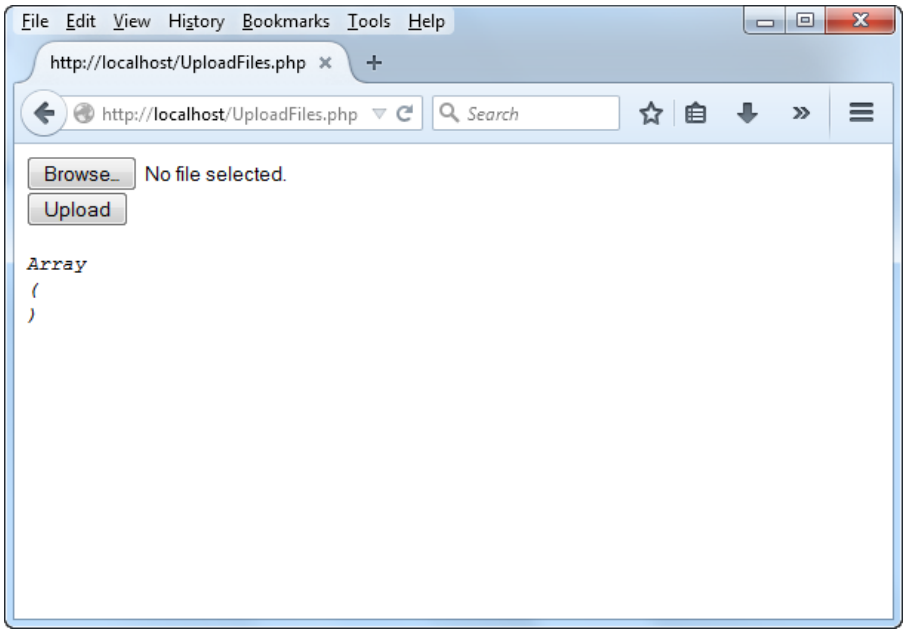
این ساده ترین شکل ایجاد یک فرم آپلود است. حال اگر بر روی دکمه Browse کلیک کنید پنجره ای به صورت زیر به نمایش در می آید:



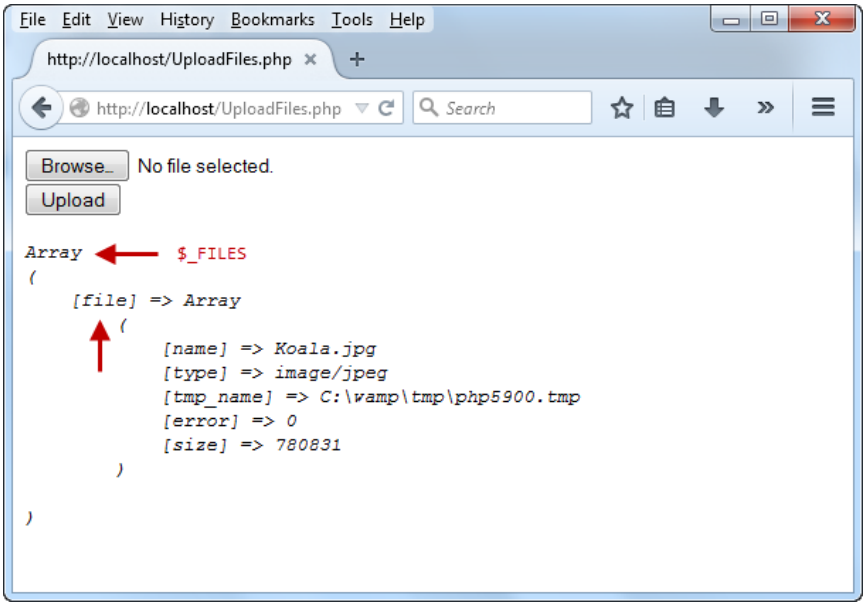
همانگونه که در شکل مشاهده می کنید هر نوع فایلی قابل انتخاب است و کاربر هیچ محدودیتی در انتخاب فایل ندارد.در آپلود فایل ها با یک آرایه سراسری به نام $_FILES سر و کار داریم. این آرایه اطلاعاتی در مورد فایل های آپلود شده در اختیار ما می گذارد. برای آشنا شدن با کارکرد این آرایه کدهای اول درس را به صورت زیر تغییر دهید:

```
<form enctype="multipart/form-data" method="POST">

    <input type="file" name="file"/>
    <br>
    <input type="submit" name="submit" value="Upload">

</form>

<?php
   echo '<pre>';
   print_r($_FILES);
   echo '</pre>';
?>
```

کد را ذخیره، اجرا و بر روی دکمه Upload کلیک کنید:



همانطور که در شکل بالا مشاهده می کنید، چون هنوز فایلی آپلود نشده، آرایه $_FILES خالی می باشد. حال یک عکس انتخاب می کنیم (مثلا عکسKoala.jpg) و سپس بر روی دکمه آپلود کلیک می کنیم:



مشاهده می کنید که آرایه $_FILES خود آرایه ای از آرایه ها است. در اینجا آرایه file که نام دکمه Browse است (name="file") درون آرایه $_FILES قراردارد. طبیعی است که برای دسترسی به مقدار عنصر name آرایه file باید به صورت زیر عمل کنیم:

```
$_FILES["file"]["name"]
```

هر فایلی که آپلود می شود مشخصاتی از آن در آرایه ای همنام با دکمه Browse آن ذخیره می شود. در جدول زیر در مورد هر کدام از این مشخصات توضیح داده شده است:

خاصیت	توضیح
name	نام فایلی که آپلود شده است
type	نوع فایلی که آپلود شده است
tmp_name	مسیر موقتی در سرور است که فایلها در آن ذخیره می شوند.
error	خطایی که در هنگام آپلود رخ می دهد.
size	سایز فایلی که آپلود شده است

همانطور که اشاره شد باید برای کاربر محدودیت هایی در آپلود فایل یجاد کنیم. فرض کنید می خواهیم به کاربر حق انتخاب فقط عکس را بدهیم. برای این کار لازم است که فرم را طوری برنامه نویسی کنیم که فقط پسوند های مجاز عکس (jpg,jpeg,PNG و ...) را قبول کند. دو تابع از پیش تعریف شده در PHP وجود دارند که این کار را برای ما انجام می دهند، explode و end ، اما چگونه؟ به کد زیر توجه کنید:

```
1  <form enctype="multipart/form-data" method="POST">
2
3      <input type="file" name="file"/>
4      <br>
5      <input type="submit" name="submit" value="Upload">
6
7  </form>
8  <?php
9      if(isset($_POST['submit']))
10     {
11         $Allowextension = array("gif", "jpeg", "jpg", "png");
12         $FileExtension = explode(".", $_FILES["file"]["name"]);
13         $extension = end($FileExtension);
14         if (in_array($extension, $Allowextension))
15         {
16             if ($_FILES["file"]["error"] == 0)
17             {
18                 echo 'File upload successfuly!';
19             }
20             else
21             {
22                 echo 'Error in uploading File!';
23             }
24         }
```

```
25          else
26          {
27              echo 'Invalid file';
28          }
29      }
30  ?>
```

ابتدا در خط 11 پسوندهایی که قرار است فرم قبول کند را درون یک آرایه می نویسیم:

```
$Allowextension = array("jpg","jpeg","PNG");
```

سپس در خط 12 پسوند فایلی را که کاربر انتخاب کرده است را به دست می آوریم. اینجاست که کاربرد تابع **explode()** مشخص می شود. این تابع یک رشته را می گیرد و آن را بر اساس کاراکتری که خودمان مشخص می کنیم تکه تکه و سپس در یک آرایه قرار می دهد. به کد زیر توجه کنید:

```
$FileExtension= explode(".", $_FILES["file"]["name"]);
```

در کد بالا ما نام فایل را که **Koala.jpg** است را بر اساس علامت نقطه (.) تکه تکه می کنیم. خروجی تابع **explode()** یک آرایه است. پس **$FileExtension**در کد بالا به صورت زیر است:

```
$FileExtension= array("Koala","jpg");
```

چون ما فقط عنصر آخر این آرایه را که همان پسوند فایل است می خواهیم پس در خط 13 با استفاده از تابع **end()** که آخرین عضو یک آرایه را بر می گرداند مقدار این عنصر را به دست می آوریم:

```
$extension = end($FileExtension);
```

تابع **in_array()** نیز چک می کند که آیا یک عنصر که مورد نظر ماست در داخل یک آرایه وجود دارد یا نه؟ در خط 14 با دستور **if** چک می کنیم که اگر پسوندی که به دست آورده ایم در داخل پسوندهای مجاز بود اجازه اجرای دستورات بعدی داده شود. در بدنه دستور **if** و در خط 16 هم چک می کنیم که اگر مقدار خطایی که هنگام آپلود فایل رخ داده است **0** باشد یعنی فایل آپلود شده باشد پیغام مناسب به کاربر نمایش داده شود. برای ایجاد محدودیت در اندازه فایل آپلودی هم می توانید خط 14 کد بالا را به صورت زیر تغییر دهید:

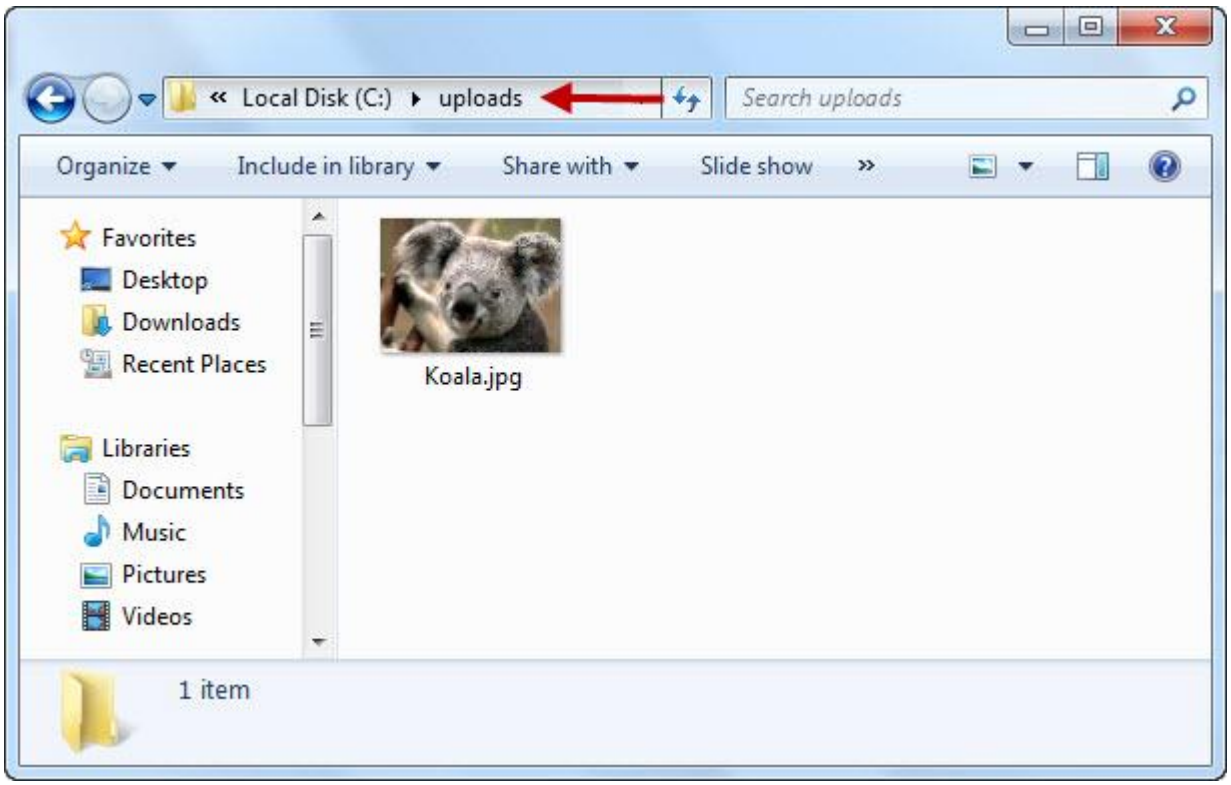
```
if (in_array($extension, $Allowextension) && ($_FILES["file"]["size"] <=
20971520))
```

در کد بالا گفته ایم که فرم اجازه آپلود عکسهایی که اندازه آنها کوچکتر یا مساوی دو مگابایت است، را به کاربر بدهد. اندازه را باید بر حسب بایت بدهیم. همانطور که گفتیم **tmp_name** مسیر موقتی است که فایل ها در آن ذخیره می شوند. تا کنون هر چه فایل با استفاده از کدهای بالا آپلود کرده ایم در این مسیر ذخیره شده اند. فرض کنید که می خواهیم فایل ها در مثلا پوشه

ای به نام **uploads** در درایو **C:** ذخیره شوند. برای این کار از متد **move_uploaded_file()** استفاده می شود. ابتدا یک پوشه به نام **uploads** در درایو **C:** ایجاد کنید و سپس کد زیر را در بالای خط **16** کد بالا بنویسید:

```
move_uploaded_file($_FILES["file"]["tmp_name"],
'C:\uploads/'.$_FILES["file"]["name"]);
```

همانطور که در کد بالا مشاهده می کنید این تابع دو آرگومان می گیرد، یکی مسیر موقتی که فایل در آن قرار دارد و دیگری مسیری که قرار است فایل به آنجا منتقل شود. با اجرای برنامه و انتخاب یک عکس و زدن دکمه **Upload** عکس به مسیر جدید منتقل می شود:



کار با پوشه ها

PHPبرای کار با پوشه ها (ایجاد، حذف و ...) نیز دارای توابع از پیش تعریف شده ای می باشد. مثلا برای ایجاد یک پوشه در PHPاز تابع mkdir() به صورت زیر استفاده می شود:

```
<?php
    mkdir("C:\Test");
?>
```

این تابع پارامتری می گیرد که هم مسیر و هم نام فایلی را که می خواهید ایجاد کنید، می باشد. مثلا در مثال بالا اگر کد را ذخیره و اجرا کنید پوشه ای با نام Test در درایو C: ایجاد می شود. برای حذف پوشه هم از تابع rmdir() به صورت زیر استفاده می شود:

```
<?php
    rmdir("C:\Test");
?>
```

این تابع نیز مسیر و نام پوشه ای که قرار است حذف شود را از شما می گیرد و آن را حذف می کن. برای به دست آوردن مسیر پوشه ای که در آن هستید هم از تابع getcwd() استفاده می شود:

```
<?php
    echo getcwd();
?>
```

خروجی کد بالا به صورت C:\wamp\www می باشد. چون من الان در داخل پوشه www کد بالا را اجرا کرده ام و این مسیر ممکن است برای شما متفاوت باشد .توابع دیگری برای کار با پوشه ها وجود دارد که لیست آنها در قسمت مرجع سایت آمده است.

پروتکل های ارسال ایمیل

پست الکترونیکی یکی از مهمترین سرویس های اینترنت است که شباهت زیادی به پست معمولی دارد. این سرویس، اتصال غیر هم زمان رابرای افراد پدید می آورد. بدین معنا که افراد هر زمان مایل باشند می توانند اقدام به ارسال و یا مطالعه ی نامه های خود نمایند، بدون این که نیاز باشد این اعمال را با زمان و برنامه ریزی دیگران منطبق کنند. هنگامی که یک نامه ی الکترونیکی ارسال می شود، انتظار این است که سرویس دهنده ی پست الکترونیکی، آن نامه را به درستی به مقصدارسال نماید. مراحل ارسال بدون توجه به سخت افزار و نرم افزار و تنها با استفاده از پروتکل های انتقال پست الکترونیکی انجام می شود. قبل از پرداختن به چگونگی ارسال ایمیل توسط PHP ابتدا لازم است با چندین اصطلاح و پروتکل آشنا شوید.

Mail Serverچیست؟

Mail Serverنرم افزاری می باشد که امکانات و سرویس های خاصی برای ارسال ایمیل از یک سرور به سایت ها و ایمیل های دیگران فراهم می کند.

SMTPچیست ؟

SMTPیا Simple Mail Transfer Protocol یکی از پروتکل های TCP/IP برای ارسال یا انتقال ساده ایمیل است که مانند یک دستیار است و ایمیل را از فرستنده دریافت و برای گیرنده ارسال می کند . پروتکل SMTP به دلیل محدودیت‌هایی در نگهداری نامه‌ها، معمولا با پروتکل‌های POP3 یا (post office protocol3) یا (internet message access protocol) IMAPاستفاده می‌شود که برای کاربران امکان ذخیره نامه‌ها را روی یک سرور یا دانلود آنها را از سرور فراهم می‌کند. در حقیقت می‌توان گفت، SMTPبرای ارسال نامه‌ها و POP3یا IMAP برای دریافت نامه‌ها به کار می‌روند. به عبارت ساده‌تر، سرور SMTP، مانند وب سرور یک رایانه است که مانند مسیریاب عمل می‌کند. هنگامی که پیام‌های پست الکترونیکی از کاربران را دریافت می‌کند آنها را به گیرندگان مورد نظر می‌فرستند SMTP .فقط به نام کاربری و دامنه نیاز دارد تا مستقیم پیغام را به سمت گیرنده مسیریابی کند و به طور پیشفرض بر روی پورت 25قرار دارد. البته مدیران سرور برای افزایش امنیت می توانند پورت آن را تغییر دهند.

POP3چیست؟

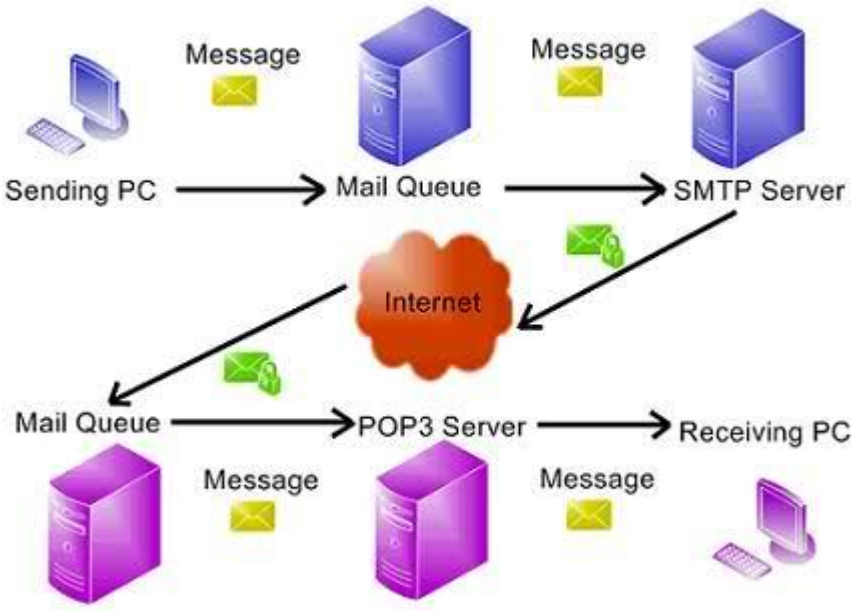
مخفف عبارت Post Office Protocol 3 است و یک پروتکل (قانون) استاندارد برای دریافت ایمیل از سرور است. به طور خلاصه کارش این است که نامه‌های شما را بدون مراجعه مستقیم به صندوق پستی با استفاده از نرم افزارهای ویژه مثل Outlookبر روی هارد ذخیره می‌کند. در حالت پیش‌فرض تمام نامه‌های موجود در پوشه Inbox از روی سرور به پوشه Inbox

محلی منتقل شده و از روی سرور حذف می‌گردند حتی اگر ویروسی باشند بنابراین از باز کردن و خواندن نامه‌هایی که گیرنده‌ی نامه را نمی‌شناسید خودداری کنید. از مزایای POP3 این است که به صورت OffLine (عدم اتصال به سرور پست الکترونیک) نیز می‌توانید نامه‌های الکترونیکی خود را که قبلاً دانلود کرده‌اید ببینید.
توجه: در استفاده از این پروتکل برای افزایش سرعت دستیابی به پیام‌ها سعی کنید حجم و تعداد نامه‌ها در پوشه‌ی Inbox بر روی سرویس‌دهنده کم باشد.

IMAPچیست؟

مخفف عبارت Internet Message Access Protocol است و همانند POP3 یک پروتکل استاندارد برای دریافت ایمیل از سرور است اما دارای مزایایی نسبت به پرتکل POP3می‌باشد. در POP3 پس از دریافت ایمیل‌ها، ایمیل‌ها از روی سرور پاک می‌شود. شما از طریق IMAP این امکان را خواهید داشت که بدون دانلود کردن ایمیل‌های خود از روی سرور ، درون ایمیلهای خود جستجو کنید ، پوشه بسازید ، نامه‌های الکترونیکی را پاک کنید و mailbox خود را برای نامه‌های الکترونیکی جدید چک کنید . این امکانات بتدریج باعث جایگزینی IMAPبه جای POP3می شود. یکی از پر استفاده‌ترین موارد استفاده از IMAP حالت اشتراکی است مثلا در شرکتی که باید چند نفر اجازه دسترسی به پست الکترونیک شرکت را داشته باشند IMAP راه حل مناسبی است.

سناریوی زیر عملیات پروتکل SMTP را به تصویر می کشد:



ارسال ایمیل در PHP

برای ارسال ایمیل در PHP از تابع mail() استفاده می شود. دستور استفاده از این تابع به صورت زیر است:

```
mail(to,subject,message,headers,parameters)
```

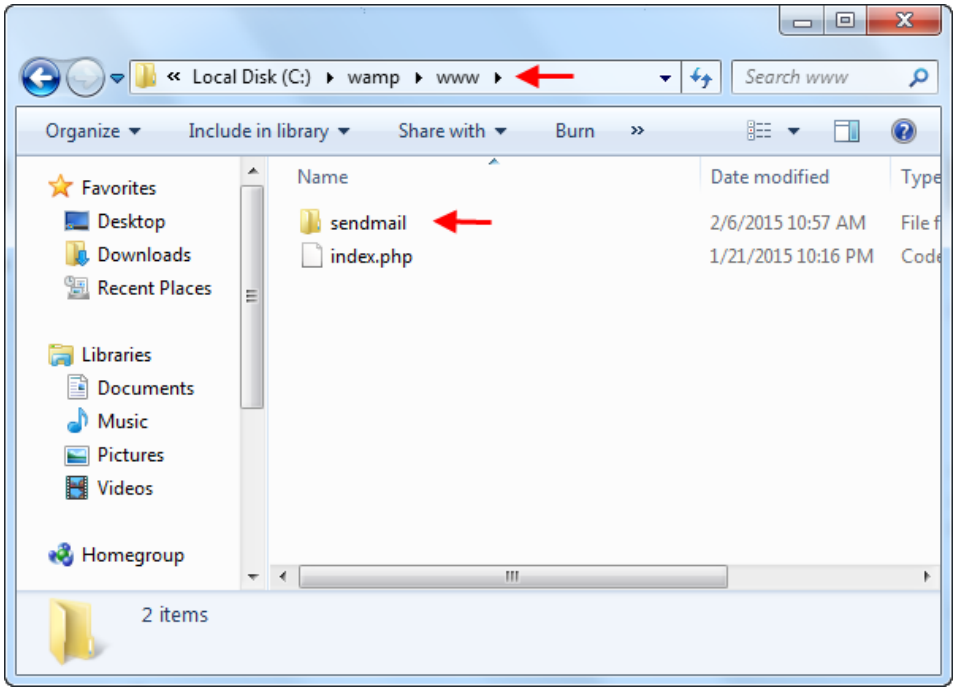
در جدول زیر در مورد پارامترهای این تابع توضیح داده شده است:

پارامتر	توضیح
to	نوشتن این پارامتر اجباری است و ایمیل گیرنده و یا گیرندگان را مشخص می کند.
subject	نوشتن این پارامتر اجباری است و از نوع متن بوده و تنها میتواند یک خط باشد. موضوع عنوان ایمیل را مشخص می کند.
message	نوشتن این پارامتر اجباری است و از نوع متن بوده و نباید بیش از ۷۰ کاراکتر باشد. متن نوشته شده داخل ایمیل را مشخص میکند و هر خط باید با کاراکتر (n/) از خط قبل و بعد جدا شود.
header	نوشتن این پارمتر اختیاری است Cc .یا Bcc را مشخص میکند. برای جدا کردن هر خط از خط قبلی باید از کارکتر (\r\n)استفاده شود.
Parametrs	نوشتن این بخش اختیاری است. برای اضافه کردن پارامتر ها به برنامه ارسال ایمیل استفاده می شود.

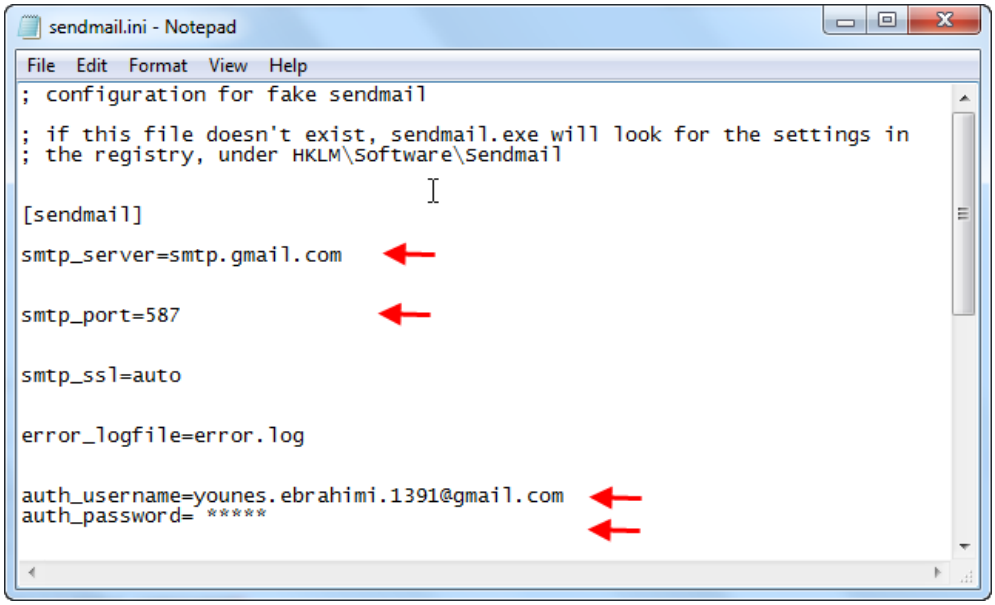
در حالت عادی و از طریق نرم افزار WAMPServer امکان ارسال ایمیل وجود ندارد و باید تغییراتی در فایل PHP.ini داده شود و همچنین از یک فایل جانبی نیز استفاده کنیم. همانطور که در درس قبل اشاره شد ایمیل از طریق یک سرور SMTPو یک پورت ارسال می شود. دو سرور SMTP معروف Google و Yahoo می باشند .برای ارسال ایمیل ابتدا فایل زیر را دانلود کنید:

دانلود فایل SendMail

سپس آن را در مسیر C:\wamp\www کپی کنید:



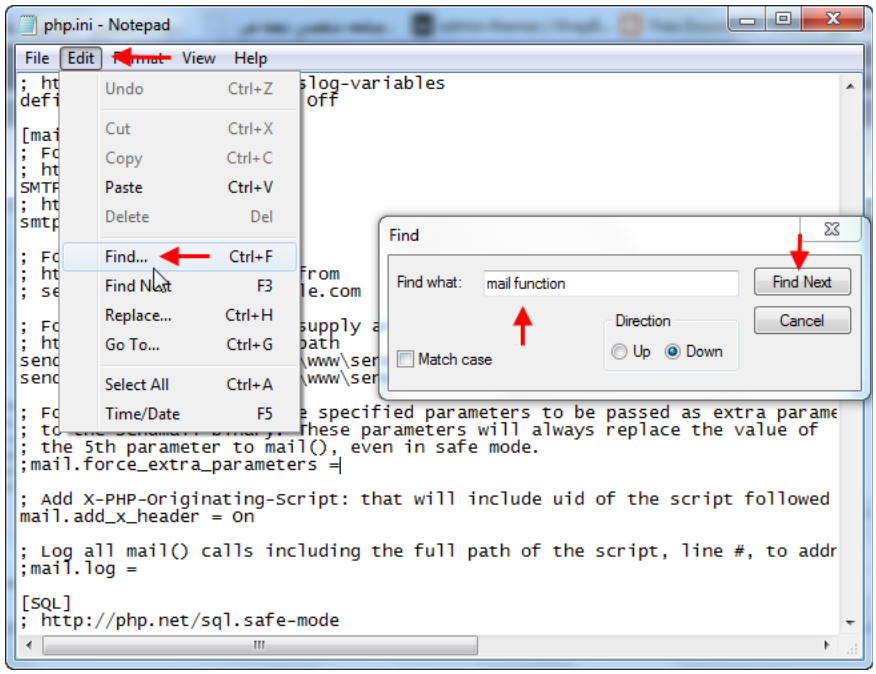
بعد به داخل پوشه رفته و تغییرات زیر را در فایل sendmail.ini اعمال کنید:



نکته ای که در شکل بالا وجود دارد این است که در قسمت auth_username و auth_passwordآدرس Email یا Gmail همراه با پسورد اصلی خودتان را بنویسید .چون این آدرس و پسورد توسط سرور SMTP اعتبارسنجی شده و به شما اجازه ارسال ایمیل را می دهد .فایل sendmail.ini دارای قسمت هایی است که در جدول زیر درباره کاربرد آنها توضیح داده شده است:

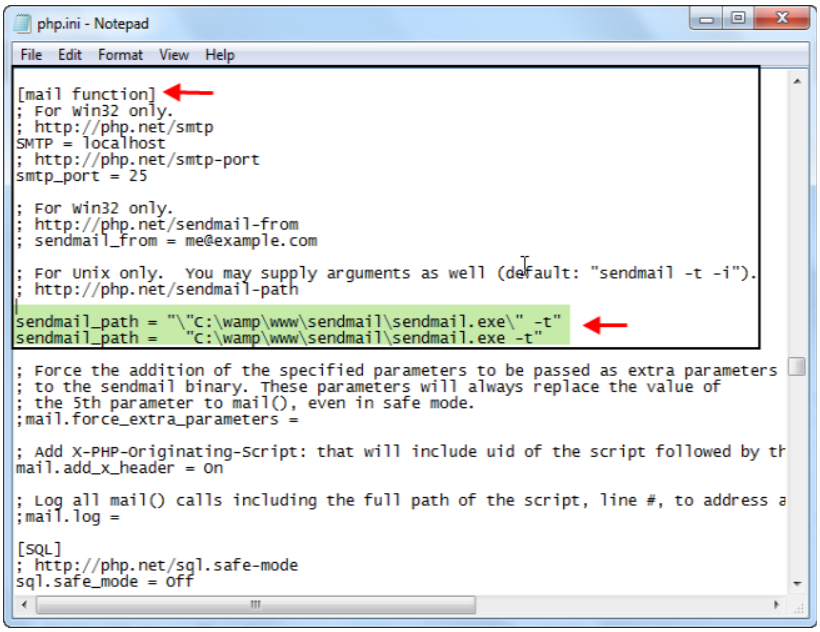
پارامتر	توضیح
smtp_server	آدرس سرور SMTP
smtp_port	شماره پورت سرور. به صورت پیشفرض پورت 25 می باشد و برای سرورهایی که از پروتکل های امن مانند ssl استفاده می کنند مانند جی میل 465 می باشد و برای 587 می باشد
auth_username	نام کاربری
auth_password	رمز عبور
force_sender	آدرس پست الکترونیک فرستنده
error_logfile	ذخیره خطاها در یک فایل متنی
smtp_ssl	نوع پروتکل ارسال اطلاعات که می تواند مقدار auto,none,ssl وtls داشته باشد برای سرورهایی که به صورت امن اطلاعات ارسال میکنند مانند GMail

حال نوبت به ویرایش فایل PHP.ini می رسد. به مسیر C:\wamp\bin\apache\apache2.4.4\bin رفته و این فایل را پیدا کنید و بعد از باز کردن آن با Notepad ، مانند شکل زیر به قسمت [mail function] بروید:

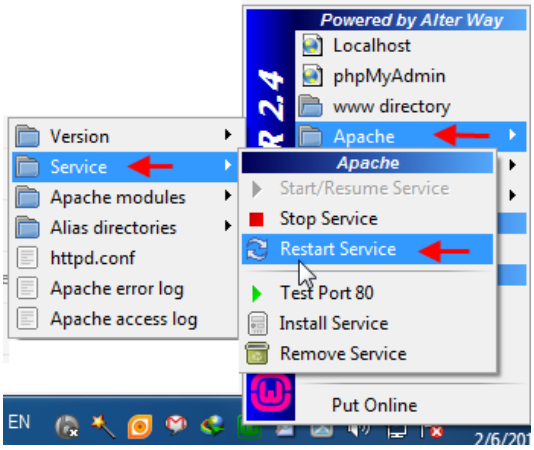


بعد از پیدا کردن قسمت [mail function] دو خط زیر را که همان مسیر فایلی است که دانلود کرده اید (SendMail) به آن اضافه کنید:

```
sendmail_path = "\"C:\wamp\www\sendmail\sendmail.exe\" -t"
sendmail_path =   "C:\wamp\www\sendmail\sendmail.exe -t"
```



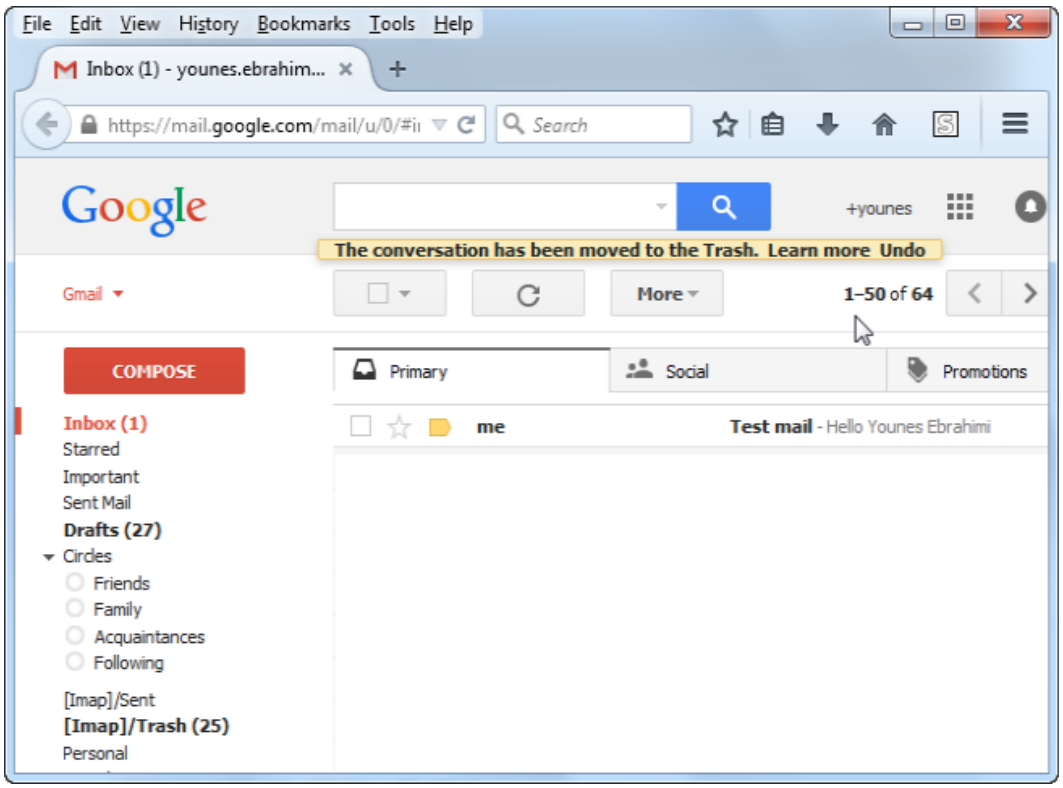
بعد از اعمال تغییرات فوق سرویس Apache را ریستارت کنید:

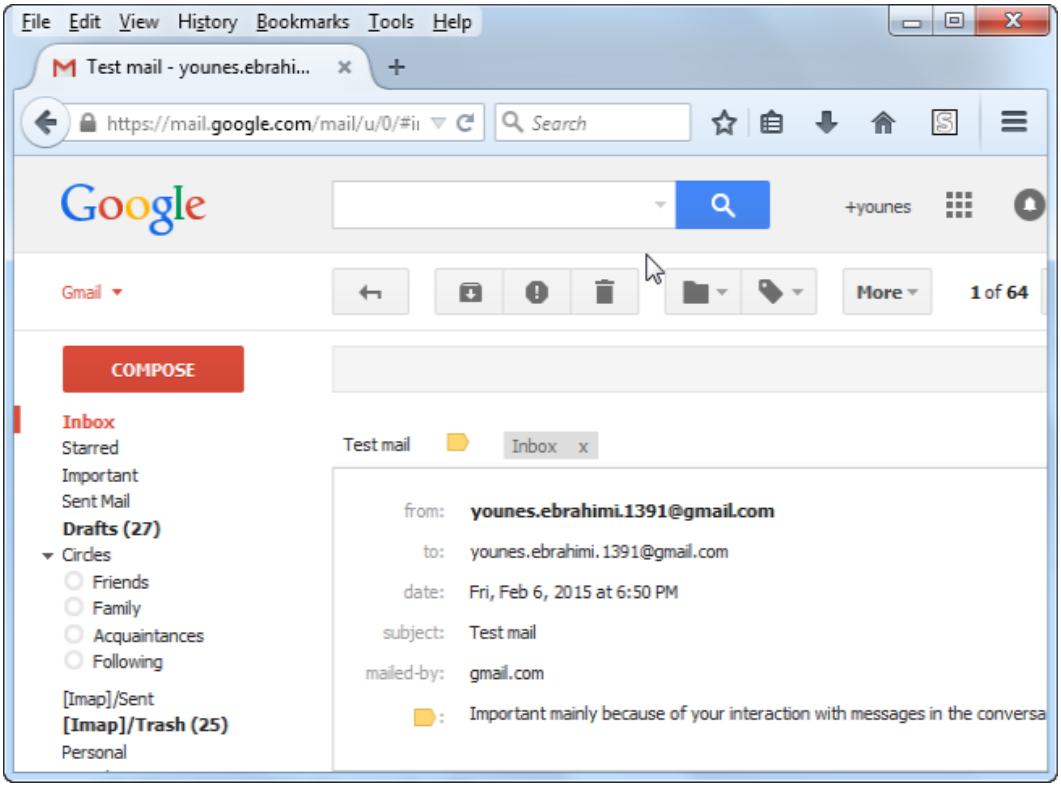


سپس یک فایل با پسوند **php** با نام **SendMail.php** در پوشه **www** ایجاد کرده و کدهای زیر را به آن اضافه کنید:

```
<?php
    $to = "younes.ebrahimi.1391@gmail.com";
    $subject = "Test mail";
    $message = "Hello Younes Ebrahimi";
    $from = "younes_ebrahimi_1391@gmail.com";
    $headers = "From:" . $from;
    mail($to,$subject,$message,$headers);
    echo "Mail Sent.";
?>
```

بعد از اجرای کد بالا پیغامی مبنی بر ارسال ایمیل به شما نمایش داده می شود. در کدهای بالا من از آدرس **Gmail** خودم برای **Gmail**خودم پیامی ارسال کرده ام و شما می توانید آن را آدرس های خودتان تغییر دهید. حال اگر به قسمت **inbox** جیمیل مراجعه کنید مشاهده می کنید که پیام ارسال شده است:

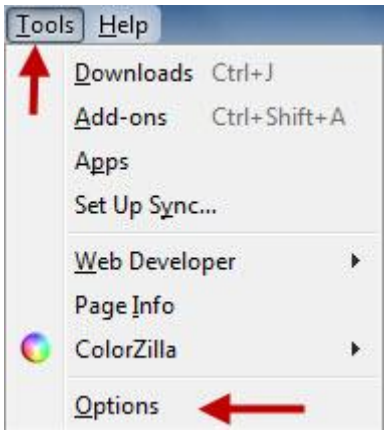


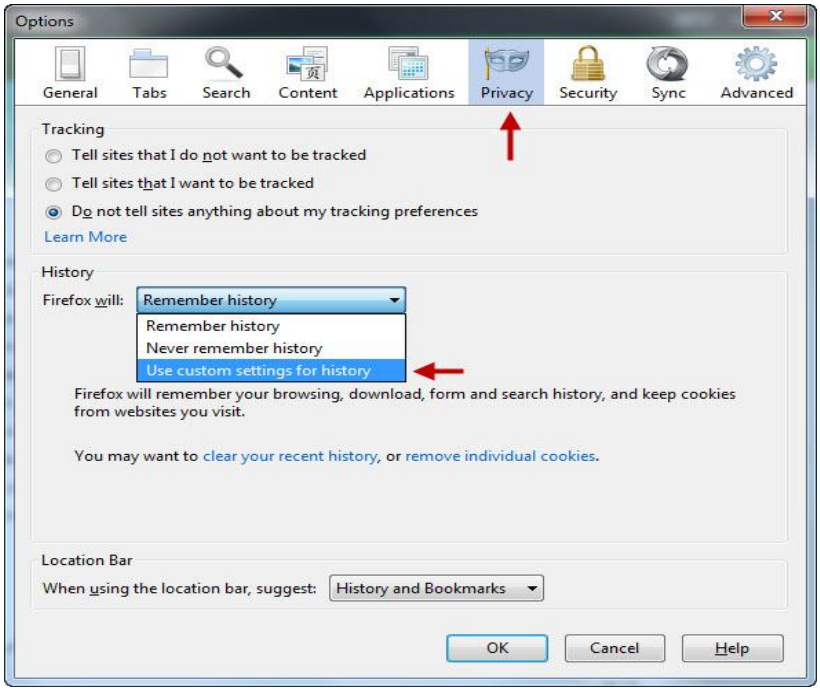

File Edit View History Bookmarks Tools Help
Test mail - younes.ebrahi...
https://mail.google.com/mail/u/0/#i
Search
Google
+younes
Gmail
More
1 of 64
COMPOSE
Inbox
Starred
Important
Sent Mail
Drafts (27)
Circles
Friends
Family
Acquaintances
Following
[Imap]/Sent
[Imap]/Trash (25)
Personal
Test mail
Inbox x
from: younes.ebrahimi.1391@gmail.com
to: younes.ebrahimi.1391@gmail.com
date: Fri, Feb 6, 2015 at 6:50 PM
subject: Test mail
mailed-by: gmail.com
Important mainly because of your interaction with messages in the conversa

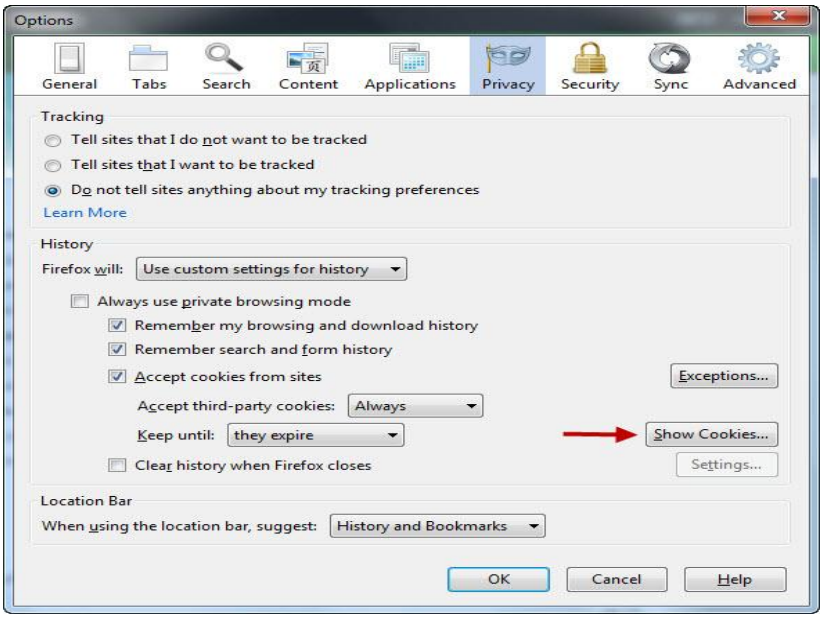
کوکی (cookie) چیست؟

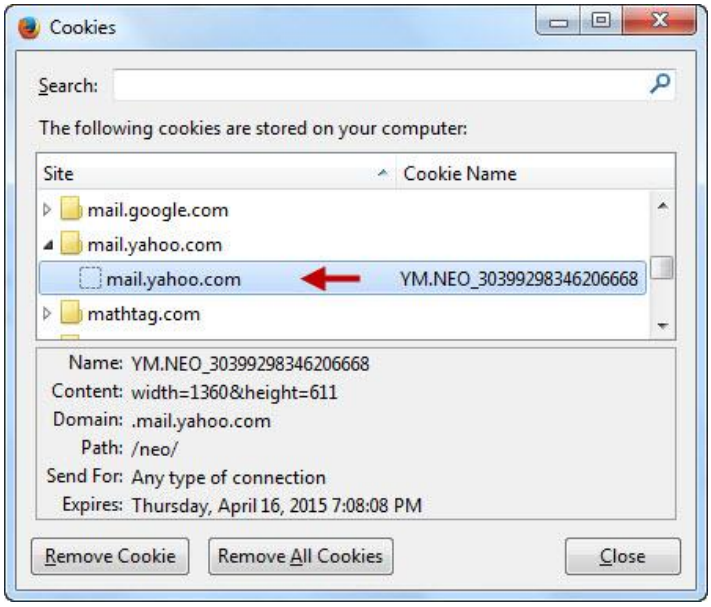
کوکی (cookie) یک قطعه اطلاعات (یک فایل متنی) است که سرور بر روی کامپیوتر کاربر ذخیره کرده و در مراجعات کاربر به همان سایت ، از آن اطلاعات برای شناسایی وی استفاده می کند . این اطلاعات معمولا راجع به شناسه کاربری ، رمز عبور ، تنظیمات یک کاربر بر روی سایت و ... می باشد . زمانی که کاربر به وسیله کامپیوتر خود همان سایت را باز می کند ، مرورگر اطلاعات کوکی ها را به سرور ارسال می کند . از کوکی برای بازیابی اطلاعات کاربری و یا سایر تنظیماتی که کاربر در یک سایت ایجاد کرده است ، در مراجعات بعدی به همان سایت استفاده می شود. برای مثال فرض کنید که در یک سایت فروم عضو شده و یک رمز عبور و شناسه کاربری را برای خود تعیین کرده اید . هنگامی که برای اولین بار اطلاعات کاربری خود را در مرورگر وارد می کنید ، این اطلاعات در یک کوکی بر روی مرورگر کامپیوتر شما ذخیره می شود . هنگامی که در مراجعه بعدی به همان سایت می روید ، متوجه می شوید که مرورگر اطلاعات کاربری شما را وارد نموده و شما log in شده اید ، بدون اینکه خودتان کاری انجام داده باشید .اینکه کوکی ها در کجا ذخیره می شوند، بسته به تنظیمات مرورگرهای مختلف، متفاوت است .مثلا در زیر مسیر ذخیره شدن کوکی ها را در مرورگر firefox مشاهده می کنید

:




Options
General
Tabs
Search
Content
Applications
Privacy
Security
Sync
Advanced
Tracking
Tell sites that I do not want to be tracked
Tell sites that I want to be tracked
Do not tell sites anything about my tracking preferences
Learn More
History
Firefox will:
Remember history
Remember history
Never remember history
Use custom settings for history
Firefox will remember your browsing, download, form and search history, and keep cookies from websites you visit.
You may want to clear your recent history, or remove individual cookies.
Location Bar
When using the location bar, suggest:
History and Bookmarks
OK
Cancel
Help


Options
General
Tabs
Search
Content
Applications
Privacy
Security
Sync
Advanced
Tracking
Tell sites that I do not want to be tracked
Tell sites that I want to be tracked
Do not tell sites anything about my tracking preferences
Learn More
History
Firefox will:
Use custom settings for history
Always use private browsing mode
Remember my browsing and download history
Remember search and form history
Accept cookies from sites
Exceptions...
Accept third-party cookies:
Always
Keep until:
they expire
Show Cookies...
Clear history when Firefox closes
Settings...
Location Bar
When using the location bar, suggest:
History and Bookmarks
OK
Cancel
Help



مثلا در آخرین شکل، همانطور که با فلش نمایش داده شده است، مشخصات ایمیل ما در یک فایل Cookie ذخیره شده است که در نتیجه در مراجعات بعدی به ایمیل نام کاربری و کلمه عبور از ما پرسیده نمی شود. در واقع مرورگر از کوکی برای ایجاد ، ذخیره و ارسال مجدد این اطلاعات به سرور استفاده کرده است. مرورگر ها معمولا برای ایجاد و ذخیره کوکی ها از کاربر سوال کرده و یا گزینه ای را جهت ایجاد آن قرار می دهند . پس از تایید کاربر اقدام به ایجاد و ذخیره کوکی ها می نمایند . توسط زبان PHPشما به راحتی می توانید کوکی های خود را ایجاد کرده و مجددا آنها را دریافت و ارسال نمایید . در این بخش قصد داریم تا شما را با نحوه ایجاد و خواندن کوکی ها در زبان PHP آشنا نماییم.

ایجاد کوکی (cookie) در PHP

از تابع setcookie() برای ایجاد و ذخیره یک کوکی در زبان PHP استفاده می شود . برای ایجاد یک کوکی تعیین چند چیز مهم است:

- **نام**
- **مقدار**
- **مدت زمان اعتبار**

نکته : توجه داشته باشید که کوکی ها پس از یک مدت زمان تعیین شده ، اعتبار خود را از دست می دهند و باید دوباره فراخوانی شوند . در این حالت می گوییم ، کوکی expire شده است . برای مثال فرض کنید در یک سایت login کرده اید . سپس کامپیوتر و مرورگر خود را برای مدتی رها می کنید (مرورگر و صفحه جاری را نمی بندید) . پس از مراجعه دوباره متوجه می شوید ، که مرورگر شما را logout کرده است و دوباره باید وارد شوید . دلیل این مسئله پایان یافتن مدت زمان اعتبار یک کوکی است . شکل کلی تعریف و ایجاد یک کوکی به وسیله تابع setcookie() در PHP به صورت زیر است:

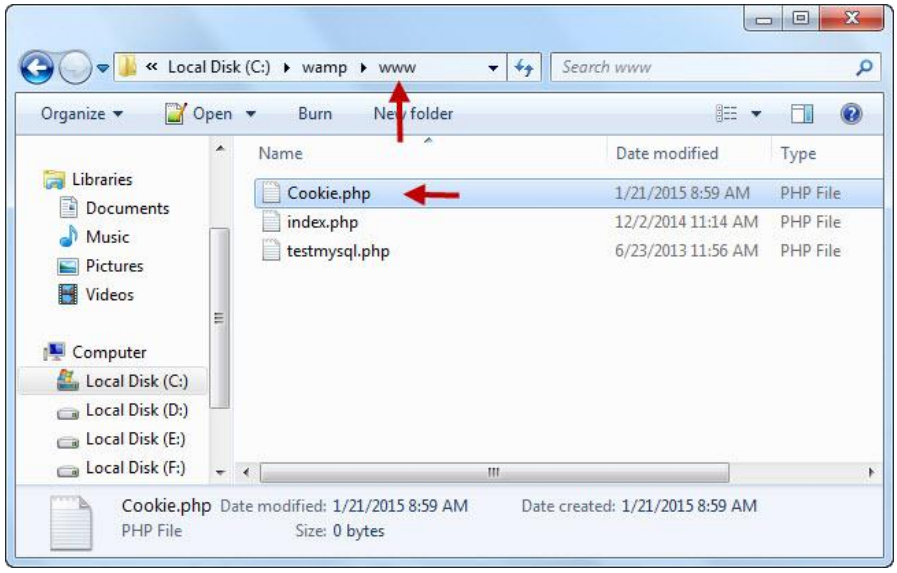
```
setcookie(name , value , expire , path , domain);
```

در شکل زیر قسمت های مختلف یک کوکی نمایش داده شده است:

پارامتر	توضیح
name	پارامتر name تعیین کننده نام کوکی است . از این نام برای بازیابی و فراخوانی کوکی در سطح برنامه استفاده می شود.
Value / Content	پارامتر value مقدار کوکی را تعیین می کند .این مقدار بر روی کامپیوتر ذخیره می شود.
Domain / Host	پارامتر domain ، دامنه ای که کوکی بر روی آن قابل دسترس است را تعیین می کند. برای مثال اگر مقدار آن 'www.w3-farsi.ir' تنظیم شود ، فقط در این دامنه قابل دسترسی است. تعیین این پارامتر اختیاری است.
path	این پارامتر تعیین کننده مسیری بر روی سرور سایت است که کوکی در آن ، قابل دریافت و ذخیره است. برای مثال اگر مقدار آن برابر با '/' تعیین شود ، در تمام سایت قابل دسترسی است . اما اگر روی مقدار '/en/' تنظیم شود ، فقط در دایرکتوری en در سایت قابل دسترسی است. تعیین این پارامتر اختیاری است.
expire	پارامتر expire تعیین کننده مدت زمان اعتبار کوکی ، بر حسب ثانیه است . پس از اتمام این مدت زمان ، کوکی از بین خواهد رفت. برای تعیین مدت زمان اعتبار کوکی ، معمولا از تابع time ( ) استفاده می شود.

برای آشنایی بیشتر با عملکرد کوکی یک مثال می زنیم. ابتدا تمام کوکی های مرورگر (در اینجا firefox) را پاک می کنیم:

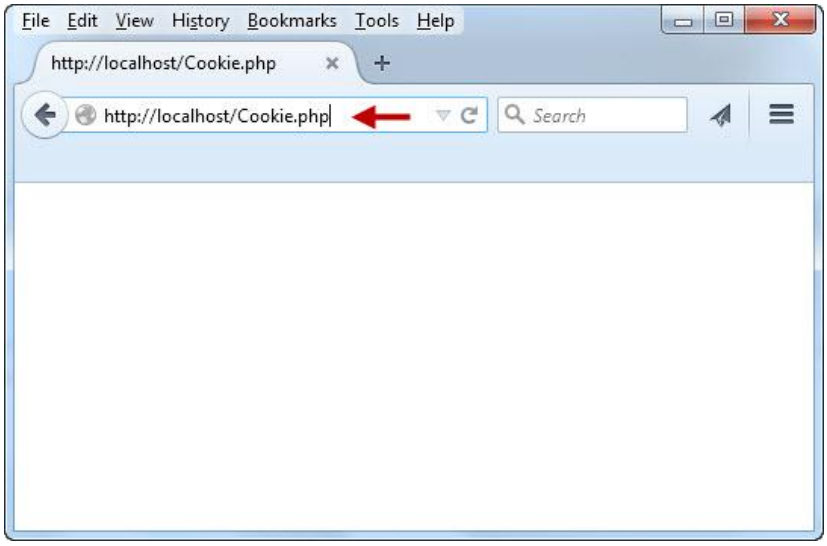
سپس یک فایل با نام Cookie.php در مسیر C:\wamp\www ایجاد کنید:



حال با استفاده از یک ویرایشگر متن، فایل بالا را باز کرده و کد زیر را در داخل آن بنویسید وذخیره کنید:

```php
<?php
    setcookie('Username','Ali');
?>
```

کد بالا را در مرورگر اجرا کنید:



با رفتن به مسیر فایلهای کوکی مشاهده می کنید که یک کوکی به صورت زیر ایجاد شده است:

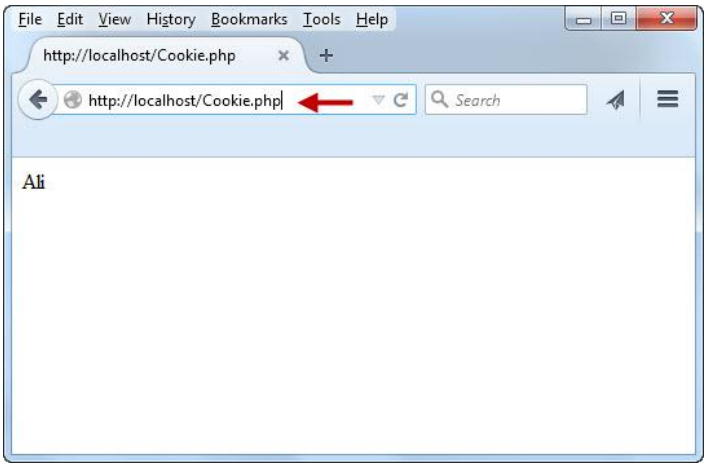
با توجه به شکل بالا ،یک کوکی با نام **Username** و مقدار **Ali** ایجاد شده است ولی از آنجاییکه ما برای این کوکی طول عمر تعریف نکرده ایم**(Expire)** ، این کوکی بعد از بسته شده مرورگر پاک می شود. برای جلوگیری از این کار یک زمان انقضا به صورت زیر برای آن تعریف می کنیم:

```
<?php
    setcookie('Username','Ali',time()+60);
?>
```

در کد بالا با استفاده از تابع **()time** که تاریخ فعلی سرور را مشخص می کند و عدد **60** (ثانیه)، مشخص کرده ایم که اطلاعات کاربر از لحظه ورود تا **1** دقیقه بعد در کوکی ذخیره و بعد از آن حذف شود. حال فرض کنید که کاربر مرورگر را می بندد و قبل از پایان اعتبار کوکی که در این مثال یک دقیقه است دوباره مرورگر را باز کرده و به سایت سر می زند. این بار سرور محتویات فایل کوکی را خوانده و کاربر و تنظیمان او را تشخیص می دهد و دیگر اعتبار سنجی نمی کند. پس از ایجاد و مقداردهی کوکی، مرورگر باید بتواند از کوکی و وحتوبات آن استفاده کند. آرایه سراسری **$_COOKIE** برای به دست آوردن مقدار کوکی به کار می رود. کد بالا را به صورت زیر تغییر دهید و آنرا در مرورگر اجرا کنید:

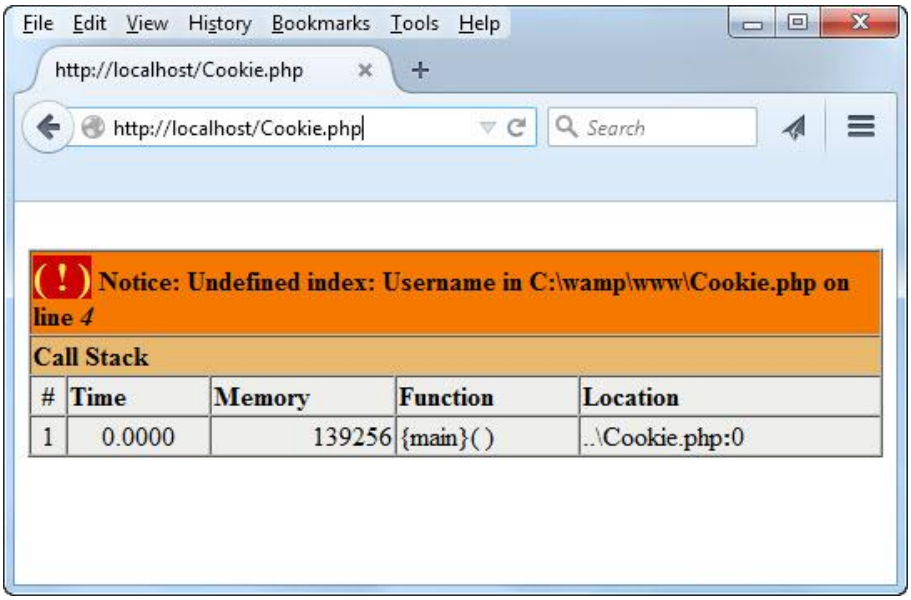
```
<?php
    setcookie('Username','Ali',time()+60);

    echo $_COOKIE["Username"];
```

```
?>
```



همانطور که در شکل بالا مشاهده می کنید برای به دست آوردن مقدار یک کوکی کافیست نام آن را (در این مثال Username) به آرایه سراسری $_COOKIE بدهیم. در نتیجه مقدار آن که در این مثال Ali است در مروگر چاپ می شود .البته مقدار Ali تا زمانی قابل شناسایی توسط سرور است که کوکی منقضی نشده باشد که در این مثال تا قبل از یک دقیقه. حال اگر بعد از گذشت یک دقیقه صفحه را Refresh کنید، چون کوکی پاک می شود در نتیجه با پیغام خطای زیر مواجه می شوید

:



Session (سشن) چیست

Session (سشن یا جلسه) به ما اجازه می دهد که در سمت سرور مقادیری را ذخیره کرده و از آنها در صفحات مختلف استفاده کنیم. این کار توسط یک آرایه فوق سراسری به نام $_SESSION انجام می شود. نمونه بارز استفاده از Session ها را می توانیم در هنگام وارد شدن به یک سایت (لاگین شدن) مشاهده کرد. به طوریکه وقتی وارد یک سایت می شوید و صفحات مختلف آن را مرور می کنید نام کاربری شما در کل صفحات در دسترس و قابل مشاهده است .فرایند ذخیره سازی داده ها و استفاده از آنها در Session به صورت است:

1. **کاربر از طریق مرورگر یک درخواست می فرستد.**
2. **با فرض اینکه این درخواست ارسال نام کاربری و کلمه عبور برای ورود به سایت باشد، یک فایل تصادفی و منحصر به فرد با پسوند sess_ در سرور و در پوشه tmp ایجاد می شود.**
3. **با ایجاد فایل در پوشه tmp ،سرور یک کوکی در داخل مرورگر ایجاد و یک رشته 32 رقمی و یکتا را در داخل آن قرار می دهد. این رشته 32 رقمی را SID یا session ID می نامند و نامی که برای آن در کوکی اختصاص داده می شود، PHPSESSID (به صورت پیشفرض) می باشد : $_COOKIE['PHPSESSID'];**
4. **برای خواندن اطلاعات از فایلی که در سرور ایجاد شده است (فایل با پسوند sess_) مرورگر با هر بار درخواست، این IDرا به سرور ارسال کرده و سرور هم فایل مخصوص آن را از داخل پوشه tmp پیدا کرده و مقادیر داخل آن را می خواند.**

تنظیمات پیشفرض Session در فایل php.ini به صورت زیر می باشد:

کاربرد و توضیح	مقدار پیشفرض	گزینه
filesبه معنای استفاده از فایل ها برای ذخیره سشن می باشد.	files	session.save_handler
مسیر ذخیره فایل های سشن را نشان می دهد.	"c:/wamp/tmp"	session.save_path
مقدار پیشفرض 1 به معنای این است که از کوکی ها برای ذخیره SID استفاده شود.	1	session.use_cookies
استفاده از این گزینه به معنای این است که فقط از کوکی ها برای اگر مقدار پیشفرض آن را به 0 تغییر دهید، SIDاز طریق کوکی ارسال نمی شود.	1	session.use_only_cookies
اگر مقدار پیشفرض session.use_only_cookies را به صفر و مقدار پیشفرض این گزینه را به 1 تغییر دهید، SIDاز طریق لینک ها یا فرم ها به صورت خودکار ارسال می شود و دیگر نیازی به استفاده از کوکی نیست.	0	session.use_trans_sid
نام پیشفرضی که به کوکی سشن اختصاص داده می شود.	PHPSESSID	session.name
اگر مقدار پیشفرض آن را به 1 تغییر دهید با شروع اسکریپت، سشن هم شروع به کار می کند یعنی تابع session_start فراخوانی می شود.	0	session.auto_start
زمان اعتبار کوکی مربوط به سشن را مشخص می کند.	0	session.cookie_lifetime

اجازه دهید که نحوه ایجاد و استفاده از سشن ها را توضیح دهیم.

ایجاد Session

ایجاد سشن در **PHP** بسیار ساده است و با استفاده از متد **session_start()** انجام می شود. وقتی که این متد فراخوانی می شود، یک فایل **session** در سرور و کوکی معادل آن که دارای رشته **32** رقمی **SID** است در مرورگر کاربر ایجاد می شود. اگر این کوکی از قبل وجود داشته باشد و توسط مرورگر ارسال شده باشد متد **session_start()** از همان استفاده می کند و کوکی جدید ایجاد نمی کند .برای فراخوانی این متد حتما باید آن را در اولین خط اسکریپت خود بنویسید:

```
<?php
   session_start();
?>
```

همانشوط که اشاره شد برای اعتبار سنجی کاربر و ارسال **SID** به سرور، سشن هم از کوکی استفاده می کند. حال اگر کاربر کوکی های مرورگر را غیر فعال کند، **SESSION** این کار را چگونه انجام می دهد؟ اگر یک لینک به صفحه مورد نظر داشته باشید سشن مقدار **PHPSESSID** را به صورت زیر اضافه می کند:

```
<a href="page.php?PHPSESSID=b8306b025a76a250f0428fc0efd20a11">Other Page</a>
```

اگر از طریق همین لینک مقادیر دیگری هم ارسال شود، **PHPSESSID**به صورت زیر اعمال می شود:

```
<a href="page.php?id=1&PHPSESSID=b8306b025a76a250f0428fc0efd20a11">Other
Page</a>
```

و اگر اطلاعات قرار است از طریق فرم ارسال شوند، **PHPSESSID**از طریق یک تگ **input** با خاصیت **hidden** ارسال می شود:

```
<form action="" method="">
    <input type="hidden" name="PHPSESSID"
value="b8306b025a76a250f0428fc0efd20a11" />
</form>
```

خواندن و نوشتن داده های Session

کار با داده های سشن بسیار راحت است. برای اینکار داده ها را به صورت کلید/مقدار در آرایه فوق سراسری $_SESSION ذخیره می کنید. در سشن بر خلاف کوکی که از تابع setCookie() برای ذخیره داده ها استفاده می کردید، چنین تابعی وجود ندارد و شما مستقیما می توانید یک نام دلخواه به عنوان کلید برای داده های که می خواهید ذخیره کنید، انتخاب نمایید. مثلا در زیر نام یک شخص را در یک Session ذخیره کرده ایم:

```
$_SESSION["firstName"] = "John";
```

در کد بالا، کلید firstName و مقدار هم John می باشد. برای خواندن این سشن هم مانند آرایه ها کافیست که از دستور echoبرای چاپ این مقدار استفاده کنیم:

```
echo ($_SESSION["firstName"]);
```

با اجرای دستور بالا مقدار John چاپ می شود.

پاک کردنSession

سشن برخلاف کوکی دارای طول عمر نیست. یعنی با بسته شدن مرورگر اطلاعات سشن هم پاک می شوند. گاهی اوقات لازم است که شما فورا سشن را پاک کنید .مثلا وقتی که کاربر از طریق سایت شما یک خرید انجام داده است و بدون اینکه مرورگر را ببندد می خواهد دوباره یک خرید دیگر انجام دهد و شما می خواهید اطلاعات خرید قبلی را پاک کنید. برای پاک کردن سشن از تابع زیر استفاده می شود:

```
<?php
   session_destroy();
?>
```

کد بالا تمام اطلاعات موجود در Session را حذف می کند، ولی اگر بخواهید اطلاعات یک سشن خاص، مثلا firstName را پاک کنید می توانید از تابع unset() استفاده نمایید:

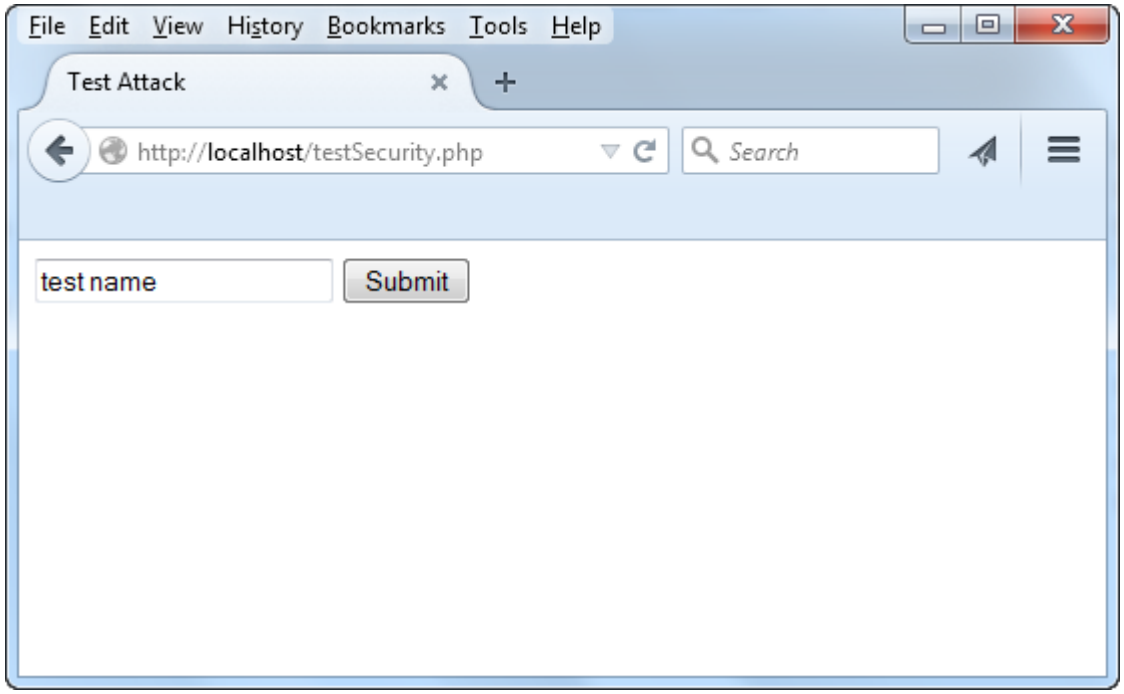
```
unset($_SESSION["firstName"]);
```

امنیت در اجزای فرم های HTML

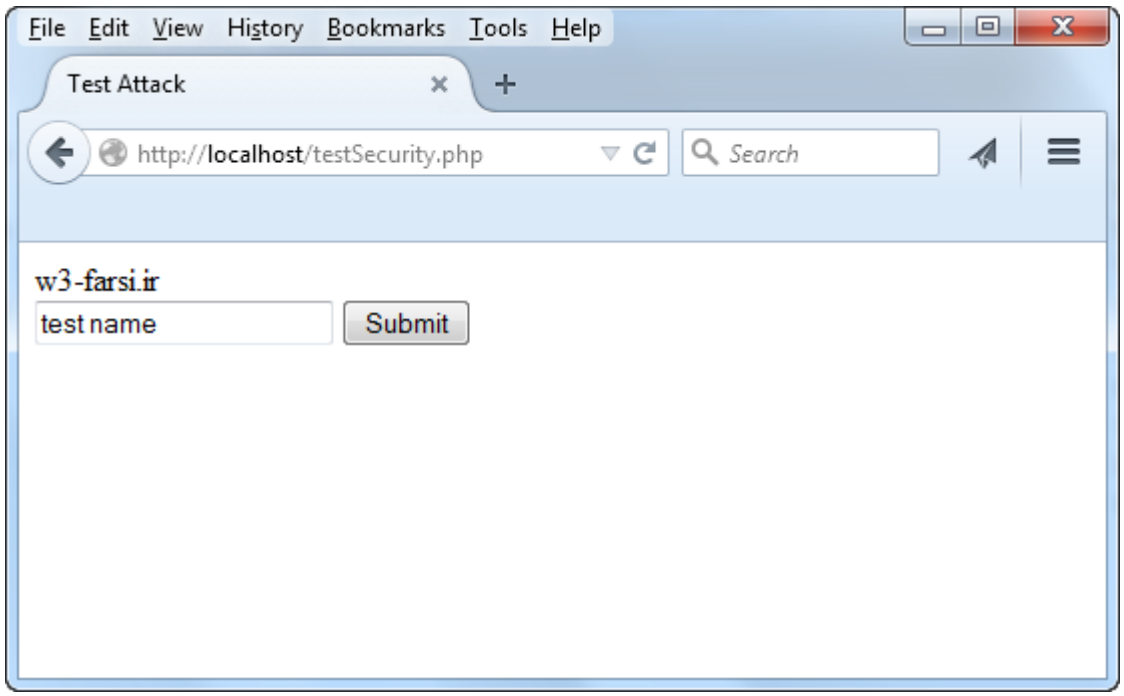
اگر در داخل فرمتان عناصری مانند جعبه متن و ... دارید باید امنیت اطلاعات وارد شده توسط کاربران را چک کنید.چون از طریق مثلا جعبه متن هکرها می توانند دستوراتی بنویسند که در سایت شما اختلالاتی به وجود آورند. به عنوان یک مثال به کد زیر توجه کنید:

```
<html>
<head>
<title>Test Attack</title>

    <?PHP
    if (isset($_POST['Submit']))
    {
          $first_name = $_POST['first_name'];
          echo $first_name;
     }
    ?>

</head>
<BODY>

    <Form Method = "Post" action ="testSecurity.php">
    <input type = "text" name = "first_name" value ="test name">
    <input type="submit" name="Submit" value="Submit">
    </Form>

</BODY>
</html>
```

کد بالا را با نام testSecurity.php در پوشه www ذخیره و اجرا کنید:



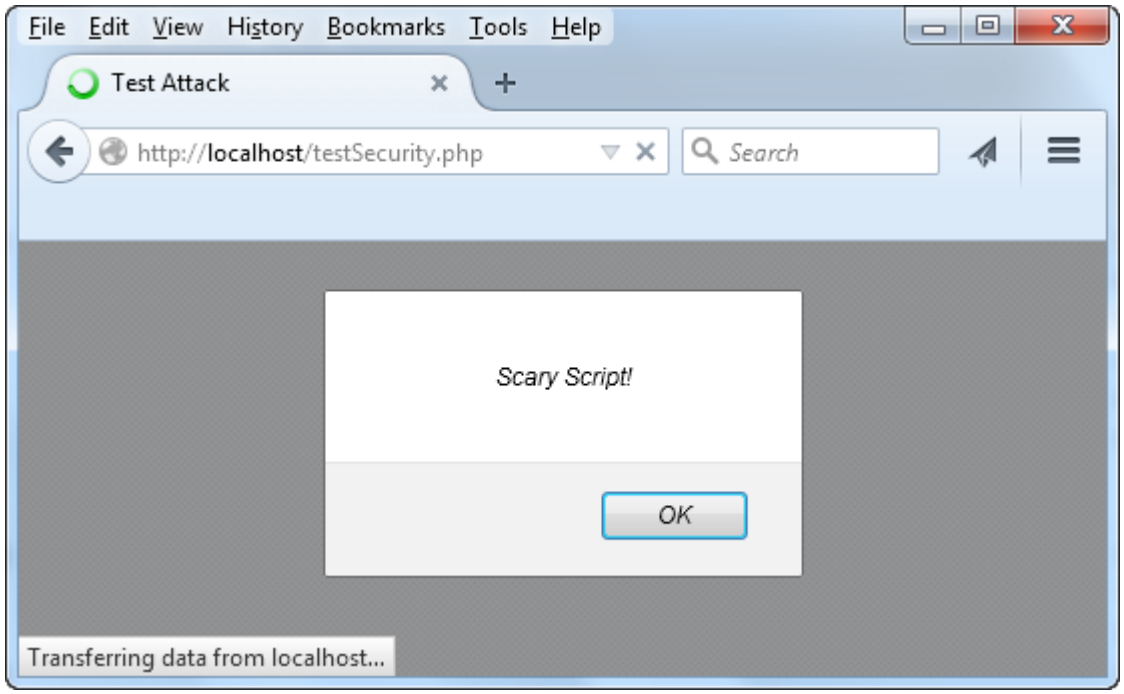
از کد بالا ما این انتظار را داریم که وقتی کاربر در درون جعبه متن، متنی نوشت در بالای جعبه متن نمایش داده شود:



حال فرض کنید که کاربر کد زیر را در داخل جعبه متن بنویسد:

```
<SCRIPT>alert("Scary Script!")</SCRIPT>
```

اگر این کد را در درون جعبه متن بنویسید و بر روی دکمه ارسال کلیک کنید اوضاع طور دیگر خواهد بود:



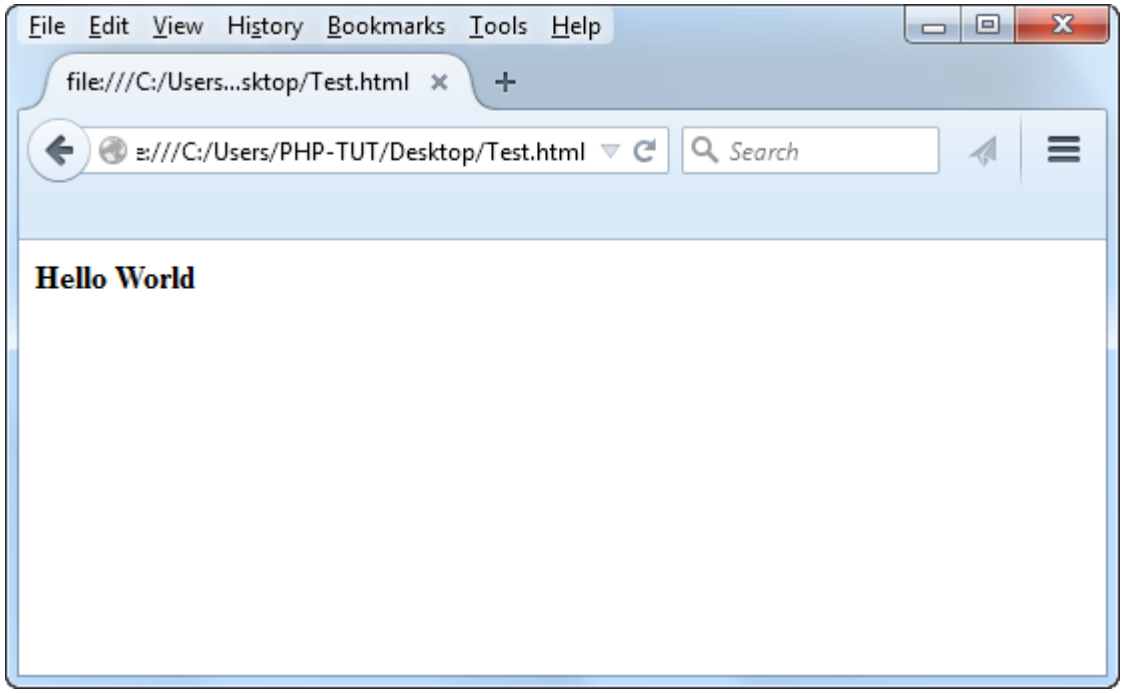
ممکن است این سوال برایتان پیش بیاید که این کد ممکن است چه تاثیری بر روی سایت بگذارد و یا چه اختلالی به وجود آورد؟ جواب این است که ممکن است که این کد هیچ خطری برای سایت شما نداشته باشد چون فقط یک مثال ساده بود ولی برنامه نویسان و در اصل هکرهای حرفه ای می توانند به راحتی از طریق همین جعبه های متن به قسمت های مهم سایت شما لطمه بزنند. در کل شما نباید به کابران اجازه دهید از عناصر HTML و سایر اسکریپت ها در جعبه های متن استفاده کنند. در درس بعد در مورد چک ورودی های کاربران بیشتر توضیح می دهیم.

تابع htmlspecialchars()

همانطور که در درس قبل دیدید، اگر در جعبه های متن از عناصر HTML و یا اسکریپت های دیگر مانند کدهای جاوا اسکریپت استفاده کنیم، برای HTML قابل درک هستند و آنها را پردازش می کند و اگر کدها مخرب باشند ممکن است به سایت آسیب برسانند. پس در نتیجه باید این کدها را قبل از پردازش غیر قابل فهم کنیم. برای این کار از یکی از توابع از پیش تعریف شده PHPبه نام htmlspecialchars() استفاده می کنیم. یکی از راه های تجریه عناصر HTML ، تجزیه علائم >و < به اجزای تشکیل دهنده آنهاست. مثلا علامت >در اصل > می باشد و علامت <به صورت <می باشد. تابع htmlspecialchars() هم همین کار را می کند. یعنی علامت بزرگتر و کوچکترها را تجزیه می کند که در این صورت اگر از طریق جعبه متن وارد شوند برای HTML غیر قابل پردازش می شوند.برای درک بهتر کد زیر را در نظر بگیرید:

```
<B> Hello World </B>
```

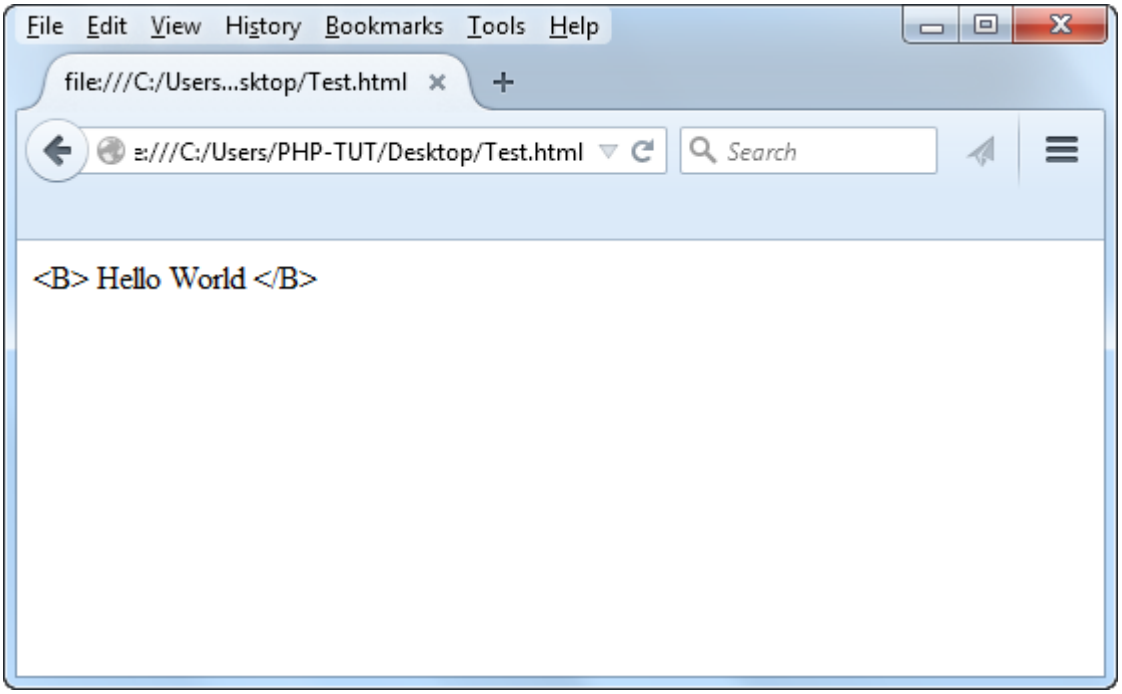
این کد نوشته را به صورت ضخیم نمایش می دهد:



ولی اگر همین کد را به صورت اجزای تشکیل دهنده در آوریم HTML فکر می کند که شما شکل علائم بزرگتر و کوچکتر را می خواهید:

```
&lt;B&gt; Hello World &lt;/B&gt;
```

در نتیجه خروجی کد بالا به صورت زیر خواهد بود:

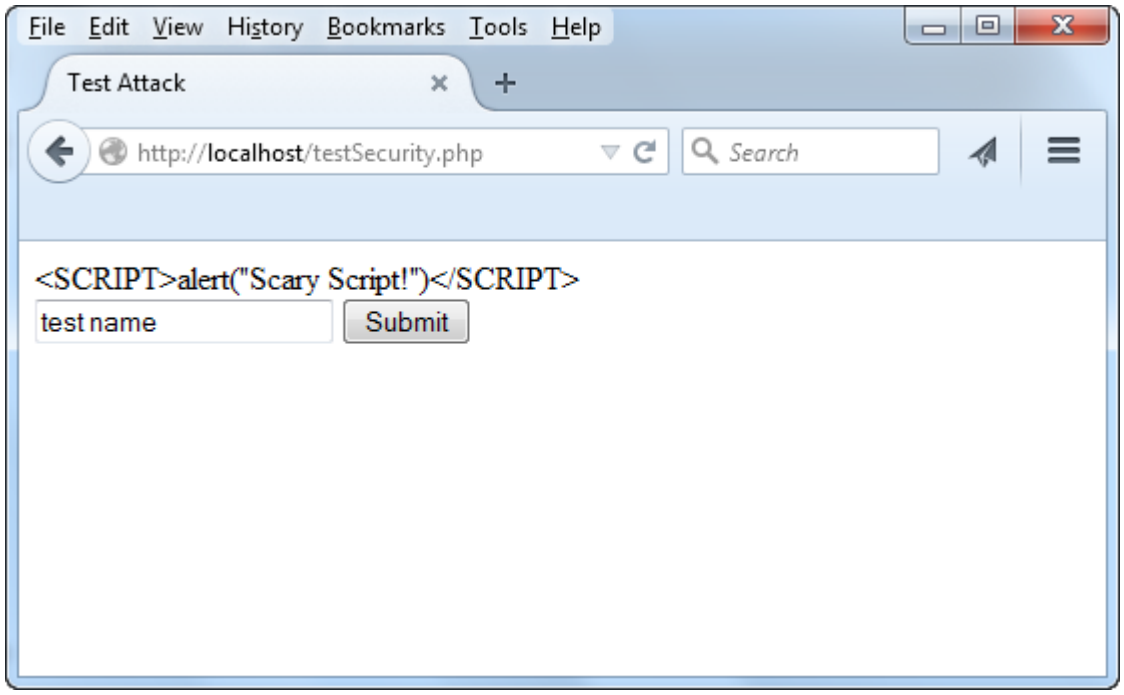


می توانیم مثال درس قبل را به صورت زیر بنویسیم:

```
<html>
<head>
<title>Test Attack</title>

    <?PHP
        if (isset($_POST['Submit']))
        {
              $first_name = htmlspecialchars($_POST['first_name']);
              echo $first_name;
        }
    ?>

</head>
<BODY>

    <Form Method = "Post" action ="testSecurity.php">
    <input type = "text" name = "first_name" value ="test name">
    <input type="submit" name="Submit" value="Submit">
    </Form>

</BODY>
</html>
```

همانطور که در خط 8 کد بالا مشاهده می کنید، ابتدا داده های دریافتی توسط آرایه $_POST را به وسیله این تابع تجزیه کرده و سپس در متغیر قرار می دهیم.حال اگر کد زیر را در جعبه متن وارد کنیم، دیگر مشکلی به وجود نمی آید:

```
<SCRIPT>alert("Scary Script!")</SCRIPT>
```



از شکل بالا به این نتیجه می رسیم که اگر علائم < و > در جعبه متن وارد شوند پردازش می شوند و به صورت کد با آنها رفتار می شود ولی اگر ابتدا تجزیه شوند فقط شکل آنها چاپ می شود و پردازش نمی شوند.

htmlentities()

این تابع شبیه به تابع **htmlspecialchars()** است با این تفاوت که کاراکترهای غیر انگلیسی مانند فرانسوی و آلمانی و ... را هم چک می کند. پس خط 8 کد بالا را می توانید به صورت زیر هم بنویسید:

```
$first_name = htmlentities($_POST['first_name']);
```

تابع strip_tags()

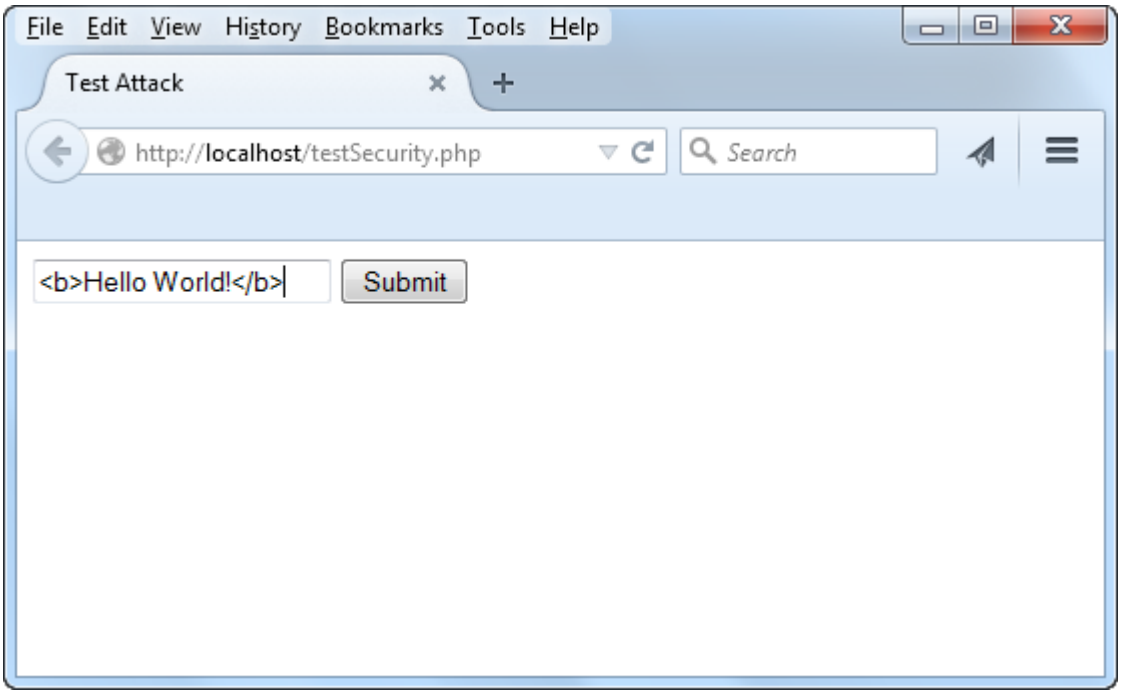
سومین گزینه تامین امنیت فرم های HTML استفاده از تابع strip_tags() است. کار این تابع حذف عناصر HTML از ورودی های کاربر است .دستور استفاده از این تابع به صورت زیر است:

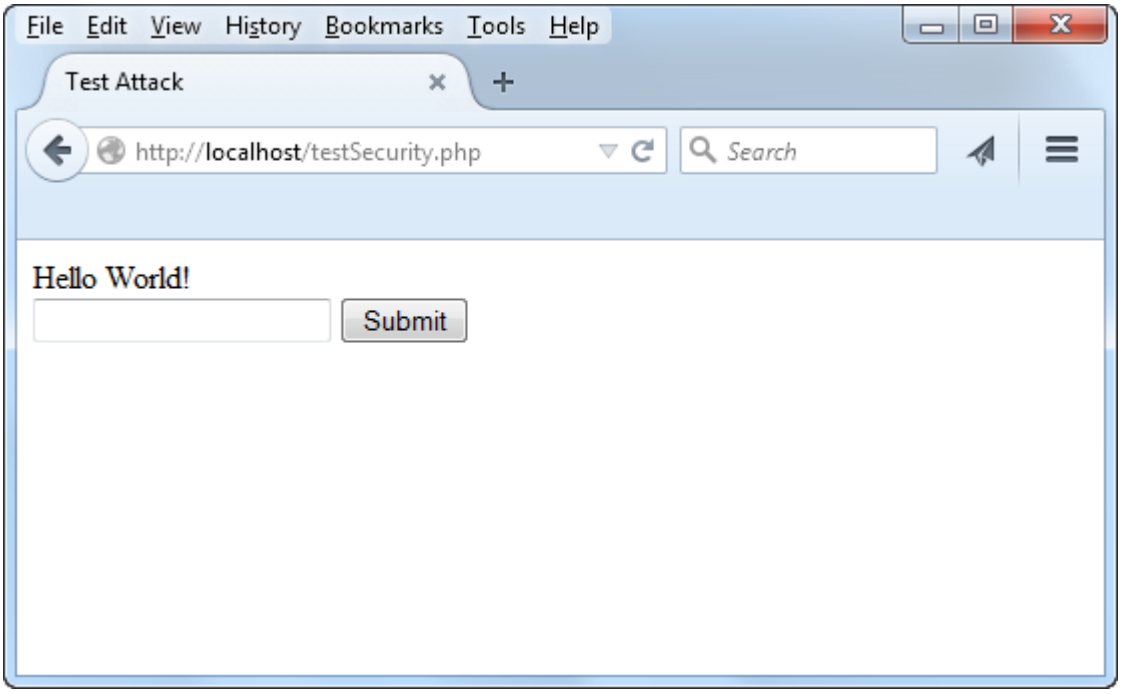
```
strip_tags( $string, html_tags_to_ignore )
```

آرگومان اول این تابع یک رشته است که می خواهیم عناصر HTML از آن حذف شوند و دومین آرگومان هم عنصر HTML ی است که نمی خواهیم پاک کنیم، یعنی از نظر خود ما مشکل ساز نیستند.اگر به این تابع فقط رشته را بدهیم تمام عناصر HTML را پاک می کند. به مثالی در مورد کارکرد این تابع توجه کنید:

```
<html>
<head>
<title>Test Attack</title>

    <?PHP
        if (isset($_POST['Submit']))
        {
              $first_name = strip_tags($_POST['first_name']);
              echo $first_name;
        }
    ?>

</head>
<BODY>

    <Form Method = "Post" action ="testSecurity.php">
    <input type = "text" name = "first_name" value ="test name">
    <input type="submit" name="Submit" value="Submit">
    </Form>

</BODY>
</html>
```

فرض کنید که ما جمله Hello World! را وارد جعبه متن و بر روی دکمه ارسال کلیک می کنیم:

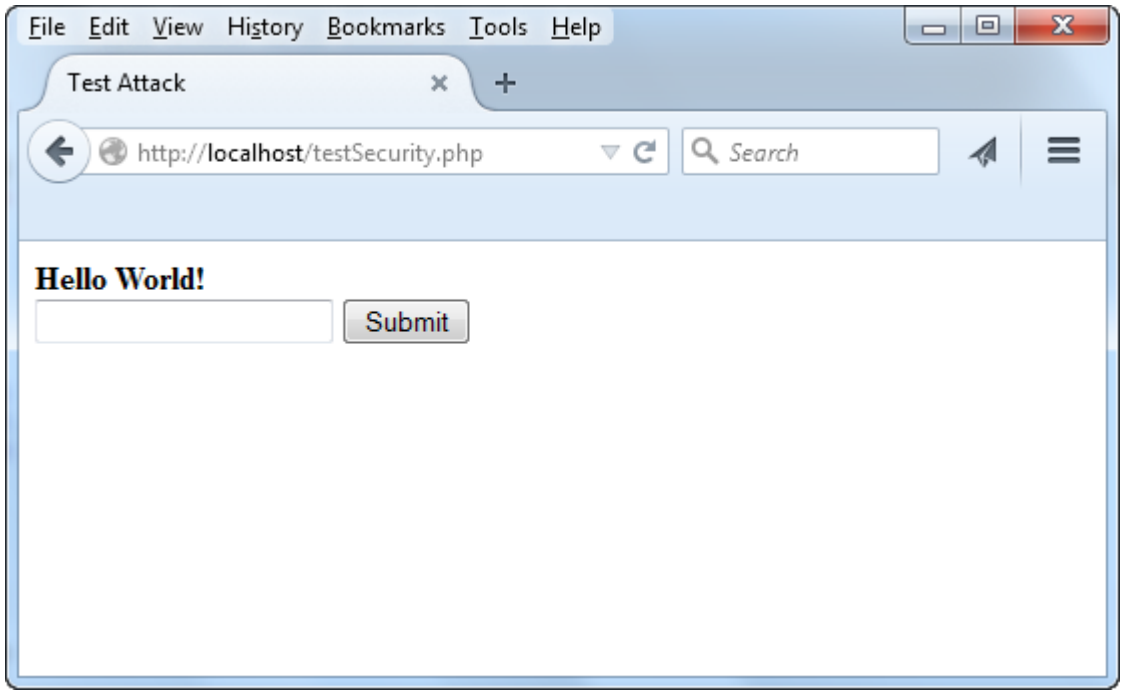




همانطور که در شکل بالا مشاهده می کنید با کلیک بر روی دکمه ارسال تگ های <b> از رشته پاک می شوند. حال می توانیم خط 8 کد بالا را به صورت زیر بنویسیم و از تابع بخواهیم که تگ <b> را حذف نکند:

```
$first_name = strip_tags($_POST['first_name'],"<b>");
```

با اجرای دوباره برنامه و زدن دکمه ارسال متن به صورت ضخیم در می آید چون کار تگ <b>ضخیم کردن متن است:



زبان نشانه گذاری توسعه پذیر (XML)

زبان نشانه گذاری توسعه پذیر (XML) به شما اجازه می دهد که داده ها را در یک متن و قالب ساخت یافته ذخیره کنید. این زبان به طور گسترده به عنوان یک دیتابیس جایگزین و برای ذخیره اطلاعات مربوط به پیکربندی نرم افزارها به کار می رود. XML از لحاظ دستوری شبیه به HTML بوده و اگر با HTML آشنایی داشته باشید یادگیری این زبان برایتان راحت تر است. در زیر یک سند XML را مشاهده می کنید :

```
<Persons>
    <Person>
        <Name>John Smith</Name>
        <Age>30</Age>
        <Gender>Male</Gender>
    </Person>

    <Person>
        <Name>Mike Folly</Name>
        <Age>25</Age>
        <Gender>Male</Gender>
    </Person>

    <Person>
        <Name>Lisa Carter</Name>
        <Age>22</Age>
        <Gender>Female</Gender>
    </Person>
</Persons>
```

سند XML ترکیبی از عناصر XML می باشد. یک عنصر XML شامل یک تگ آغازی ، یک تگ پایانی و داده ای است که در بین این دو تگ قرار می گیرد.

```
<open>data</close>
```

می توان بر اساس داده ای که یک عنصر XML در خود نگهداری می کند یک نامبرای عنصر انتخاب کرد. به این نکته توجه کنید که عناصر به حروف بزرگ و کوچک حساسند، بنابراین دو کلمه person و Person با هم متفاوتند. XML فضاهای خالی را نادیده می گیرد، بنابراین به جای نوشتن یک فایل در یک خط می توانید آن را در چند خط بنویسید تا خوانایی آن بالاتر رود. بین عناصر XML ممکن است رابطه پدر- فرزندی وجود داشته باشد

```
<parent>
    <child1>data</child1>
    <child2>
    <grandchild1>data</grandchild1>
    </child2>
```

```
</parent>
```

سند XML بالا دارای اطلاعاتی برای سه شخص می باشد. هر سند XML باید دارای یک عنصر ریشه (root) باشد. در مثال اول این درس، عنصر Persons ، عنصر ریشه (پدر) و دیگر عناصر داخل آن در حکم فرزندان آن می باشند. جزییات هر شخص در داخل عنصر Person قرار دارند. عناصر فرزند عنصر Person عبارتند از Name، Age و Gender. صفات XML، روشی دیگر برای اضافه کردن داده به یک عنصر می باشند.

```
<Personname="John Smith">some data</Person>
```

عنصر بالا یک خاصیت به نام name دارد که مقدار آن John Smith می باشد. مقادیر باید در داخل کوتیشن (' ') یا دابل کوتیشن (" ") قرار بگیرند. در زیر روش اضافه کردن صفات نشان داده شده است.

```
<elementatt1="value1" att2="value2" ... attN="valueN">data</element>
```

همانطور که مشاهده می کنید، می توان به یک عنصر چندین صفت اضافه کرد.

```
<Personname="John Smith" age="30" gender="Male">some data</Person>
```

اجازه دهید که به عناصر مثال ابتدای درس صفاتی اضافه کنیم.

```
<Persons>
    <Person name="John Smith">
        <Age>30</Age>
        <Gender>Male</Gender>
    </Person>
    <Perso nname="Mike Folly">
        <Age>25</Age>
        <Gender>Male</Gender>
    </Person>
    <Person name="Lisa Carter">
        <Age>22</Age>
        <Gender>Female</Gender>
    </Person>
</Persons>
```

عنصر Name هر شخص (person) را حذف و صفت معادل آن (name) را برای هر عنصر می نویسیم. اسناد XML می توانند دارای یک تعریف XML باشند. تعریف XML شامل اطلاعاتی درباره سند XML مانند نسخه (همیشه نسخه 1.0 پیشنهاد می شود) و نوع رمزگذاری (encode) متن آن می باشد.

```
<?xmlversion="1.0" encoding="UTF-8" ?>
```

این تعریف در بالاترین بخش سند و درست قبل از عنصر اصلی نوشته می شود. برای فایل XML می توان توضیحات نیز نوشت. نحوه نوشتن توضیحات در XML به صورت زیر است.

```
<!-- This is an XML comment -->
```

می توان با استفاده از یک ویرایشگر متن ساده فایل های XML تولید کرد.

Document Object Model یا DOM چیست

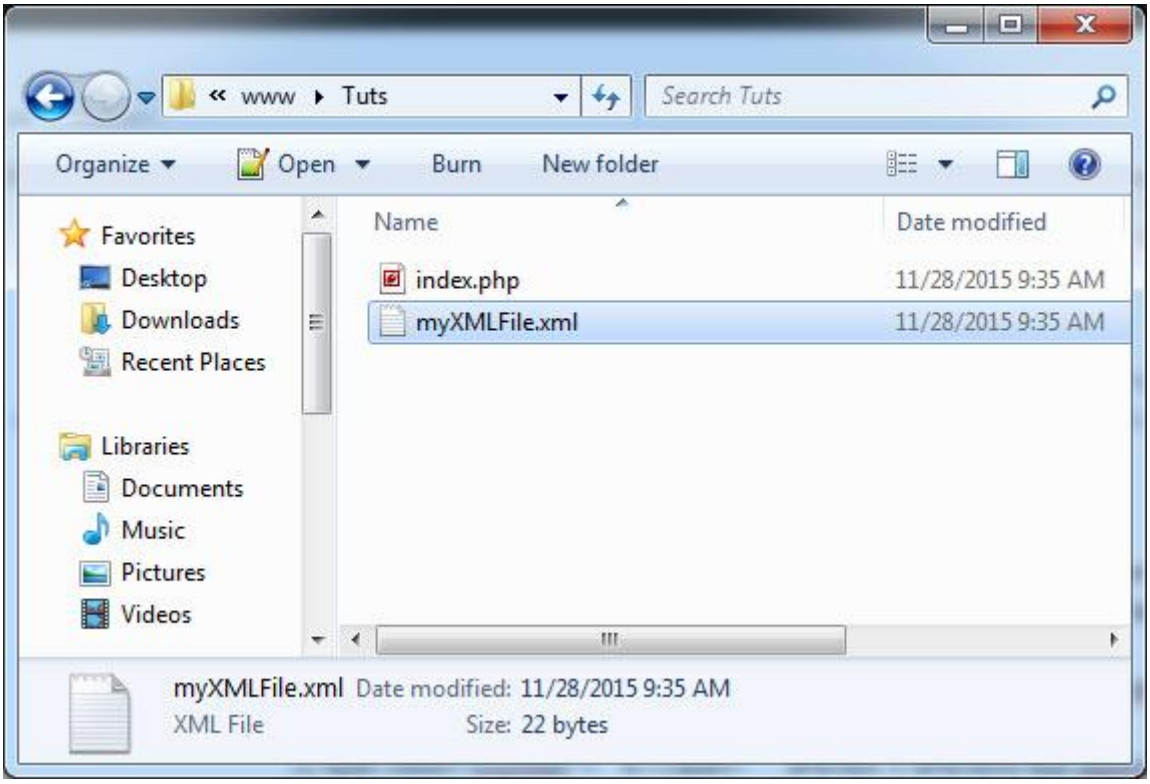
Document Object Model یک رابط برنامه نویسی برای سندهای XML و HTML است . با استفاده از DOM ، نحوه دستیابی و انجام پردازش های لازم در رابطه با سند های XML و HTML فراهم می گردد . برنامه نویسان با استفاده از DOM، قادر به ایجاد یک سند ،حرکت در طول ساختار سند، افزودن ، اصلاح و یا حذف عناصر (گره های) یک سند XML و یا HTMLمی باشند DOM . توسط کنسرسیوم وب استاندارد و بمنظور استفاده از طریق زبان های برنامه نویسی متعددی طراحی شده است . در PHP کلاس ها و متدهایی برای کار با سند XML وجود دارد که در این درس با آنها آشنا خواهید شد.

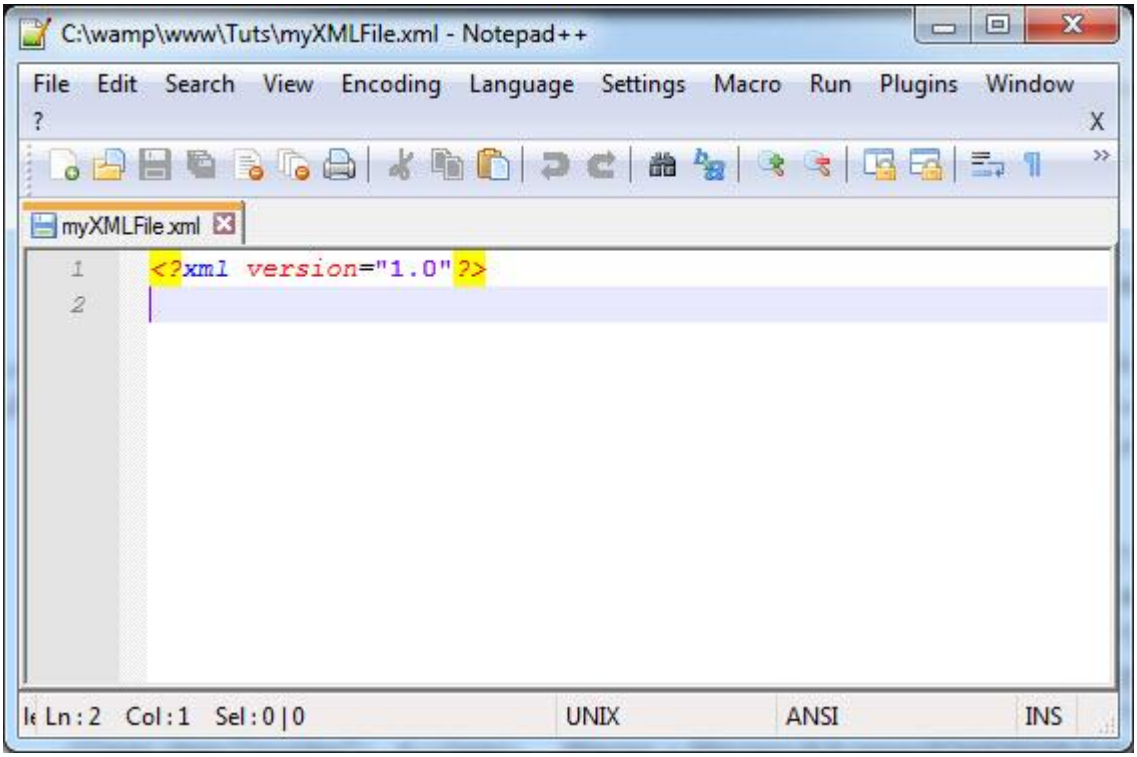
ایجاد فایلXML

برای ایجاد و ذخیره کردن یک فایل XML به صورت زیر عمل می شود:

```
<?php
    $DOM = new DOMDocument('1.0');

    $DOM->formatOutput = true;

    $DOMString = $DOM->saveXML();

    $DOM->save( 'myXMLFile.xml' );
?>
```

همانطور که در کد بالا مشاهده می کنید شیء ایی از کلاس DomDocument ایجاد کرده ایم و مقدار 1.0 را به آن ارسال کرده ایم. این کلاس مسئول به وجود آوردن سند XML است formatOutput .مسئول قالب بندی (ایجاد تو رفتگی) گره ها می باشد. متد saveXML() باعث ذخیره موقتی کدها در یک رشته و متد Save() باعث ذخیره نهایی آن در یک فایل می شود. با اجرای کد بالا ، فایلی به صورت زیر ایجاد می شود:


www ▸ Tuts
Search Tuts
Organize
Open
Burn
New folder
Favorites
Desktop
Downloads
Recent Places
Libraries
Documents
Music
Pictures
Videos
Name
Date modified
index.php
11/28/2015 9:35 AM
myXMLFile.xml
11/28/2015 9:35 AM
myXMLFile.xml Date modified: 11/28/2015 9:35 AM
XML File
Size: 22 bytes

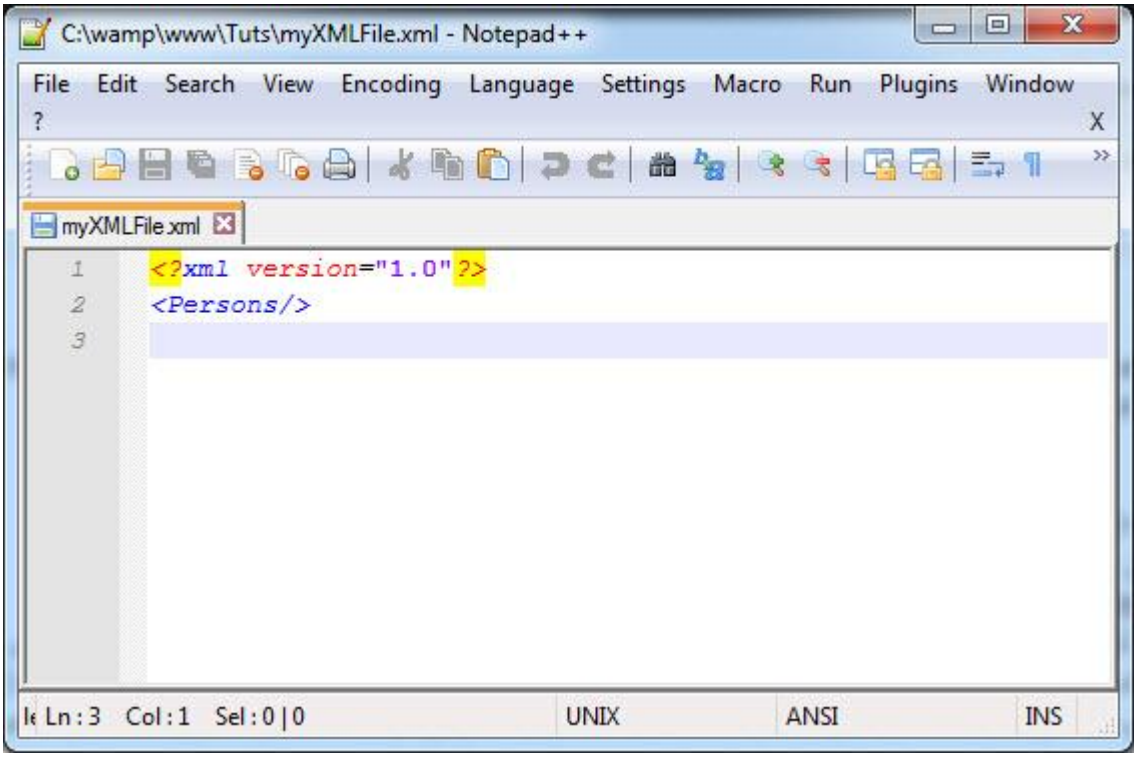

C:\wamp\www\Tuts\myXMLFile.xml - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window
myXMLFile.xml
1 <?xml version="1.0"?>
2
Ln : 2 Col : 1 Sel : 0 | 0
UNIX
ANSI
INS

ایجاد گره مادر(root)

برای ایجاد گره مادر در فایل XML به روش زیر عمل می شود:

```
$Persons = $DOM->appendChild($DOM->createElement('Persons'));
```

ابتدا یک متغیر تعریف می کنیم (چون با این متغیر در ادامه کار داریم) سپس نام شئ DOM$ را می آوریم. این بدین معناست که می خواهیم در سند ایجاد شده یک گره ایجاد کنیم. حال با فراخوانی متد appendChild() که مسئول اضافه کردن گره به سند است و ارسال خروجی تابع createElement() که گره را ایجاد می کند (از این متد خواسته ایم که یک گره با نام Persons ایجاد کند) به آن اولین گره یا همان گره مادر را ایجاد می کنیم.



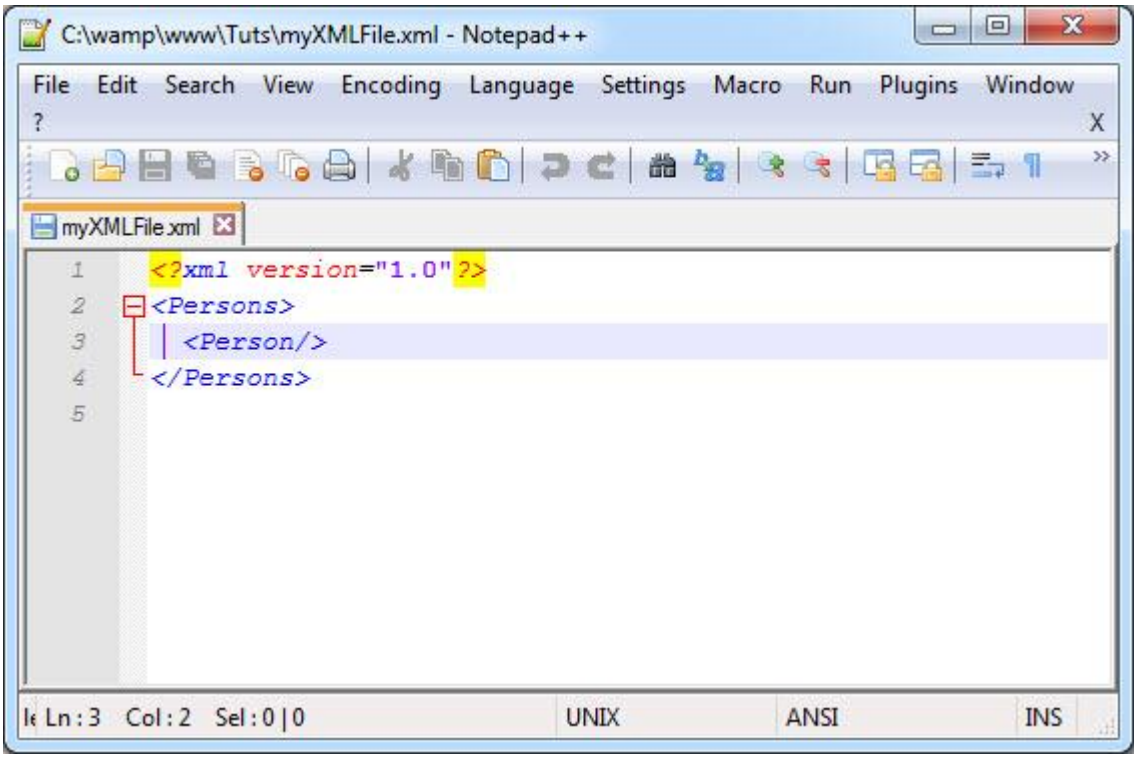
ممکن است این سوال برایتان پیش بیاید که چرا گره تگ پایانی ندارد؟ بعد از اضافه کرده زیر گره این گره به صورت خودکار بسته می شود.

ایجاد زیر گره

برای ایجاد زیر گره به گره مادر، در فایل XML به روش زیر عمل می شود:

```
$Person = $Persons->appendChild($DOM->createElement('Person'));
```

تنها تفاوت ایجاد زیر گره با گره مادر در این است که در سمت راست علامت مساوی نام گره مادر ایجاد شده در بالا (متغیر Persons) را می نویسیم.

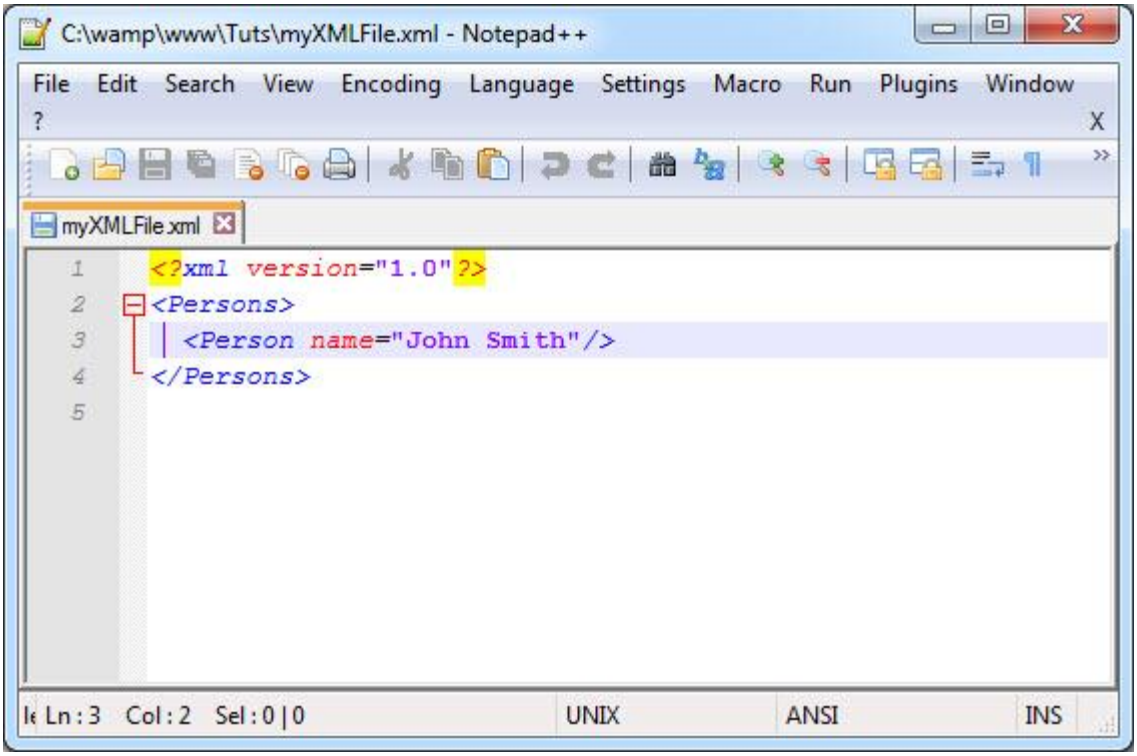


اضافه کردن خاصیت به گره

برای اضافه کردن خاصیت به گره به صورت زیر عمل می شود:

```
$Person ->setAttribute("name", "John Smith");
```

برای این کار ابتدا نام متغیری که گره در آن قرار داده شده است را نوشته و سپس با استفاده از متد **setAttribute()** خاصیتی به آن اختصاص می دهیم.

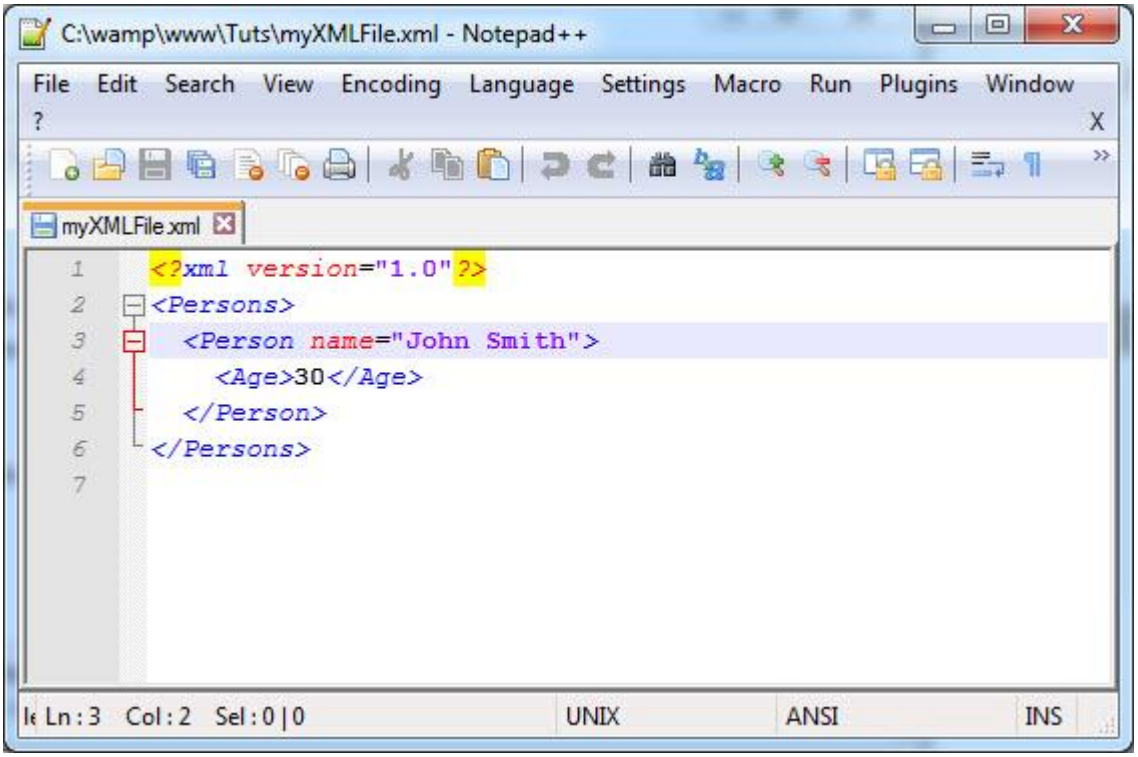


اضافه کردن مقدار به گره

فرض کنید که می خواهیم یک زیر گره به یک گره اضافه کرده و مقداری در داخل آن بنویسیم:

```
$Age    = $Person -> appendChild($DOM->createElement('Age'));
$Age    -> appendChild($DOM->createTextNode('30'));
```

در کد بالا یک گره به نام **Age** به گره **Person** اضافه می کنیم و سپس با استفاده از متد **createTextNode()** مقداری به آن اضافه می کنیم.

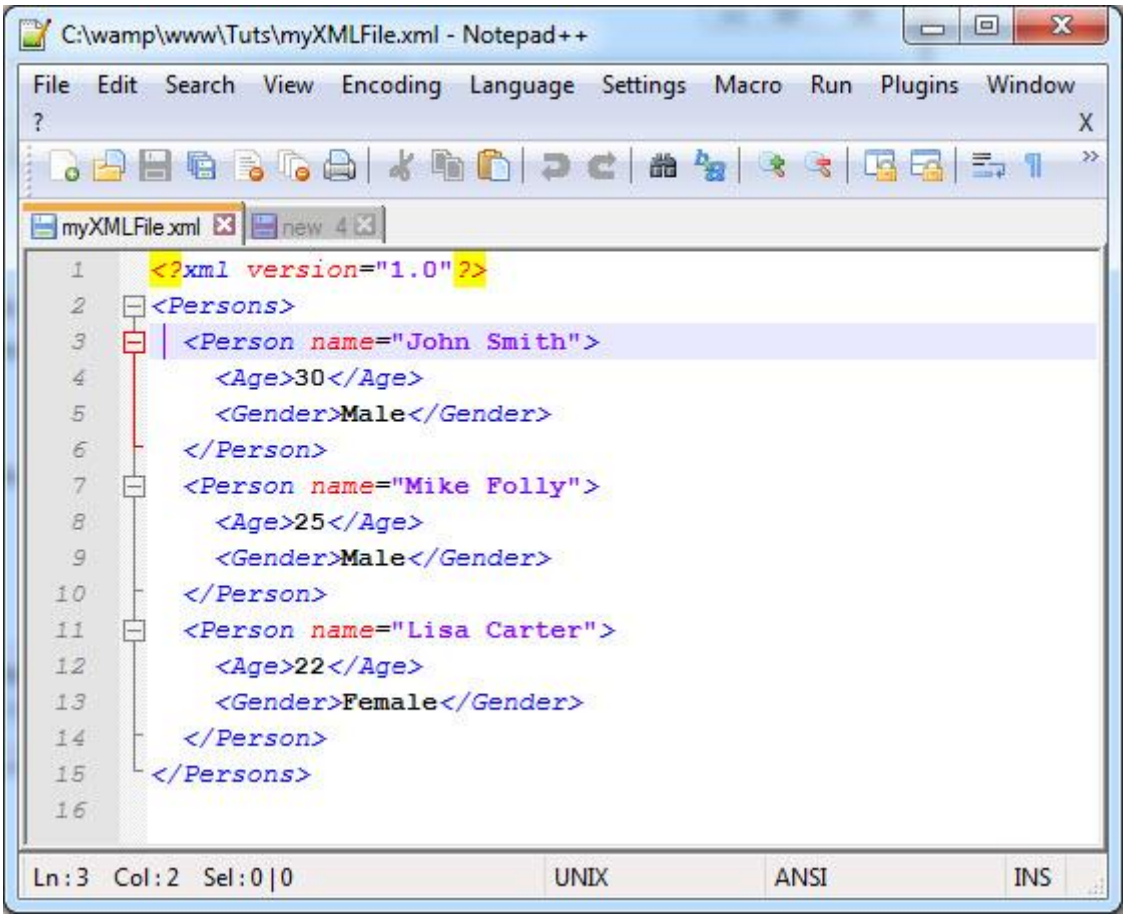


در زیر مثال کاملی با توجه به آموزش های بالا آورده شده است:

```
<?php
     $DOM = new DomDocument('1.0');

     $Persons = $DOM->appendChild($DOM->createElement('Persons'));

     $Person = $Persons->appendChild($DOM->createElement('Person'));
     $Person ->setAttribute("name", "John Smith");
     $Age    = $Person -> appendChild($DOM->createElement('Age'));
     $Age    -> appendChild($DOM->createTextNode('30'));
     $Gender = $Person -> appendChild($DOM->createElement('Gender'));
     $Gender -> appendChild($DOM->createTextNode('Male'));

     $Person = $Persons->appendChild($DOM->createElement('Person'));
     $Person ->setAttribute("name", "Mike Folly");
     $Age    = $Person -> appendChild($DOM->createElement('Age'));
     $Age    -> appendChild($DOM->createTextNode('25'));
     $Gender = $Person -> appendChild($DOM->createElement('Gender'));
     $Gender -> appendChild($DOM->createTextNode('Male'));

     $Person = $Persons->appendChild($DOM->createElement('Person'));
     $Person ->setAttribute("name", "Lisa Carter");
```

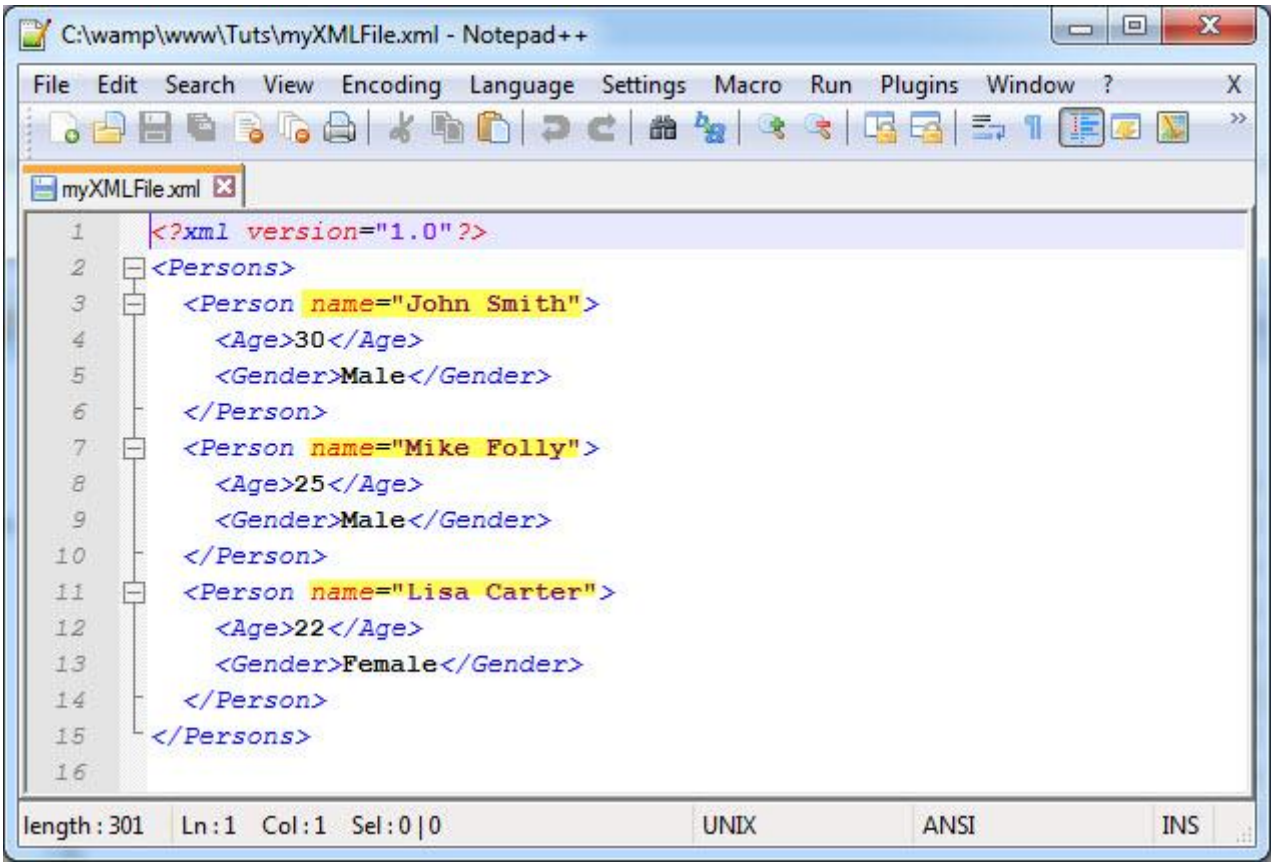
```
    $Age    = $Person -> appendChild($DOM->createElement('Age'));
    $Age    -> appendChild($DOM->createTextNode('22'));
    $Gender = $Person -> appendChild($DOM->createElement('Gender'));
    $Gender -> appendChild($DOM->createTextNode('Female'));

    $DOM->formatOutput = true;

    $DOMString = $DOM->saveXML();

    $DOM->save('myXMLFile.xml');
?>
```

نتیجه اجرای کد بالا:



به دست آوردن مقدار خاصیت یک گره

حال فرض کنید که می خواهیم مقدار خاصیت هایی که با رنگ زرد در شکل زیر مشخص شده است را به دست آوریم



برای این کار به صورت زیر عمل می کنیم:

```
<?php
    $DOM = new DOMDocument();
    $DOM->load( 'myXMLFile.xml' );

    $searchNode = $DOM->getElementsByTagName( "Person" );

    foreach( $searchNode as $searchNode )
    {
        $PersonAttributeName = $searchNode  -> getAttribute('name');

        $PersonAge           = $searchNode  -> getElementsByTagName(
"Age" );
        $PersonAgeValue      = $PersonAge   -> item(0)->nodeValue;

        $PersonGender        = $searchNode  -> getElementsByTagName(
```

```
16  "Gender" );
17          $PersonGenderValue    = $PersonGender-> item(0)->nodeValue;
18
19          echo $PersonName .'<br/>'. $PersonAgeValue .'<br/>'.
20  $PersonGenderValue.'<br/><br/>';
21      }
22  ?>
23
```

```
John Smith
30
Male

Mike Folly
25
Male

Lisa Carter
22
Female
```

در کد بالا و با استفاده از متد **Load()** ، فایل **XML** را باز یا بارگذاری می کنیم. سپس در خط **5** و با استفاده از متد **getElementsByTagName()** به عنصر یا عناصری که دارای نام **person** هستند دست می یابیم. با استفاده از متد **getAttribute()** در خط **10** مقدار خاصیت **name** گره ها و با استفاده از **nodeValue** مقدار موجود بین عناصر **Age** و **Gender** را به دست می آوریم.

به دست آوردن مقدار موجود در زیر گره های یک گره خاص

فرض کنید که می خواهیم مقدار موجود در زیر گره های **Age** و **Gender** را به دست آوریم برای این کار به صورت زیر عمل می کنیم :

```
<?php
     $DOM = new DOMDocument();

     $DOM->load( 'myXMLFile.xml' );

     $Person = $DOM->documentElement;

     $NodeText = $Person->getElementsByTagName('Person')->item(0);

     echo $NodeText->textContent;
 ?>
```

```
30 Male
```

حذف یک گره

برای حذف یک گره و زیر گره های آن از متد removeChild() به صورت زیر استفاده می شود:

```
<?php
    $DOM = new DOMDocument();

    $DOM->load( 'myXMLFile.xml' );

    $Person = $DOM->documentElement;

    $NodeToRemove = $Person->getElementsByTagName("Person")->item(0);
    $RemoveChild  = $Person->removeChild($NodeToRemove);

    $DOM->saveXML();

    $DOM->save('myXMLFile.xml');
?>
```

```
Mike Folly
25
Male

Lisa Carter
22
Female
```

در خط 6 کد بالا اعلام می کنیم که یک عنصر را می خواهیم. سپس در خط 8 آیتم صفرم یعنی اولین گره از گره هایی که نام آنها Person است را به آن انتصاب می دهیم. و در خط 9 با استفاده از متد removeChild() آن را حذف و در آخر فایل را ذخیره می کنیم. متدهای زیادی در PHP برای کار با فایل های XML وجود دارد که در سایت رسمی php.net لیست کامل آنها وجود دارد:

http://php.net/manual/en/book.dom.php

چون تمرکز ما بیشتر روی افزونه Simplexml است به همین مقدار بسنده می کنیم.

SimpleXML چیست

PHPچندین روش برای تجزیه فایل های XML ارائه می دهد که معمولترین آنها استفاده از SimpleXML است . SimpleXMLتمام محتویات فایل XML را خوانده و ان را تبدیل به آرایه ای از اشیاء می کند. امتیاز SimpleXML این است که برای خواندن محتویات فایل XML نیاز به کدنویسی ندارید. قبل از شروع به یادگیری SimpleXML نگاهی به برخی از متدهای پرکاربرد آن می اندازیم :

توابعSimpleXML

تابع	توضیح
__construct()	یک شیء جدید از SimpleXMLElement ایجاد می کند.
addAttribute()	یک خاصیت به عنصر خاص اضافه می کند.
addChild()	یک زیر گره به یک گره خاص اضافه می کند.
asXML()	فایل XML را در قالب یک رشته بر می گرداند.
attributes()	خاصیت ها/مقادیر یک عنصر را بر می گرداند.
children()	گره های فرزند یک گره خاص را بر می گرداند.
count()	تعداد زیر گره های یک گره خاص را می شمارد.
getDocNamespaces()	فضای نام تعریف شده در سند را بر می گرداند.
getName()	نام گره XML را بر می گرداند.
getNamespaces()	فضای نام استفاده شده در سند را بر می گرداند.
saveXML()	مترادفasXML()
simplexml_load_file()	یک فایل XML را به شیء SimpleXMLElement تبدیل می کند.
simplexml_load_string()	یک رشته XML را به شیء SimpleXMLElement تبدیل می کند.

توابع گردش درSimpleXML

تابع	توضیح
current()	گره جاری را بر می گرداند.
getChildren()	زیرگره های گره جاری را بر می گرداند.
hasChildren()	چک می کند که آیا گره جاری دارای زیر گره ای است یا نه؟
key()	کلید جاری را بر می گرداند.
next()	حرکت به سمت گره بعدی
rewind()	رفتن به عنصر اولی
valid()	چک می کند که آیا گره جاری معتبر است یا نه؟

فرض کنید می خواهیم همان فایل XML درس قبل را با استفاده از SimpleXml ایجاد کنیم:

```
1  <?php
2      $DOM =<<<XML
3  <?xml version="1.0"?>
4  <Persons>
5  <Person name="John Smith">
6  <Age>30</Age>
7  <Gender>Male</Gender>
8  </Person>
9  <Person name="Mike Folly">
10 <Age>25</Age>
11 <Gender>Male</Gender>
12 </Person>
13 <Person name="Lisa Carter">
14 <Age>22</Age>
15 <Gender>Female</Gender>
16 </Person>
17 </Persons>
18     XML;
19
20     $xml=new SimpleXMLElement($DOM);
21     $xml->saveXML('myXMLFile.xml');
22 ?>
```

همانطور که در کد بالا مشاهده می کنید تنها کاری که ما انجام داده ایم این است که ساختار مورد نظرمان را در داخل یک رشته (خطوط 2-18) نوشته و سپس به سازنده کلاس SimpleXMLElement خط 20) ارسال و در نهایت با استفاده از متد saveXML()ذخیره کرده ایم. به همین راحتی! با این اجرای کد بالا فایل XMLمورد نظرمان ایجاد می شود. همانطور که در کد بالا مشاهده می کنید گره اصلی (Persons) شامل سه گره فرزند (Person) است و هر گره فرزند خود دارای سه گره فرزند Name، Ageو Gender می باشد.

بارگذاری یا Load فایلXML

برای خواندن فایل XML از متد simplexml_load_file() استفاده می شود:

```
<?php
    $Persons = simplexml_load_file('c:/wamp/www/Tuts/myXMLFile.xml');
    echo '<pre>';
    print_r($Persons);
    echo '</pre>';
?>
```

```
SimpleXMLElement Object
(
    [Person] => Array
        (
            [0] => SimpleXMLElement Object
                (
                    [@attributes] => Array
                        (
                            [name] => John Smith
                        )

                    [Age] => 30
                    [Gender] => Male
                )

            [1] => SimpleXMLElement Object
                (
                    [@attributes] => Array
                        (
                            [name] => Mike Folly
                        )

                    [Age] => 25
                    [Gender] => Male
                )

            [2] => SimpleXMLElement Object
                (
                    [@attributes] => Array
                        (
                            [name] => Lisa Carter
                        )

                    [Age] => 22
                    [Gender] => Female
                )
```

```
        )
)
```

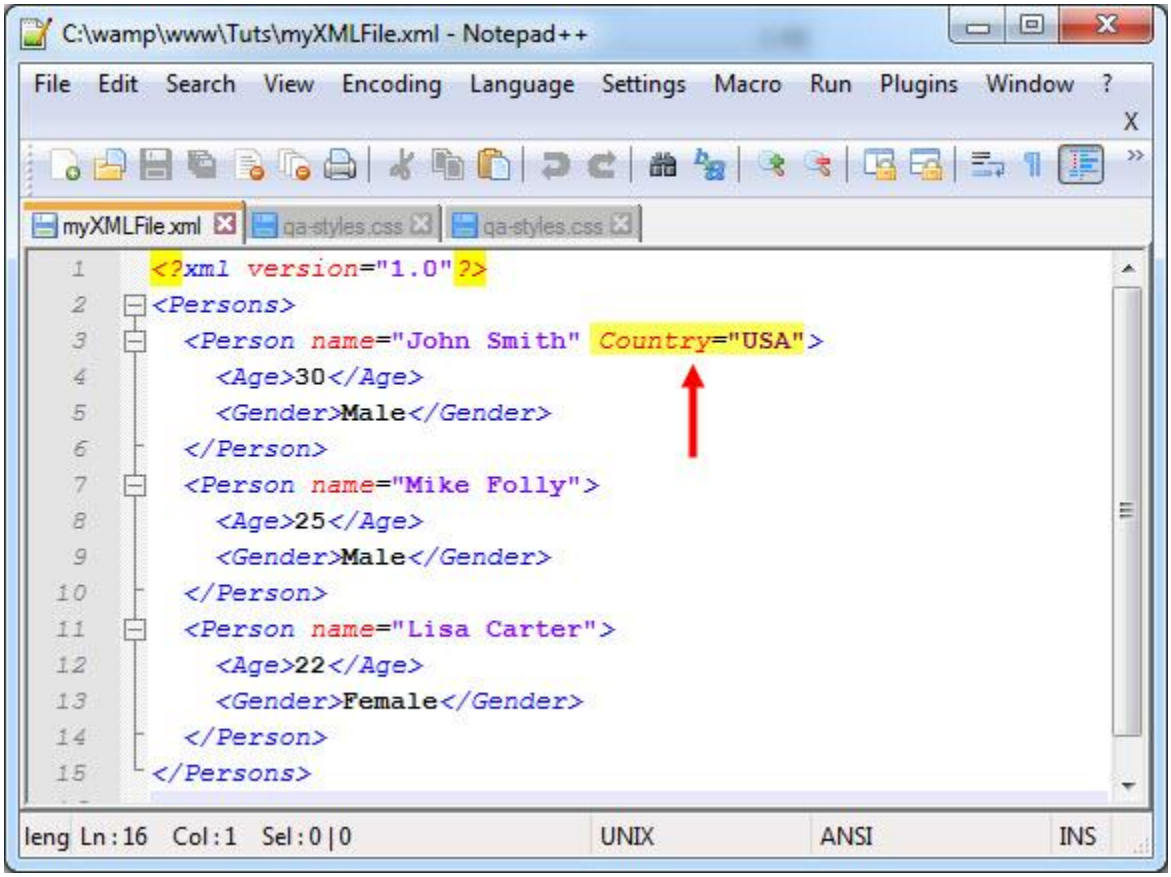
همانطور که در خروجی بالا مشاهده می کنید گره یا عنصر اصلی (Persons) آرایه ای از اشیاء Person است. که این آرایه خود دارای سه عنصر می باشد. که این عناصر همان گره های موجود در فایل XML هستند. هر عنصر در این آرایه یک شی است که شامل خاصیت هایName ، Ageو Gender می باشد. با این فایل XML در درس های آینده کار می کنیم.

متد addAttribute()

متد addAttribute() یک خاصیت به عنصر خاص اضافه می کند .مثلا برای اضافه کردن یک خاصیت به اولین عنصر person فایل XML ی که در بالا ایجاد کرده ایم به صورت زیر عمل می کنیم:

```
<?php
    $Persons = simplexml_load_file('c:/wamp/www/Tuts/myXMLFile.xml');

    $firstPerson = $Persons->Person[0];

    $firstPerson->addAttribute("Country","USA");

    $Persons->saveXML('myXMLFile.xml');
?>
```

با اجرای کد بالا یک خاصیت با نام Country و مقدار USA به اولین شخص یا گره اضافه می شود.

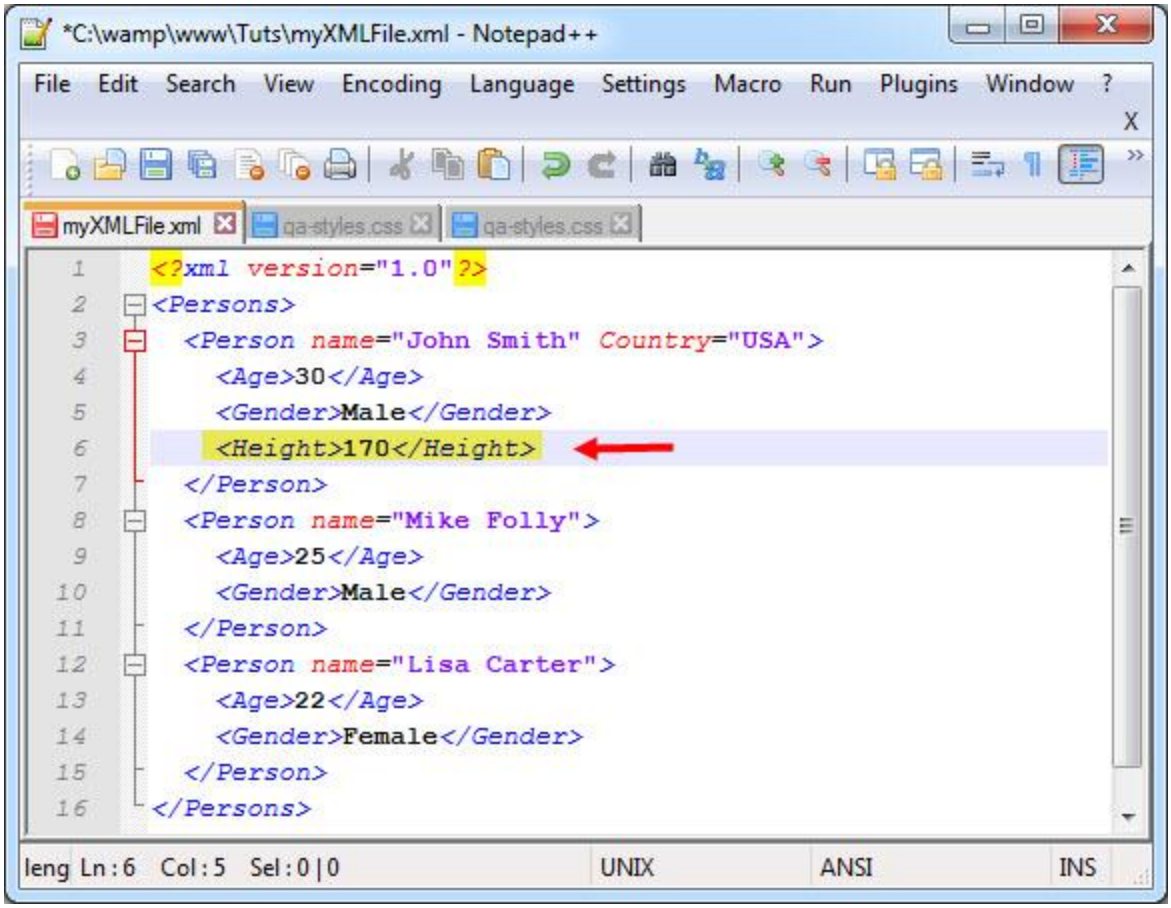


متد addChild()

متد **addChild()** یک گره به یک گره خاص اضافه می کند. مثلا برای اضافه کردن یک گره به اولین عنصر **person** فایل **XML**مان به صورت زیر عمل می کنیم:

```
<?php
    $Persons = simplexml_load_file('c:/wamp/www/Tuts/myXMLFile.xml');

    $firstPerson = $Persons->Person[0];

    $firstPerson->addChild("Height","170");

    $Persons->saveXML('myXMLFile.xml');
?>
```

با اجرای کد بالا فایل **XML** به صورت زیر در می آید:



متد Attributes()

متد Attributes() خاصیت ها/مقادیر یک عنصر را بر می گرداند.

```
<?php
    $Persons = simplexml_load_file('c:/wamp/www/Tuts/myXMLFile.xml');

    $Attributes = $Persons->Person[0]->attributes();

    echo $Attributes;
?>
```

```
John Smith
```

هانطور که در کد بالا مشاهده می کنید در حالت پیشفرض این متد مقدار اولین خاصیت را بر می گرداند. مثلا گره Person دو خاصیت name و Country دارد که مقدار خاصیت name را بر گرداند. برای به دست اوردن یک خاصیت خاص مثل Countryباید نام آن را بنویسید:

```
<?php
    $Persons = simplexml_load_file('c:/wamp/www/Tuts/myXMLFile.xml');

    $Attributes = $Persons->Person[0]->attributes()->Country ;

    echo $Attributes;
?>
```

```
USA
```

و برای به دست آوردن تمامی مقادیر خاصیت ها از حلقه foreach به صورت زیر استفاده می کنیم:

```
<?php
    $Persons = simplexml_load_file('c:/wamp/www/Tuts/myXMLFile.xml');

    foreach($Persons->Person[0]->attributes() as $key => $value)
    {
        echo $value .'<br/>';
    }
?>
```

```
John Smith
USA
```

متد children()

متد children() مقادیر موجود در زیرگره های یک گره خاص را بر می گرداند. مثلا گره Person دارای سه زیر گره است و می خواهیم مقادیر موجود در آنها را به دست آوریم:

```
<?php
    $Persons = simplexml_load_file('c:/wamp/www/Tuts/myXMLFile.xml');

    foreach($Persons->Person[0]->children() as $child)
    {
        echo $child .'<br/>';
    }
?>
```

```
30
Male
170
```

حال اگر به فایل XML مراجعه کنید، متوجه می شوید که مقادیر موجود در Age و Gender و Height برگشت داده شده است .

متد count()

متد **count()** تعداد زیر گره های یک گره را بر می گرداند. مثلا تعداد زیر گره های اولین گره **Person** به صورت زیر محاسبه می شود:

```
<?php
    $Persons = simplexml_load_file('c:/wamp/www/Tuts/myXMLFile.xml');

    $childNumber= $Persons->Person[0]->children();

    echo $childNumber -> count();
?>
```

```
3
```

زیر گره های **Person** عبارتند از**Age** ، **Gender**و**Height.**

متد getName()

از متد **getName()** برای به دست آوردن نام یک گره یا زیر گره استفاده می شود. فرض کنید که می خواهیم نام تمام زیر گره های ، اولین گره **Person** را به دست بیاوریم:

```
<?php
    $Persons = simplexml_load_file('c:/wamp/www/Tuts/myXMLFile.xml');

    foreach ($Persons->Person[0]->children() as $child)
    {
        echo $child->getName() . "<br>";
    }
?>
```

```
Age
Gender
Height
```

MYSQL چیست؟

بیشتر برنامه های امروزی از روش های مختلفی برای ذخیره سازی داده ها استفاده می کنند. یک برنامه می تواند دارای انواع مختلفی از منابع داده مانند فایل text و فایل XML و همچنین یک database باشد. از Databaseها معمولا برای ذخیره انواع داده مانند نام ،آدرس، سن، جنس و شغل یک شخص، آهنگ، تصویر و بسیاری چیزهای دیگر استفاده می شود.

یک دیتابیس مجموعه ای است از انواع مختلف داده های ساخت یافته در داخل جداولی که شامل فیلدها و رکوردها می باشند. بیشتر برنامه های امروزی از دیتابیس برای ذخیره اطلاعات استفاده می کنند. دیتابیس های رابطه ای شامل داده های هستند که به صورت سازمان یافته با همدیگر در ارتباط هستند. این نوع دیتا بیس ها شامل یک یا چند جدول به هم پیوسته هستند. جداول شامل سطر و ستون هستند. در دیتابیس ها یک سطر نشان دهنده یک رکورد است. به عنوان مثال در یک دیتابیس که شامل رکوردهای کارمند است، یک سطر نشان دهنده یک رکورد از یک کارمند است. ستون نشان دهنده فیلدها یا خواص و صفت می باشد. به عنوان مثال یک کارمند دارای یک فیلد مثلا FirstName (نام) و یک فیلد LastName (نام خانوادگی) و یک فیلد Age (سن)است. می توانید بین چندین جدول ارتباط برقرار کنید. به عنوان مثال یک جدول کارمند می تواند دارای یک فیلد به نام City_IDباشد.سپس جدول دیگر به نام Cities می تواند شامل فیلدهای City_ID و CityName باشد. شما می توانید بین این دو جدول ارتباط برقرار کنید. MYSQL یکی از استاندارد ترین راه های برقراری ارتباط با دیتابیس می باشد .این زبان دارای دستوراتی است که شما به وسیله آنها می توانید داده های دیتابیس را بروز کرده، بازیابی، اضافه و حذف کنید. همچنین به شما اجازه ایجاد و تغییر دیتابیس و جداول و ایجاد ارتباط بین جداول مختلف را می دهد. Database Management Systemsیا DBMS مانند MYSQL به شما اجازه دسترسی سریع به داده های دیتابیس را می دهد و شامل ابزارهای مختلفی برای پرس و جو، ایجاد، حذف و آپدیت دیتابیس می باشد. بیشتر DBMS ها یک محیط گرافیکی جهت انجام امور مختلف برای شما فراهم می آورند. مثالهایی از DBMS عبارتند از Access، Oracle و SQL SERVER.

MYSQL توسط شرکت اوراکل توسعه، توزیع، و پشتیبانی می‌شود. در درسهای آینده پروژه هایی ارائه شده است که با مطالعه انها نحوه اتصال به دیتابیس MYSQL و دستکاری داده ها را خواهید آموخت.

مبانی MYSQL

قبل از اینکه اقدام به دسترسی و تغییر دیتابیس ها کنید لازم است در مورد مبانیMYSQL اطلاعاتی داشته باشید. از این زبان برای پرس و جو در دیتابیس استفاده می شود.درک MYSQL بسیار ساده و آسان است. از MYSQL تنها برای پرس و جوی داده استفاده نمی شود، بلکه تقریبا از آن برای انجام هر کاری مثلا ایجاد، بروزرسانی و حذف دیتابیس و جداول استفاده می شود. از آن جاییکه این کتاب در مورد MYSQL نیست فقط در مورد مباحثی از آن که مورد نیازمان است توضیح می دهیم.

در MYSQL دستوراتی برای کار با دیتابیس وجود دارد که در زیر به آنها اشاره شده است:

- **CREATE : برای ایجاد یک دیتابیس ، جدول و یا یک اندیس به کار می رود.**
- **ALTER : برای تغییر در ساختار جدول به کار می رود.**
- **DROP : برای حذف یک دیتابیس و یا یک جدول به کار می رود.**

این زبان همچنین دارای دستورات زیر برای کار با داده های دیتابیس می باشد:

- **SELECT : برای واکشی اطلاعات از یک یا چند جدول به کار می رود.**
- **INSERT : برای درج اطلاعات در جدول به کار می رود.**
- **REPLACE : برای جایگزین کرده داده ها به کار می رود. اگر داده ای از قبل وجود داشته باشد ان را حذف و مقدار جدید را جایگزین می کند.**
- **UPDATE : برای به روز رسانی داده های جدول به کار می رود.**
- **DELETE : برای حذف داده از جدول به کار می رود.**

قبل از پرداختن به دستورات بالا، بهتر است که اجازه دهید که شما را با انواع داده ها در MYSQL آشنا کنم MYSQL . همانند سایر زبان های برنامه نویسی دارای انواعی از داده هاست که به سه نوع متنی، عددی و تاریخی تقسیم می شوند و لیست آنها در جدول زیر نمایش داده شده است:

نوع	توضیح	داده
عددی	داده عددی معمول که می‌تواند با نشانه یا بدون نشانه باشد (مثبت یا منفی) . اگر از نوع نشانه دار باشد محدوده مجاز اعداد بین -2147483648 تا 2147483647 می‌باشد ، اگر بدون نشانه باشد محدوده مجاز اعداد از 0 تا 4294967295 می‌باشد . می‌توانید تعداد ارقام اعداد را تا 11 عدد مشخص کنید.	INT
عددی	عدد صحیح خیلی کوچک که می‌تواند نشانه دار یا بدون نشانه (مثبت یا منفی) باشد . اگر نشانه دارد باشد محدوده مجاز اعداد بین -128 تا 127 می‌باشد . اما اگر بدون نشانه باشد محدوده مجاز اعداد بین 0 تا 255 می‌باشد . می‌توانید تعداد ارقام را تا 4 رقم مشخص کنید.	TINYINT
عددی	عدد صحیح کوچک می‌تواند نشانه دار یا بدون نشانه باشد . اگر نشانه دار باشد محدوده مجاز اعداد بین -32768تا 32767 می‌باشد . اگر بدون نشانه باشد محدوده مجاز اعداد بین 0 تا 65535می‌باشد .	SMALLINT

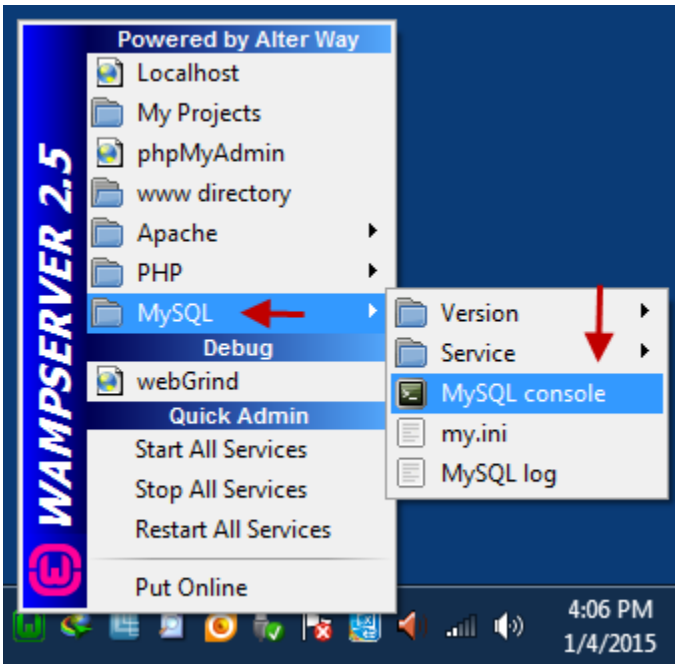
	می‌توانید تعداد ارقام را تا 5 رقم مشخص کنید.	
عددی	عدد صحیح متوسط که می‌تواند نشانه دار یا بدون نشانه باشد . اگر نشانه دار باشد محدوده مجاز اعداد بین 8388608-تا 8388607 می‌باشد . اگر بدون نشانه باشد محدوده مجاز اعداد بین 0 تا 16777215 می‌باشد . می‌توانید تعداد ارقام را تا 9 رقم مشخص کنید	MEDIUMINT
عددی	عدد صحیح بزرگ می‌تواند نشانه دار یا بدون نشانه باشد . اگر نشانه دار باشد محدوده مجاز اعداد بین -9223372036854775808تا 9223372036854775807 می‌باشد . اگر بدون نشانه باشد محدوده مجاز اعداد بین 0 تا 9223372036854775807 می‌باشد . تعداد ارقام را می‌توانید تا 20 رقم مشخص کنید.	BIGINT
عددی	عدد اعشاری که نمی‌تواند بدون علامت باشد . میتوان طول ارقام (M) و تعداد اعشار (D) را تعریف کرد . تعریف طول و تعداد اعشار اجباری نمی باشد و در صورت عدم مقدار پیش‌فرض 10,2 اشتفاده می‌شود که 2 تعداد ارقام اعشار و 10 تعداد کل ارقام می‌باشد (با احتساب ارقام اعشار).دقت اعشار می‌تواند تا 24 عدد برای FLOAT باشد.	FLOAT
عددی	عدد اعشاری با دقت مضاعف نمی‌تواند بدون علامت باشد . می‌توان طول ارقام (D) و تعداد اعشار (D) را تعریف کرد . تعریف طول و تعداد اعشار اجباری نمی باشد و در صورت عدم تعریف مقدار پیش‌فرض 16,4 اشتفاده می‌شود که 4 تعداد ارقام اعشار و 16 تعداد کل ارقام می‌باشد (با احتساب اعداد اعشار) . دقت اعشار می‌تواند تا 54 عدد برای DOUBLE باشد REAL . مترادف برای DOUBLE است.	DOUBLE
عددی	عدد اعشاری با ممیز شناور که نمی‌تواند بدون علامت باشد . در اعشار ممیز شناور هر عدد اعشار معادل یک بایت می‌باشد . تعریف طول ارقام (M) و تعداد اعشار (D) می‌باشد NUMERIC.مترادف برای DECIMALمی‌باشد.	DECIMAL
تاریخ	تاریخ با فرمت YYYY-MM-DD ، که محدوده آن بین 1000-01-01 تا 9999-12-31 می‌باشد . برای مثال December 30th, 1973 به صورت 1973-12-30. ذخیره می‌شود.	DATE
تاریخ	ترکیبی از تاریخ و زمان با فرمت YYYY-MM-DD HH:MM:SS می‌باشد ، محدوده آن بین 1000-01-01 00:00:00 و 9999-12-31 23:59:59می‌باشد . برای مثال ، 3:30 بعد از ظهر در تاریخ December 30th, 1973به صورت 1973-12-30 15:30:00 ذخیره می‌شود.	DATETIME
تاریخ	محدوده زمانی بین نیمه شب January 1, 1970و زمانی در 2037 می‌باشد . فرمت آن شبیه فرمت DATETIMEمی‌باشد بودن خط تیره در میان اعداد . برای مثال ، 3:30 بعد از ظهر در تاریخ December 30th, 1973به صورت 19731230153000 ذخیره می‌شود.	TIMESTAMP
تاریخ	زمان را با فرمت HH:MM:SS ذخیره می کند.	TIME
تاریخ	سال را با فرمت 2 رقم یا 4 رقم ذخیره می‌کند . اگر طول 2 تعیین شود ، YEARمی‌تواند بین1970 تا 2069 70)تا 69) باشد . اگر طول 4 تعیین شود YEAR میتواند بین 1901 تا 2155 باشد .طول پیش‌فرض 4 می‌باشد.	YEAR
رشته	رشته با طول ثابت که طول آن بین 1 تا 255 می‌باشد .(مثال (CHAR(5) : ، مقدار خالی سمت راست با فضای خالی پر می‌شود تا زمانی که به حداکثر طول تعیین شده برسد . تعریف طول رشته اجباری نیست اما مقدار پیش‌فرض 1 می‌باشد.	CHAR

رشته	رشته با طول منغیر که طول آن می‌تواند بین 0 تا 255 باشد . (مثال . (VARCHAR(25) : تعریف مقدار طول رشته اجباری می‌باشد.	VARCHAR
رشته	فیلد با حداکثر مقدار 65535 کاراکتر BLOB . مخفف ”Binary Large Object“ می‌باشد و برای ذخیره مقادیر بزرگ داده باینری استفاده می‌شود ، از قبیل تصویر یا انواع مختلف فایل‌ها . فیلدهایی که عنوان TEXT ذخیره شده‌اند نیز مقدار بزرگی از داده را ذخیره می‌کنند . تفاوت آن‌ها در این می‌باشد که مرتب سازی و مقایسه در فیلد های BLOBبین حروف کوچک و بزرگ تفاوت قائل می‌شود اما در فیلدهای TEXT تفاوتی قائل نمی‌شود . برای نوع های TEXT و BLOB طول تعریف نمی‌شود.	BLOB / TEXT
رشته	یک ستون TEXT یا BLOB با حداکثر مقدار 255 کاراکتر . برای نوع های TINYTEXT و TINYBLOB، طول تعریف نمی‌شود.	TINYBLOB / TINYTEXT
رشته	یک ستون TEXT یا BLOB با حداکثر مقدار 16777215 کاراکتر . برای نوع های MEDIUMTEXTو MEDIUMBLOB ، طول تعریف نمی‌شود.	MEDIUMBLOB / MEDIUMTEXT
رشته	یک ستون TEXT یا BLOB با حداکثر مقدار 4294967295 کاراکتر . برای نوع های LONGTEXTو LONGBLOB ، طول تعریف نمی‌شود.	LONGBLOB / LONGTEXT
رشته	یک نوع شمارشی می باشد ، که یک اصطلاح عامیانه برای لیست است . هنگامی که یک فیلد ENUM تعریف می‌کنید ، شما لیستی از گزینه ها ایجاد می‌کنید که یک مورد باید انتخاب شود (یا می‌تواند NULLباشد) . برای مثال شما می‌خواهید فیلد دارای مقدار ”A“ یا ”B“یا ”C“ باشد ، فیلد ENUMباید به صورت ENUM(‘A’,’B’,’C’) تعریف شود و تنها همان مقادیر تعیین شده (یا NULL)می‌تواند برای آن فیلد جمع آوری شود.	ENUM
رشته	نوع SET نیز همانند ENUM می باشد و تنها تفاوت آن در این می باشد که امکان انتخاب چند گزینه از لیست را نیز میسر می سازد.	SET

حال اکه با انواع داده ها آشنا شدید، در درس های بعدی در مورد نحوه ایجاد و ویرایش و حذف دیتابیس و جدول به دو روش کد نویسی و استفاده از محیط PHPMyAdminمی پردازیم.

ایجاد جدول و دیتابیس به روش کدنویسی

در این درس می خواهیم یک دیتابیس که شامل یک جدول و چندین رکورد است را ایجاد کنیم. قصد داریم که ایجاد دیتابیس و جدول را از دو را به شما آموزش دهیم.یک بار به روش کدنویسی و بار دیگر با استفاده از محیط گرافیکی **phpmyadmin.** جهت ورود به بخش کدنویسی به صورت زیر بر روی آیکون **wamp** کلیک کرده و با طی مراحل زیر وارد بخش کدنویسی شوید :



ایجاد دیتابیس

می توان با استفاده از دستور زیر یک جدول ایجاد کرد. به این نکته توجه کنید که **MYSQL** به بزرگی و کوچکی حروف حساس نیست. بنابراین مهم نیست که کلمات را به صورت کوچک بنویسید.

```
CREATE DATABASE<Name of Database>;
```

<Name of Database> نامی است که شما برای دیتابیس تان انتخاب می کنید. به مثال زیر توجه کنید :

```
CREATE DATABASE MyDatabase;
```

برای ذخیره سازی داده ها در دیتابیس لازم است جدول ایجاد کنیم. جدول در **MYSQL** به صورت زیر ایجاد می شود.

```
CREATE TABLE<Name of Table>
(
    <Column1><Datatype>,
    <Column2><Datatype>,
       .
       .
       .
    <ColumnN><Datatype>
);
```

<Name of Table> نامی است که شما برای جدول در نظر گرفته اید. در داخل پرانتز لیستی از ستون ها و نوع داده های آنها قرار دارد. در این بخش می خواهیم یک دیتابیس با نام University که شامل جدول Students است را ایجاد کنیم. برای ان کار دستور SQL زیر را می نویسیم :

```
mysql> CREATE DATABASE University;
```

بعد از علامت سمیکالن و زدن دکمه Enter دیتابیسی بانام University ایجاد می شود به همین راحتی!

ایجاد جدول

برای ایجاد جدول student که شامل جزییاتی در مورد چندین دانش آموز است ابتدا باید از ایجاد دیتابیس مطمئن شوید برای این کار دستور زیر را تایپ کنید:

```
mysql> USE University;
```

معنای دستور بالا این است که ما می خواهیم تغییراتی در دیتابیسی به نام Universityبدهیم. همچنین یک پیغام مبنی بر اعمال تغییرات به شما نشان داده می شود. اجازه دهید که جدول Students را ایجاد کنیم. دستور MYSQL زیر را می نویسیم:

```
mysql> CREATE TABLE Students
->(
->     studentID int AUTO_INCREMENT primary key,
->     FirstName varchar(50),
->     LastName varchar(50),
->     Gender varchar(10),
->     Age int,
->     Address varchar(50)
->);
```

جدول ایجاد شده دارای 6 ستون می باشد. به کلمه primary key که بعد از نوع داده ای studentID آورده شده است توجه کنید. این بدین معنی است که studentID کلید اصلی در جدول Students است. یک primary key مقداری است که یک سطر یا رکورد را نشان می دهد. این مقدار باید برای هر رکورد، یکتا باشد. ما از studentID به عنوان کلید اصلی استفاده کرده ایم نه از FirstName، چون نام کوچک (FirstName) چندین دانش آموز می تواند شبیه به هم باشد بنابراین در عملکرد کلید اصلی (primary key) تناقض به وجود می آید، چون که مقدار کلید اصلی برای هر رکورد باید یکتا بوده و در رکورد دیگری عین همین مقدار وجود نداشته باشد. دیگر ستون های آن به ترتیب عبارتند از FirstName ، LastName، Gender، Age و Address که نوع آنها مشخص شده است. با اجرای دستور فوق جدول ایجاد می شود.

نمایش دیتابیس

برای نمایش تمامی دیتابیس ها و اطمینان از ایجاد جدول بالا از دستور Show Databases به صورت زیر استفاده می کنیم:

```
mysql> Show Databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| phpmyadmin         |
| university         |
| wp                 |
+--------------------+
```

مشاهده می کنید که جدولی با نام university هم ایجاد شده است.

نمایش جدول

برای نمایش جداول یک دیتابیس خاص مانند university ابتدا باید دیتابیس را انتخاب و سپس با استفاده از دستور Show Tables جداول آن را نمایش دهیم:

```
mysql> use university;

mysql> Show Tables;
+----------------------+
| Tables_in_university |
+----------------------+
| students             |
+----------------------+
```

ویرایش جدول

قبل از ایجاد تغییر در جدول اجازه دهید که ستون های جدول students را نمیش دهیم، برای این کار از دستور زیر استفاده می کنیم:

```
mysql> SHOW COLUMNS FROM students;
+-----------+-------------+------+-----+---------+----------------+
| Field     | Type        | Null | Key | Default | Extra          |
+-----------+-------------+------+-----+---------+----------------+
| studentID | int(11)     | NO   | PRI | NULL    | auto_increment |
| FirstName | varchar(50) | YES  |     | NULL    |                |
| LastName  | varchar(50) | YES  |     | NULL    |                |
| Gender    | varchar(10) | YES  |     | NULL    |                |
| Age       | int(11)     | YES  |     | NULL    |                |
| Address   | varchar(50) | YES  |     | NULL    |                |
+-----------+-------------+------+-----+---------+----------------+
```

حال فرض کنید که می خواهیم به جدول students یک ستون دیگر به نام city اضافه کنیم. برای این کار از دستور ALTER به صورت زیر اضافه می کنیم:

```
mysql> ALTER TABLE students ADD COLUMN city VARCHAR(50);
```

ALTER به معنای (تغییر) می باشد و در کل دستور بالا به این معناست که "می خواهم در جدول students تغییری ایجاد کنم و آن هم اضافه کردن ستون city است."برای اطمینان از اضافه شدن ستون بار دیگر ستونهای جدول را نمایش می دهیم :

```
mysql> SHOW COLUMNS FROM students;
+-----------+-------------+------+-----+---------+----------------+
| Field     | Type        | Null | Key | Default | Extra          |
+-----------+-------------+------+-----+---------+----------------+
| studentID | int(11)     | NO   | PRI | NULL    | auto_increment |
| FirstName | varchar(50) | YES  |     | NULL    |                |
| LastName  | varchar(50) | YES  |     | NULL    |                |
| Gender    | varchar(10) | YES  |     | NULL    |                |
| Age       | int(11)     | YES  |     | NULL    |                |
| Address   | varchar(50) | YES  |     | NULL    |                |
| city      | varchar(50) | YES  |     | NULL    |                |
+-----------+-------------+------+-----+---------+----------------+
```

فرض کنید می خواهیم نوع همین ستون city را به text تغییر دهیم برای این کار از دستور MODIFY به صورت زیر استفاده می کنیم:

```
mysql> ALTER TABLE students MODIFY COLUMN city TEXT;
```

با این کد نوع ستون city از (50) varchar به text تغییر می کند. برای تغییر نام یک ستون از دستور CHANGE به صورت زیر استفاده می کنیم:

```
mysql> ALTER TABLE students CHANGE city street TEXT;
```

همانطور که در کد بالا مشاهده می کنید ابتدا نام قدیمی ستون و سپس نام جایگزین را بعد از کلمه کلیدی CHANGE می نویسید. و برای حذف یک ستون هم از دستور DROP به صورت زیر استفاده می کنید:

```
mysql> ALTER TABLE students DROP COLUMN city;
```

اضافه کردن رکوردها

حال که جدول مان را ایجاد کرده ایم اجازه دهید اطلاعات مربوط به چند دانش آموز را به آن اضافه کنیم. در زیر اطلاعات مربوط به چند دانش آموز که به جدول اضافه کرده ایم نشان داده شده است:

StudentID	FirstName	LastName	Gender	Age	Address
1	Edward	Lyons	Male	17	Spencer Street
2	Jimmie	Vargas	Male	18	Blue Bay Avenue
3	Monica	Ward	Female	16	Mapple Street
4	Joann	Jordan	Female	17	Spencer Street
5	Cheryl	Swanson	Female	17	Wacky Street
6	Clara	Webb	Female	18	Spooner Street
7	Zack	Norris	Male	19	Blue Bay Avenue
8	Randall	May	Male	18	Golden Street
9	Jessica	Cole	Female	17	Mapple Street
10	Oscar	Manning	Male	18	Mapple Street

حال که 10 رکورد داریم اجازه دهید آنها را با استفاده از دستورات SQL به جدول Students اضافه کنیم:

```
mysql> INSERT INTO Students (FirstName, LastName, Gender, Age, Address)
VALUES
->   ('Edward', 'Lyons', 'Male', 17, 'Spencer Street'),
->   ('Jimmie', 'Vargas', 'Male', 18, 'Blue Bay Avenue'),
->   ('Monica', 'Ward', 'Female', 16, 'Mapple Street'),
->   ('Joann', 'Jordan', 'Female', 17, 'Spencer Street'),
->   ('Cheryl', 'Swanson', 'Female', 17, 'Wacky Street'),
->   ('Clara', 'Webb', 'Female', 18, 'Spooner Street'),
```

```
->   ('Zack', 'Norris', 'Male', 19, 'Blue Bay Avenue'),
->   ('Randall', 'May', 'Male', 18, 'Golden Street'),
->   ('Jessica', 'Cole', 'Female', 17, 'Mapple Street'),
->   ('Oscar', 'Manning', 'Male', 18, 'Mapple Street')
-> ;
```

از یک نسخه اصلاح شده دستور INSERT INTO که به ما اجازه وارد کردن چندین رکورد را به صورت یکجا می دهد، استفاده می کنیم. هر رکورد در داخل پرانتز قرار دارد و رکوردها به وسیله کاما از هم جدا شده اند. بعد از اجرای دستورات SQL، جدول Students باید دارای 10 رکورد باشد. اگر همه چیز به درستی انجام شود، یک پیغام نشان داده می شود که نشان دهنده تعداد رکوردهایی است که دستکاری (اضافه) شده اند. اکنون می توانید این قسمت را رد کرده و به درس بعد بروید.

واکشی رکوردها

برای انجام یک پرس و جوی ساده در MYSQL می توانید از دستور SELECT استفاده نمایید. اگر بخواهید تمامی ستونهای جدول را نمایش دهید می توانید به صورت زیر عمل نمایید:

```
mysql> SELECT *  FROM Students;
+-----------+-----------+----------+--------+------+-----------------+
| studentID | FirstName | LastName | Gender | Age  | Address         |
+-----------+-----------+----------+--------+------+-----------------+
|         1 | Edward    | Lyons    | Male   |   17 | Spencer Street  |
|         2 | Jimmie    | Vargas   | Male   |   18 | Blue Bay Avenue |
|         3 | Monica    | Ward     | Female |   16 | Mapple Street   |
|         4 | Joann     | Jordan   | Female |   17 | Spencer Street  |
|         5 | Cheryl    | Swanson  | Female |   17 | Wacky Street    |
|         6 | Clara     | Webb     | Female |   18 | Spooner Street  |
|         7 | Zack      | Norris   | Male   |   19 | Blue Bay Avenue |
|         8 | Randall   | May      | Male   |   18 | Golden Street   |
|         9 | Jessica   | Cole     | Female |   17 | Mapple Street   |
|        10 | Oscar     | Manning  | Male   |   18 | Mapple Street   |
+-----------+-----------+----------+--------+------+-----------------+
```

اگر بخواهید ستون های خاصی را نمایش دهید می توانید نام آنها را به صورت زیر صریحا ذکر کنید:

```
mysql> SELECT FirstName, LastName FROM Students;
+-----------+------------+
| FirstName | LastName   |
+-----------+------------+
| Edward    | Lyons      |
| Jimmie    | Vargas     |
| Monica    | Ward       |
| Joann     | Jordan     |
| Cheryl    | Swanson    |
```

```
| Clara     | Webb       |
| Zack      | Norris     |
| Randall   | May        |
| Jessica   | Cole       |
| Oscar     | Manning    |
| amin      | mehdipoort |
+-----------+------------+
->
```

همانطور که در کد بالا مشاهده می کنید با استفاده از دستور MYSQL ی نام و نام خانوادگی افراد ثبت شده در جدول Studensنمایش داده ایم. اگر بخواهیم مشخصات یک شخص یا رکورد خاص در جدول را نمایش دهید می توانید از کلمه کلیدی WHERE(به معنای : به شرطی که) به صورت زیر استفاده کنید:

```
mysql> SELECT LastName FROM Students WHERE FirstName = 'Zack';
+----------+
| LastName |
+----------+
| Norris   |
+----------+
```

در کد بالا خواسته ایم که فامیلی شخصی که نام آن Zack است نمایش داده شود. دستورات دیگری در MYSQL برای نمایش و ارتباط هر چه بهتر با داده های جدول وجود دارد که یکی از آنها ORDER BY است. این دستور باعث مرتب شده اطلاعات نمایش داده شده بر اساس یک ستون حاص به صورت نزولی یا صعودی می شود. فرض کنید که می خواهیم اطلاعات جدول را به صورت صعودی و بر اساس ستون Gender نمایش دهیم:

```
mysql> SELECT *  FROM Students ORDER BY Gender;
+-----------+-----------+----------+--------+------+-----------------+
| studentID | FirstName | LastName | Gender | Age  | Address         |
+-----------+-----------+----------+--------+------+-----------------+
|         3 | Monica    | Ward     | Female |   16 | Mapple Street   |
|         4 | Joann     | Jordan   | Female |   17 | Spencer Street  |
|         6 | Clara     | Webb     | Female |   18 | Spooner Street  |
|         9 | Jessica   | Cole     | Female |   17 | Mapple Street   |
|         1 | Edward    | Lyons    | Male   |   17 | Spencer Street  |
|         2 | Jimmie    | Vargas   | Male   |   18 | Blue Bay Avenue |
|         7 | John      | Norris   | Male   |   19 | Blue Bay Avenue |
|         8 | Randall   | May      | Male   |   18 | Golden Street   |
|        10 | Oscar     | Manning  | Male   |   18 | Mapple Street   |
+-----------+-----------+----------+--------+------+-----------------+
```

مشاهده می کنید که چون حرف F در حروف انگلیسی قبل از حرف M است پس در ستون Gender ابتدا Female نمایش داده می شود و اگر بخواهید این ستون را به صورت نزولی مرتب کنید کافیست که بعد از نام ستون کلمه DESC را بنویسید:

```
mysql> SELECT *  FROM Students ORDER BY Gender DESC;
+-----------+-----------+----------+--------+------+-----------------+
| studentID | FirstName | LastName | Gender | Age  | Address         |
+-----------+-----------+----------+--------+------+-----------------+
|         1 | Edward    | Lyons    | Male   |   17 | Spencer Street  |
|         2 | Jimmie    | Vargas   | Male   |   18 | Blue Bay Avenue |
|         7 | John      | Norris   | Male   |   19 | Blue Bay Avenue |
|         8 | Randall   | May      | Male   |   18 | Golden Street   |
|        10 | Oscar     | Manning  | Male   |   18 | Mapple Street   |
|         3 | Monica    | Ward     | Female |   16 | Mapple Street   |
|         4 | Joann     | Jordan   | Female |   17 | Spencer Street  |
|         6 | Clara     | Webb     | Female |   18 | Spooner Street  |
|         9 | Jessica   | Cole     | Female |   17 | Mapple Street   |
+-----------+-----------+----------+--------+------+-----------------+
```

ویرایش رکوردها

برای ویرایش رکوردها از دستور UPDATE استفاده می شود. فرض کنید که می خواهیم نام شخصی در جدول که نامش Zack است را به John تغییر دهیم. برای این کار به صورت زیر عمل می کنیم:

```
mysql> UPDATE students SET FirstName='John' WHERE FirstName='Zack';
```

دستور بالا را به صورت چند ستونی هم می توان نوشت. مثلا می توان نام و نام خانوادگی و ... را هم به روش بالا و با یک دستور ویرایش کرد.

حذف رکوردها

برای حذف رکوردها از دستور DELETE استفاده می شود .فرض کنید می خواهیم یک رکورد که ID آن برابر 5 است (یعنی StudentID=5)را حذف کنیم. برای این کار به صورت زیر عمل می کنیم:

```
mysql> DELETE FROM students WHERE studentID = 5;
```

در دستور بالا از جدول students رکوردی که مقدار ستون studentID برابر 5 است، حذف می شود. برای اطمینان از حذف یک بار دیگر مقادیر جدول را نمایش می دهیم:

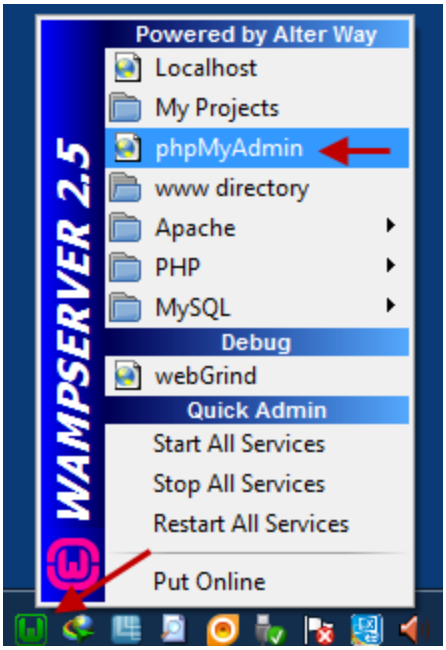
```
mysql> SELECT *  FROM Students;
+-----------+-----------+----------+--------+------+-----------------+
| studentID | FirstName | LastName | Gender | Age  | Address         |
```

```
+-----------+-----------+----------+--------+------+-----------------+
|         1 | Edward    | Lyons    | Male   |   17 | Spencer Street  |
|         2 | Jimmie    | Vargas   | Male   |   18 | Blue Bay Avenue |
|         3 | Monica    | Ward     | Female |   16 | Mapple Street   |
|         4 | Joann     | Jordan   | Female |   17 | Spencer Street  |
|         6 | Clara     | Webb     | Female |   18 | Spooner Street  |
|         7 | John      | Norris   | Male   |   19 | Blue Bay Avenue |
|         8 | Randall   | May      | Male   |   18 | Golden Street   |
|         9 | Jessica   | Cole     | Female |   17 | Mapple Street   |
|        10 | Oscar     | Manning  | Male   |   18 | Mapple Street   |
+-----------+-----------+----------+--------+------+-----------------+
```

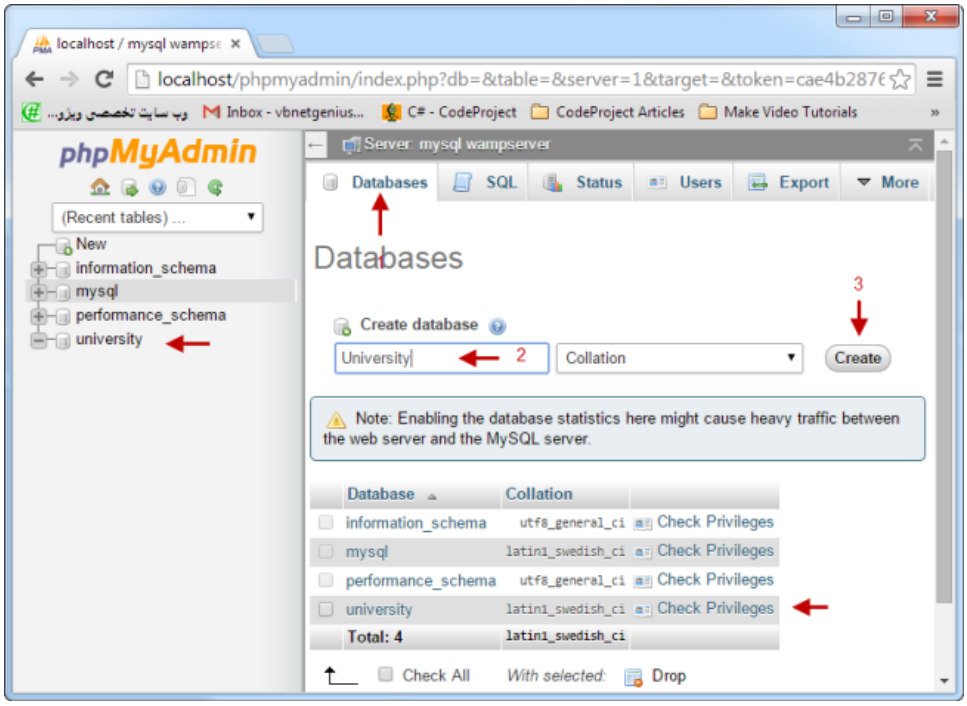
مشاهده می کنید که رکورد 5 حذف شده است.

ایجاد جدول و دیتابیس با phpMyAdmin

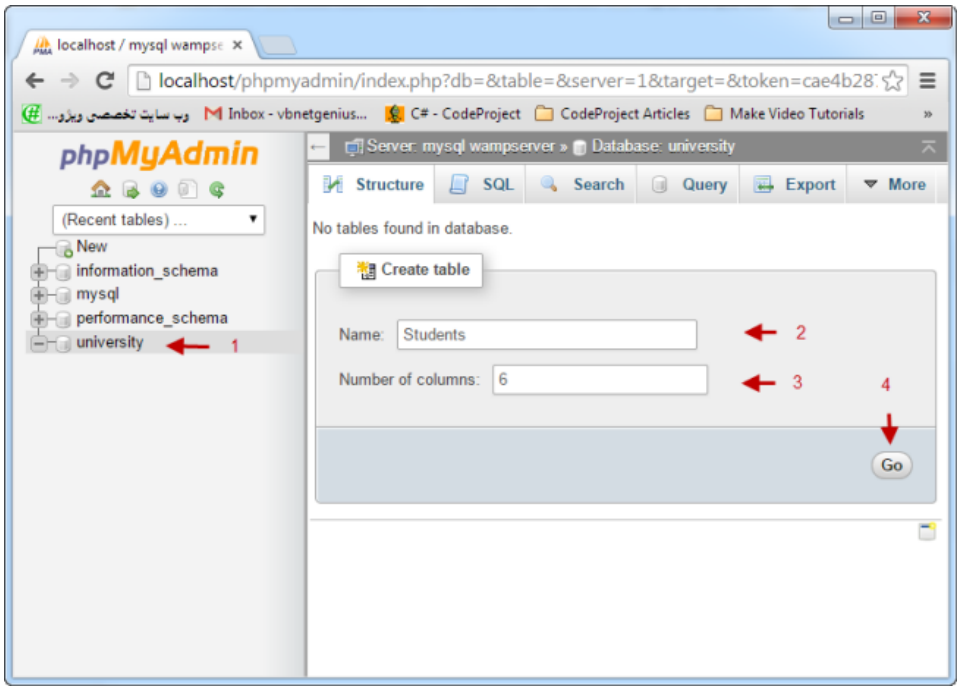
همانطور که در درس قبل مشاهده کردید ما یک دیتابیس با نام **University**که دارای جدولی با نام **Students** بود را با کدنویسی ایجاد کردیم .حال همین کار را می خواهیم با استفاده از محیط گرافیکی **phpMyAdmin** انجام دهیم برای این کار بر روی آیکون سبز رنگ **Wamp** کلیک کرده و با باز شدن پنجره بر روی گزینه **phpMyadmin** کلیک کنید:



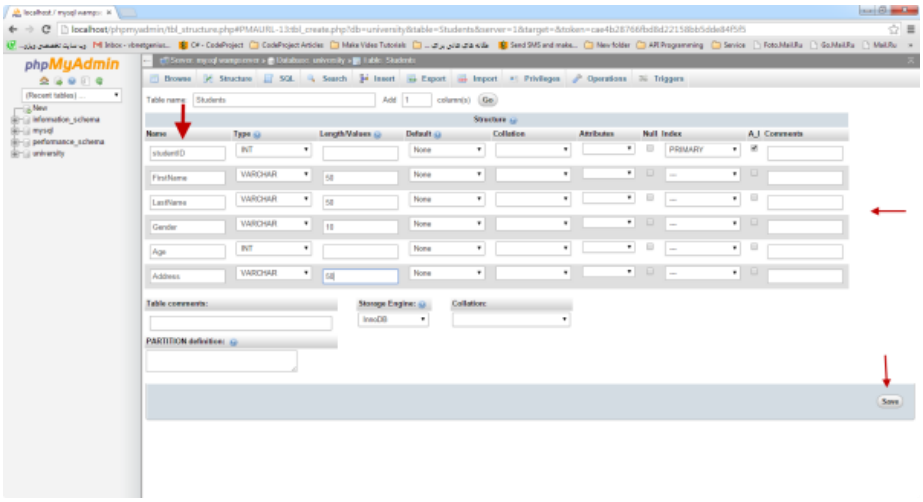
برای ایجاد دیتابیس **University** به صورت زیر عمل کنید:



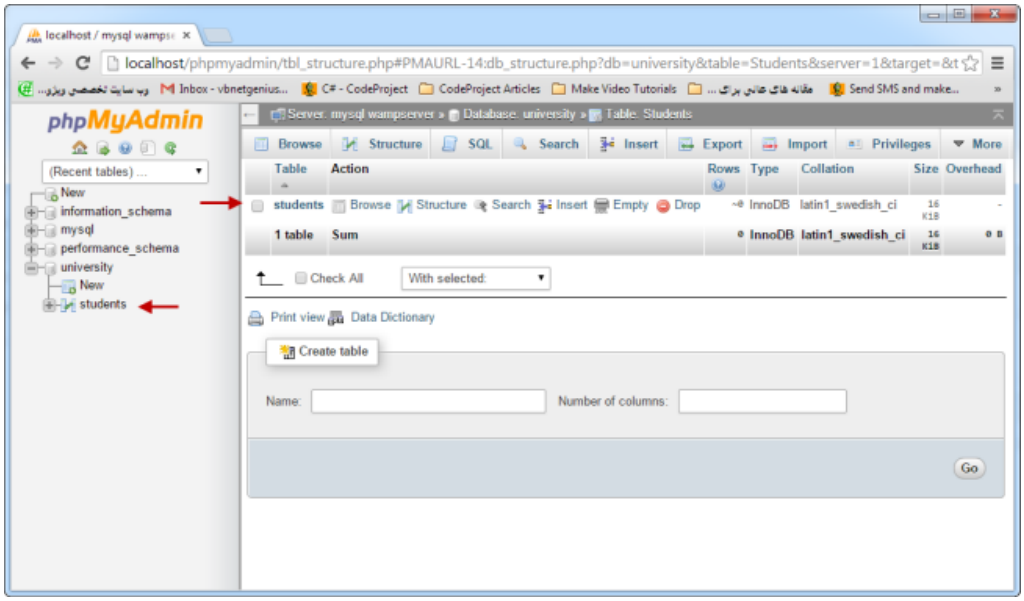
بعد از ایجاد دیتابیس بر روی آن کلیک کنید و سپس جدولی با نام Students با 6 ستون ایجاد نمایید:



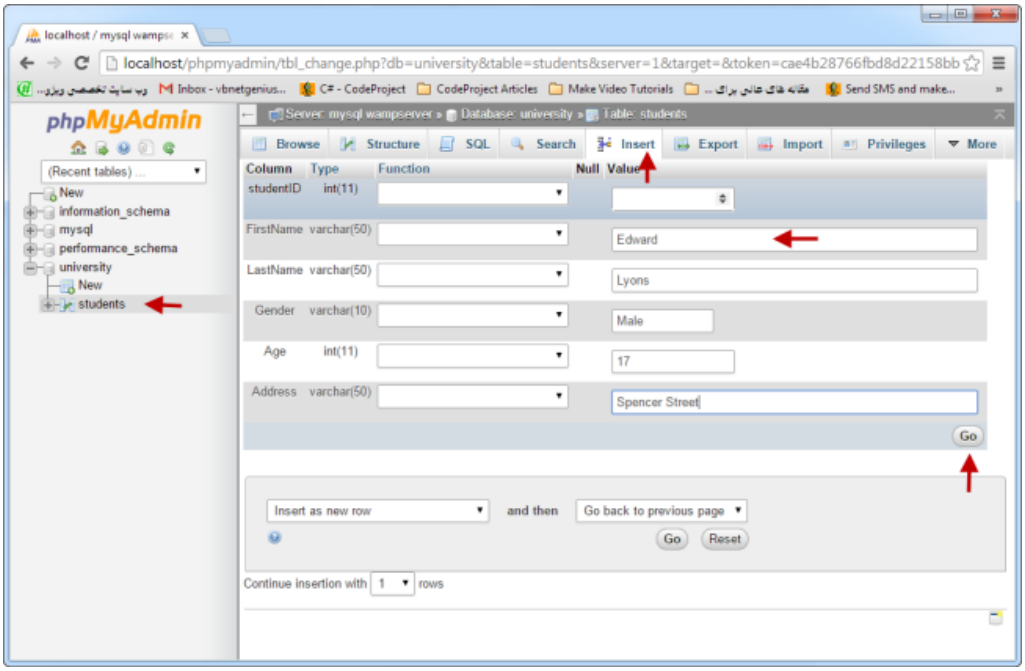
بعد از زدن دکمه GO وارد صفحه ای می شوید که از شما می خواهد عناوین ستون های جدول Students را وارد کنیدف این کار را به صورت زیر انجام دهید:



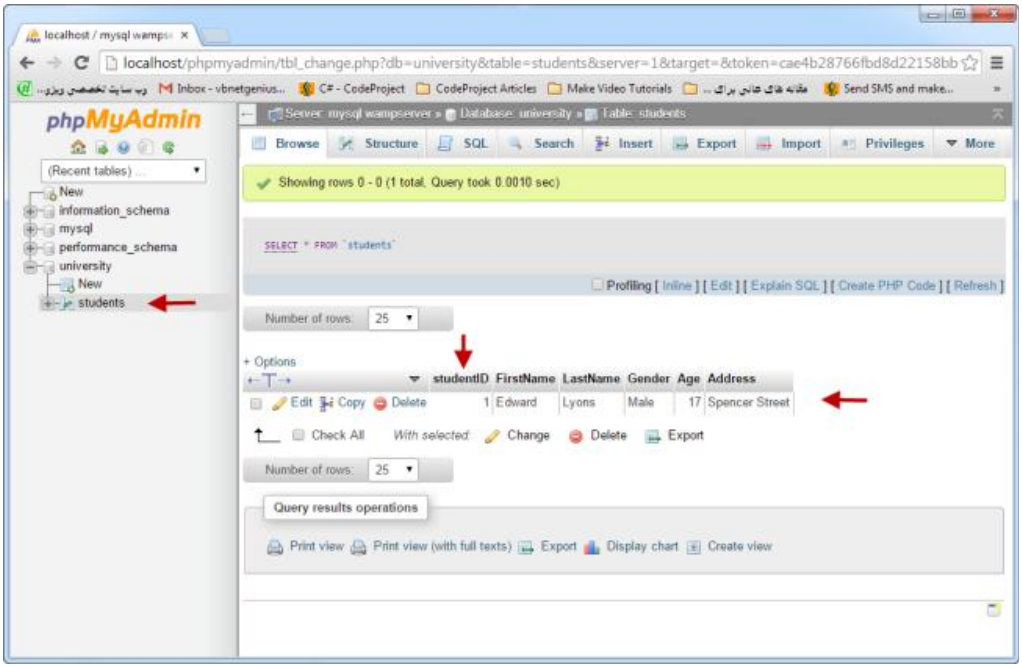
با زدن دکمه GO در شکل بالا جدول Students ایجاد می شود:



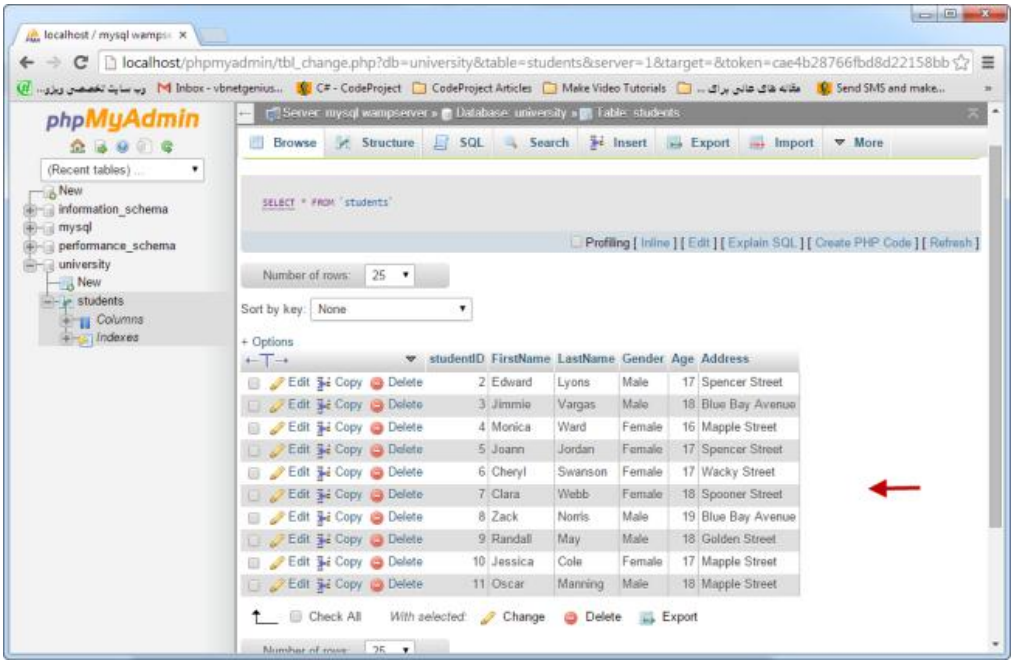
بعد از ایجاد جدول بر روی آن کلیک کرده و با رفتن به سربرگ Insert اطلاعات افراد را یک به یک وارد دیتابیس کنید:



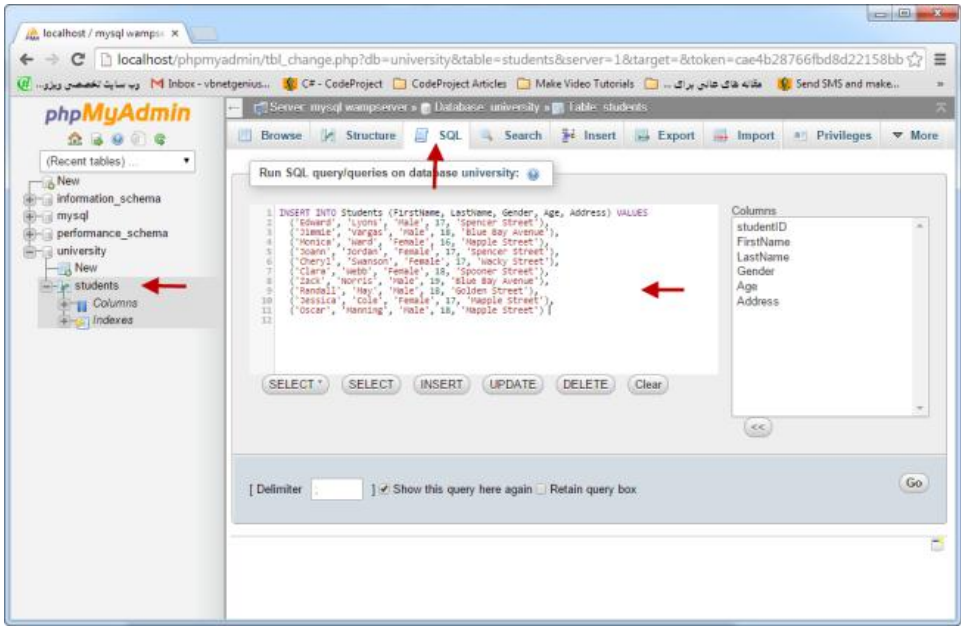
بعد از انجام عملیات فوق بر روی جدول کلیک کرده و مشاهده می کنید که اطلاعات یک نفر در جدول به صورت زیر وارد می شود:



مرحله قبل (رفتن به سربرگ Insert) را تا وارد کردن تمامی اطلاعات تکرار کنید تا در نهایت همه اطلاعات وارد جدول شوند:



روش دیگری هم برای وارد کردن اطلاعات یعنی کدنویسی در محیط گرافیکی phpMyAdmin وجود دارد. بدین صورت که شما می توانید با کلیک بر روی نام جدول و رفتن به سربرگ SQL کدها را نوشته و بر روی دکمه GO کلیک کنید تا اطلاعات وارد شوند:



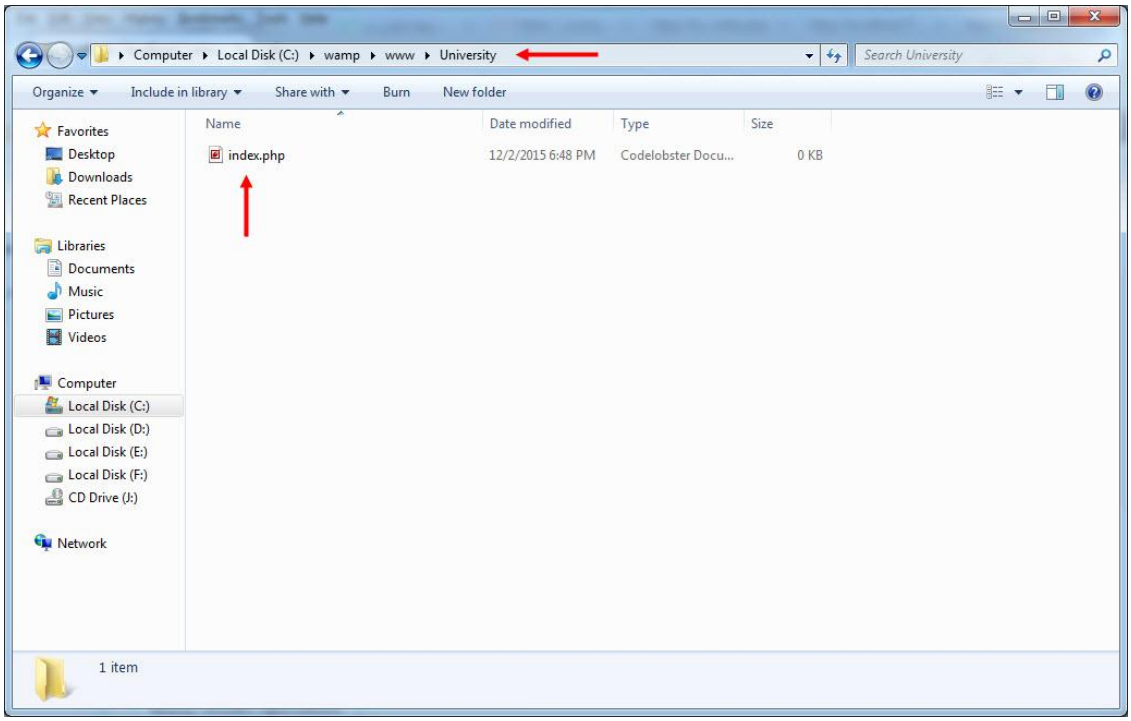
ارتباط با سرور و بانک اطلاعاتی (MySQL)

در درسهای قبلی با MYSQL و دستورات آن آشنا شدید و چگونگی ایجاد جدول، درج و حذف و ویرایش اطلاعات را با استفاده از دستورات MYSQL یاد گرفتید. از این درس به بعد می خواهیم شما را با روش اعمال تغییرات و دستکاری اطلاعات پایگاه داده با استفاده از دستورات PHP به سه روش MySQli ، MySQLو PDO آشنا کنیم. ابتدا MySQL و متدهای پر کاربرد آن را مورد بررسی قرار می دهیم:

تابع	کاربرد
mysql_affected_rows	تعداد رکورد های تحت تاثیر قرار گرفته در آخرین پرس و جوی انجام داده را می گیرد
mysql_change_user	کاربر فعال را در ارتباط جاری با بانک اطلاعاتی را عوض میکند عوض می کند
mysql_client_encoding	تنظیمات کاراکتری پیش فرض را از ارتباط جاری را باز میگرداند
mysql_close	ارتباع جاری با پایگاه داده را قطع میکند
mysql_connect	یک ارتباط جدید با پایگاه داده برقرار می کند
mysql_create_db	یک بانک اطلاعاتی میسازد
mysql_data_seek	نشانگر (پیوینتر) داخلی مجموعه جواب را حرکت می دهد
mysql_db_name	اطلاعات جواب پرس و جو را می گیرد
mysql_db_query	پرس و جو را ارسال می کند
mysql_drop_db	بانک اطلاعاتی را حذف میکند
mysql_errno	شماره خطای ایجاد شده در آخرین عملیات را باز می گرداند
mysql_error	متن خطای ایجاد شده در آخرین عملیات را باز میگرداند
mysql_escape_string	یک رشته را برای کار با بانک اطلاعاتی تطبیق می کند
mysql_fetch_array	یک مجموعه جواب بازگشتی را در آرایه ای انجمنی یا اندیسی یا هر دو می ریزد
mysql_fetch_assoc	یک مجموعه جواب بازگشتی را در آرایه ای انجمنی می ریزد
mysql_fetch_field	اطلاعات یک ستون را از یک مجموعه جواب میگیرد و یک شی را باز میگرداند
mysql_fetch_lengths	طول هر کدام از جواب های خروجی را باز می گرداند
mysql_fetch_object	یک مجموعه جواب را در یک شی می ریزد
mysql_fetch_row	یک مجموعه جواب را به صورت یک آرایه شمارشی در می آورد
mysql_field_flags	نمایه فیلد معرفی شده در مجموعه جواب را باز می گرداند
mysql_field_len	طول فیلد مشخص شده را باز میگرداند

mysql_field_name	نام فیلد مشخص شده در مجموعه جواب را باز می گرداند
mysql_field_seek	نشانه گر جواب را در فیلد مبدا مشخص شده قرار می دهد
mysql_field_table	نام جدولی را که فیلد در آن قرار دارد را باز می گرداند
mysql_field_type	نوع فیلد موجود در مجموعه جواب را باز می گرداند
mysql_free_result	حافظه را از مجموعه جواب خالی می کند
mysql_get_client_info	اطلاعات خدمات گیرنده پایگاه داده را بدست می آورد
mysql_get_host_info	اطلاعات میزبان پایگاه داده را بدست می آورد
mysql_get_proto_info	پروتکل(قوانین) مورد استفاده در پایگاه داده را میگیرد نسخه
mysql_get_server_info	اطلاعات خدمات دهنده پایگاه داده را بدست می آورد
mysql_info	اطلاعاتی در مورد جدیدترین پرسش و پاسخ را باز می گرداند
mysql_insert_id	شناسه آخرین فیلد اضافه شده را باز می گرداند
mysql_list_dbs	بانک های اطلاعاتی موجود را نشان می دهد
mysql_list_fields	ستون های جدول را نشان می دهد
mysql_list_processes	پردازش های انجام شده را نشان می دهد
mysql_list_tables	لیست جدول های یک بانک اطلاعاتی را نشان می دهد
mysql_num_fields	تعداد فیلد های یک مجموعه جواب را باز می گرداند
mysql_num_rows	تعداد سطرهای یک مجموعه جواب را باز می گرداند
mysql_pconnect	یک اتصال دائمی با پایگاه داده بر قرار می کند
mysql_ping	یک ارتباط را پینگ می کند و در صورت برقرار نبودن ارتباط آنرا برقرار میسازد
mysql_query	یک پرس و جو را ارسال میکند
mysql_real_escape_string	کاراکتر های ویژه را در یک رشته برای استفاده از پایگاه داده بهینه میکند و در این عمل از تنظیمات کاراکتری موجود در پایگاه داده استفاده می کند.
mysql_result	اطلاعات مجموعه جواب را باز میگرداند
mysql_select_db	یک بانک اطلاعاتی را نتخاب می کند
mysql_stat	وضعیت سیستم جاری را باز می گرداند
mysql_tablename	نام جدول فیلد را باز میگرداند
mysql_thread_id	جاری را باز می گرداند thread شناسه
mysql_unbuffered_query	یک پرس و جو را بدون واسطه به پایگاه داده ارسال می کند

قبل از شروع یک پوشه با نام University در داخل پوشه www ایجاد کرده و یک فایل با نام index.php در داخل آن قرار دهید:



ارتباط با بانک و جدول

همانطور که می دانید برای انجام هر گونه تغییر بر روی پایگاه داده ابتدا باید با کامپیوتری که پایگاه داده بر روی آن قرار دارد، ارتباط برقرار کنیم. برای برقراری این ارتباط از تابع mysql_connect() به صورت زیر استفاده می شود:

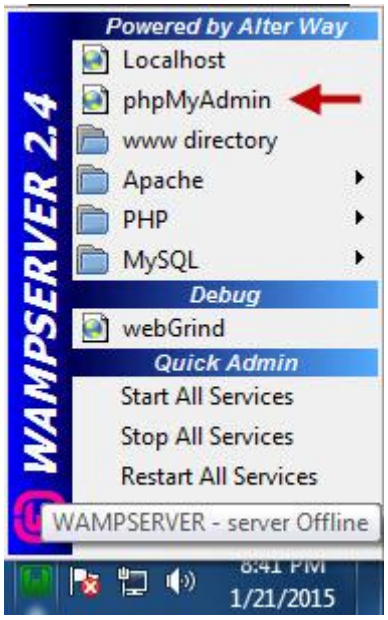
```
mysql_connect(host,username,password,datebasename);
```

پارامترهایی که این تابع قبول می کند به شرح زیر می باشند:

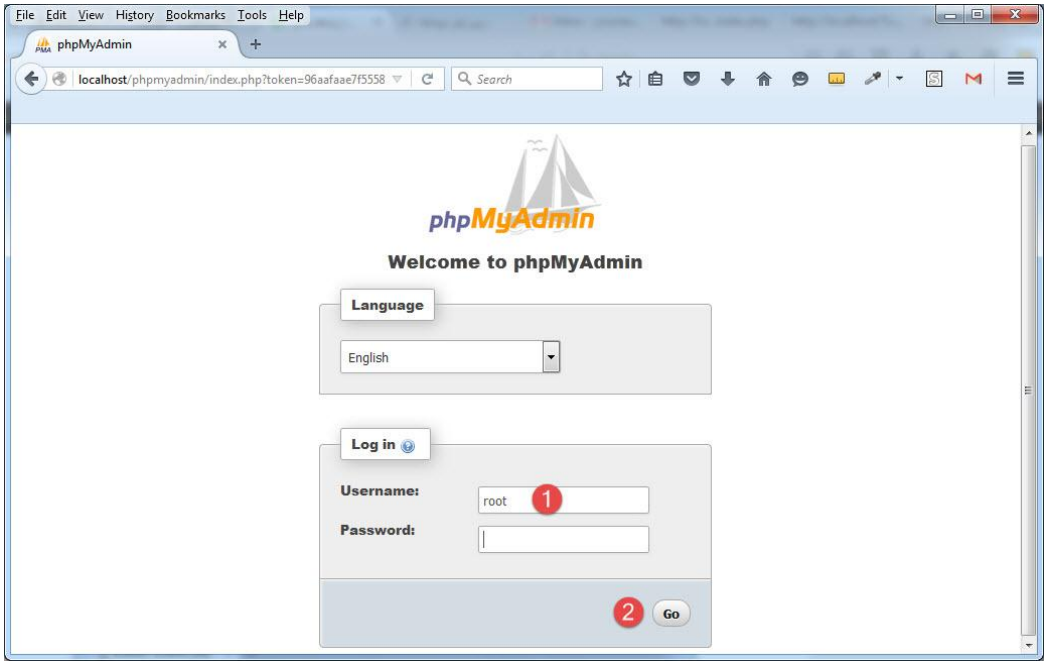
پارامتر	توضیحات
host	نام یا IP کامپیوتری که MYSQL بر روی آن نصب است.
username	نام کاربری که برای اتصال به پایگاه داده می خواهید از آن استفاده کنید.
password	رمز ورودی که برای نام کاربری انتخاب کرده اید.

databasename	نام پایگاه داده ای که می خواهید مورد استفاده قرار دهید.

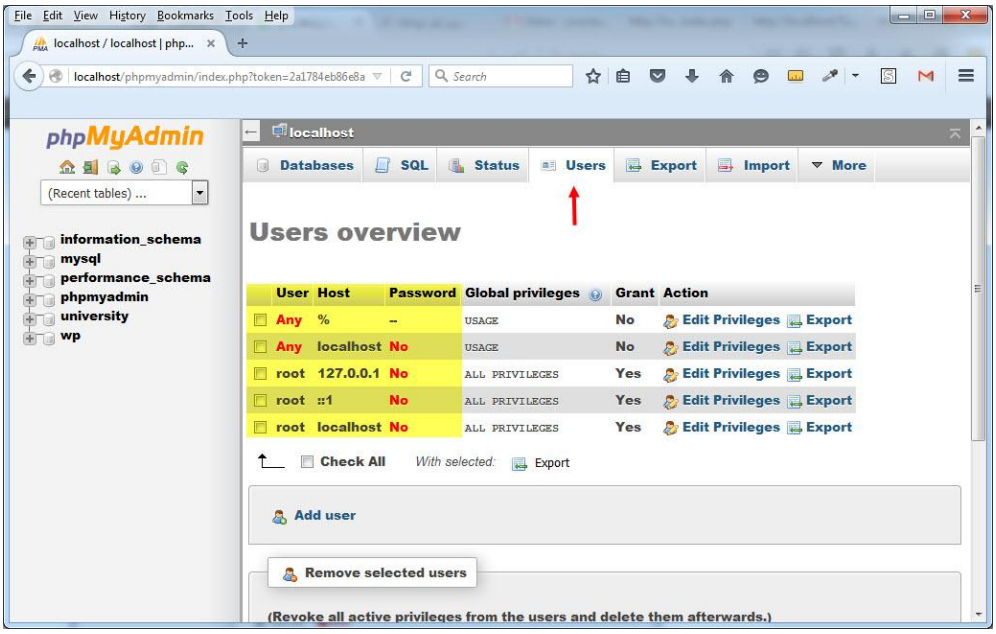
اما از کجا این پارامترها را به دست بیاورید. همانطور که در درس های اول مشاهده کردید ما نرم افزار wampserver را که شامل MYSQL هم می شود بر روی کامپیوتر نصب کردیم. پس سرور ما همان کامپیوتر ما یا localhost می باشد .برای دسترسی به نام کاربری و پسورد هم بر روی آیکون سبز رنگ wampserver کنار ساعت سیستم و سپس بر روی گزینه phpMyAdminکلیک کنید:



با کیلک بر روی این گزینه صفحه ای به صورت زیر ظاهر می شود که از شما usernameو password ی را می خواهد که بتوانید با آن وارد MYSQL شده و از پایگاه داده استفاده کنید. اگر مراحل نصب wampserver را به صورت پیشفرض طی کنید نام کاربری شما root و پسوردتان خالی خواهد بود. پس مقدار root را مانند شکل زیر وارد کرده و سپس بر روی دکمه GOکلیک کنید:



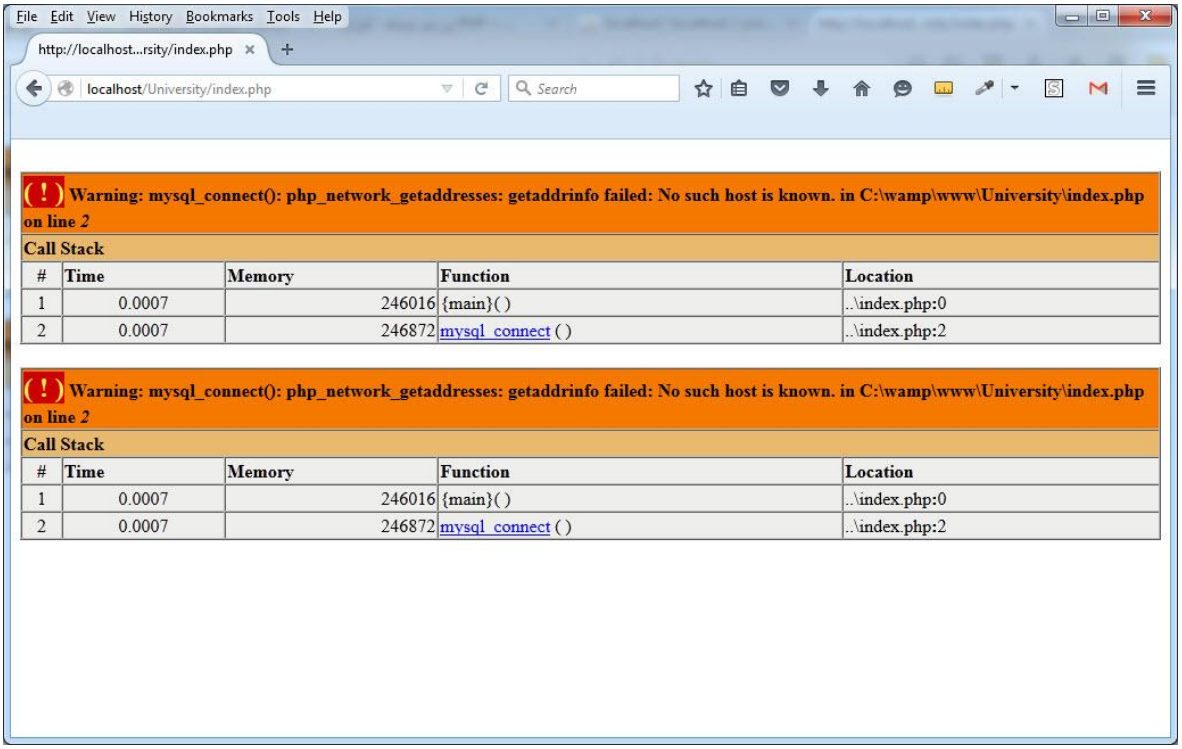
پس از ورود به محیط phpMyAdmin به سربرگ Users بروید تا لیستی از کدکاربری و پسوردهای مجازی که با آنها می توانید وارد MYSQL شوید مشاهده کنید:



علاوه بر username و password های بالا می توانید با استفاده از دکمه Add User که در شکل بالا مشاهده می کنید، نام کاربری و کلمه عبور دلخواهی برای ورود به MYSQL ایجاد کنید.به ادامه آموزش می پردازیم. فایل index.php را با یک ویرایشگر متن باز کرده و کد زیر را در آن بنویسید:

```
<?php
    mysql_connect('localhost','root','');
?>
```

با اجرای کد بالا به راحتی به سرور وصل می شویم. حال اگر خطایی اعم از اشتباه تایپی و ... در کد بالا وجود داشته باشد، پیغام خطایی مبنی بر عدم اتصال شما به سرور به شما نشان داده می شود:



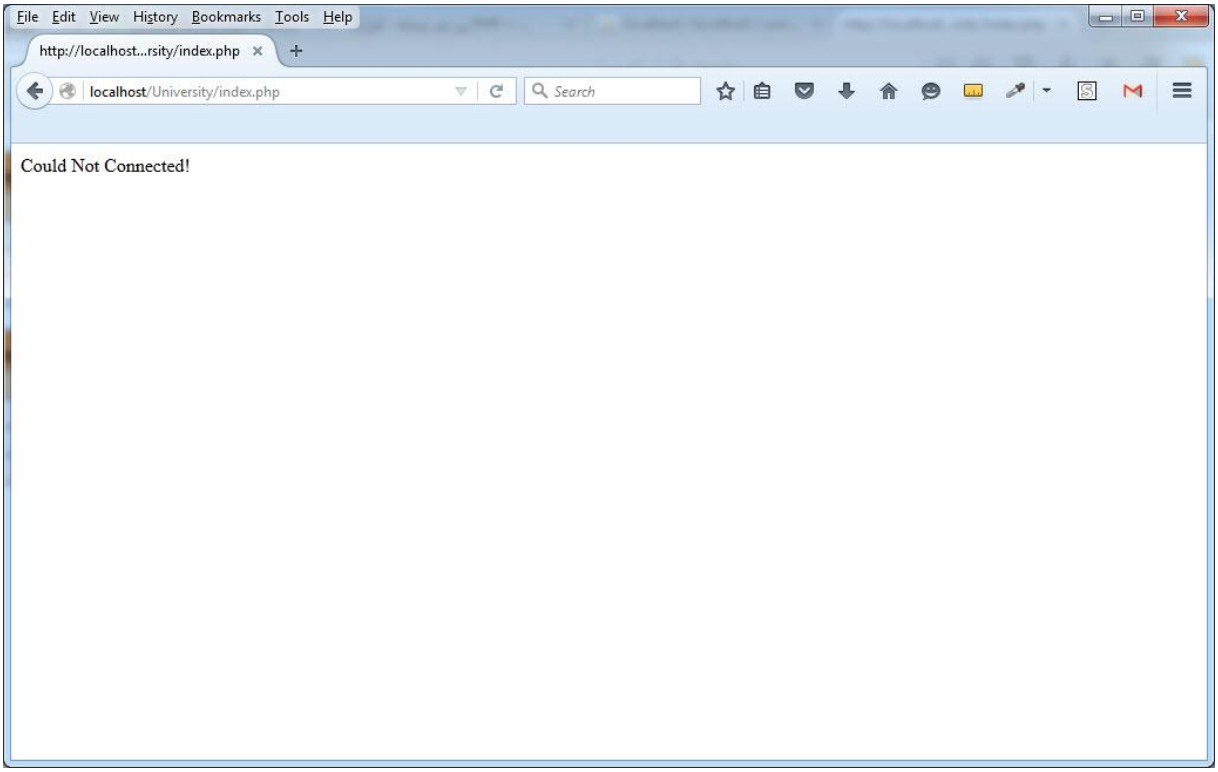
برای جلوگیری از نمایش این خطا کافیست که یک علامت @ قبل از شروع دستور به صورت زیر بگذارید:

```
<?php
    @mysql_connect('localhost','root','');
?>
```

و برای تکمیل فرایند یک پیغام خطا نیز به کاربر با استفاده از عبارت or die به شکل زیر به کاربر نشان دهید:

```
<?php
    @mysql_connect('localhost','root','') or die ('Could Not Connected!');
?>
```

این دستور باعث می شود که اگر به هر نحوی ارتباط با سرور برقرار نشد پیغامی به کاربر نمایش داده شود، برای روشن شدن مطلب کلمه **root** را به **roo** تغییر داده و کد را اجرا کنید:



حال که با نحوه اتصال به سرور آشنا شدید، می خواهیم شما را با نحوه اتصال به دیتابیس و انتخاب آن آشنا کنیم. برای انتخاب دیتابیس از تابع **mysql_select_db()**به صورت زیر استفاده می شود:

```
mysql_select_db(DatabaseName);
```

همانگونه که در کد بالا مشاهده می کنید برای انتخاب یک دیتابیس کافیست که نام آن را به تابع **mysql_select_db()** بدهیم. فرض کنید که می خواهیم با دیتابیس **University** که در درس قبل ایجاد کردیم ارتباط برقرار کرده و آن را انتخاب کنیم. برای این کار کد **index.php** را به صورت زیر تغییر داده ، ذخیره و اجرا کنید:

```
<?php
   @mysql_connect('localhost','root','') or die ('Could Not Connected!');
   @mysql_select_db('University') or die ('Could Not Connected!');
?>
```

با اجرای کدهای بالا اگر همه چیزی به خوبی پیش رود هیچ چیز بر روی مرورگر نمایش داده نمی شود ولی اگر خطایی رخ دهد و یا در ارتباط با سرور و یا بانک اشکالی به وجود آید پیغام خطایی به کاربر نمایش داده می شود. البته کد بالا را به صورت زیر بهینه تری نیز می توان نوشت:

```
<?php
    if(mysql_connect('localhost','root',''))
    {
        if(mysql_select_db('University'))
        {
            echo 'Connect!';
        }
    }
?>
```

کد بالا را به صورت خلاصه تر زیر هم می توان نوشت:

```
<?php
    if(!@mysql_connect('localhst','root','') ||
!@mysql_select_db('University'))
    {
        echo 'Could Not Connect!';
    }
    else
    {
        echo 'Connect!';
    }
?>
```

هر دو کد آخر ارتباط با بانک و سرور را چک می کنند و در صورت اختلال در هر کدام، پیغام خطای عدم برقراری ارتباط به کاربر نمایش داده می شود. در این درس با نحوه برقراری ارتباط با سرور و بانک آشنا شدید. در درس های آینده شما را با روش های درج و حذف و ... اطلاعات در بانک آشنا می کنیم.

اضافه کردن رکورد به بانک (MySQL)

در درس قبل شما با نحوه ارتباط با سرور و بانک اطلاعاتی آشنا شدید. همچنین یک جدول با سه ستون ایجاد کردیم که اطلاعات مربوط به مثلا دانشجوها را در خود جای می دهد. در این درس می خواهیم شما را با نحوه اضافه کردن اطلاعات به این جدول از بانک اطلاعاتی آشنا کنیم.ابتدا کدهای فایل **index.php** که در درس قبل ایجاد کردیم را پاک کرده و کدهای زیر را در داخل آن بنویسید:

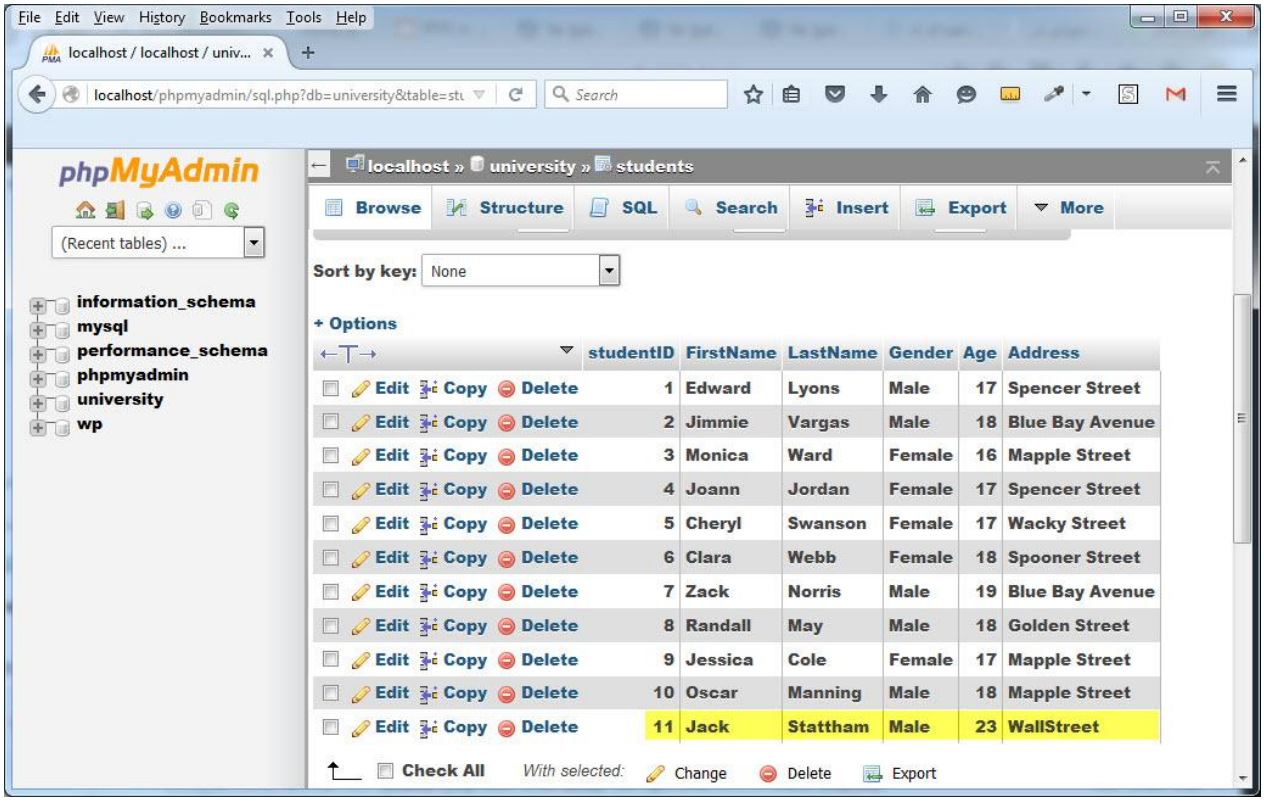
```
1  <?php
2       $ServerConnect   = @mysql_connect('localhost','root','');
3       $DatabaseConnect = @mysql_select_db('student');
4       mysql_query('SET NAMES \'utf8\'');
5
6       if($ServerConnect && $DatabaseConnect)
7       {
8
9            ...
10
11      }
12      else
13      {
14         echo'No Connect!';
15      }
16 ?>
```

همانطور که می دانید برای انجام هر گونه عملیات بر روی بانک باید با آن ارتباط برقرار کرد. در کد بالا ما هم همین کار را انجام داده ایم. دستور **if** (خط **6**) چک می کند که آیا ارتباط با بانک و دیتابیس برقرار است یا نه؟ به این نکته توجه کنید که برای راحتی کار دو ارتباط را در خطوط **2** و **3** در دو متغیر قرار داده ایم.اگر ارتباط برقرار بود که وارد بدنه شرط می شویم در غیر اینصورت در خط **14** و در قسمت **else** به کاربر پیغامی مبنی بر عدم اتصال نمایش می دهیم. برای درج کلمات فارسی در بانک حتما دستور موجود در خط **4** کد بالا و تگ **meta** در خط **5** کد ابتدای درس را بنویسید. فرض کنیم اتصال برقرار است. حال در خط **9** کد بالا و در داخل بدنه **if** کدهای زیر را وارد کنید:

```
1  <?php
2      $ServerConnect   = @mysql_connect('localhost','root','');
3      $DatabaseConnect = @mysql_select_db('University');
4      mysql_query('SET NAMES \'utf8\'');
5
6      if($ServerConnect && $DatabaseConnect)
7      {
8      mysql_query("
9                   INSERT INTO students (FirstName, LastName, Gender, Age,
10 Address)
11                  VALUES
12                  ('Jack', 'Stattham', 'Male', 23, 'WallStreet')
```

```
13              ");
14      }
15      else
16      {
17          echo 'No Connect!';
18      }
    ?>
```

در PHP تابعی به نام mysql_query() وجود دارد که دستورات MySQL ما را بر روی بانک اعمال می کند. دستور INSERT INTOمقادیر را در ردیف های تعیین شده ذخیره می کند. در کد بالا بعد از نام جدول students و داخل پرانتز نام ستون های آن را نوشته ایم. اگر عملیات درج با موفقیت انجام شود تابع mysql_query() عدد 1 را بر می گرداند که به معنای انجام موفقیت آمیز عملیات است. یک نکته در مورد کد بالا یاد آور می شویم. اول اینکه به این دلیل با جعبه متن ID کاری نداریم،چون آن را به صورت AUTO_INCREMENT تعریف کرده ایم و با صورت خودکار و با اضافه شدن هر رکورد یک واحد به آن اضافه می شود و با آن در درس های بعدی کار داریم. حال کد را اجرا به بانک نگاه کنید، مشاهده می کنید که اطلاعات وارد شده اند:



در درس بعد در مورد چگونگی انتخاب یا خواندن رکورد از بانک توضیح می دهیم.

انتخاب یا خواندن رکورد از بانک (MySQL)

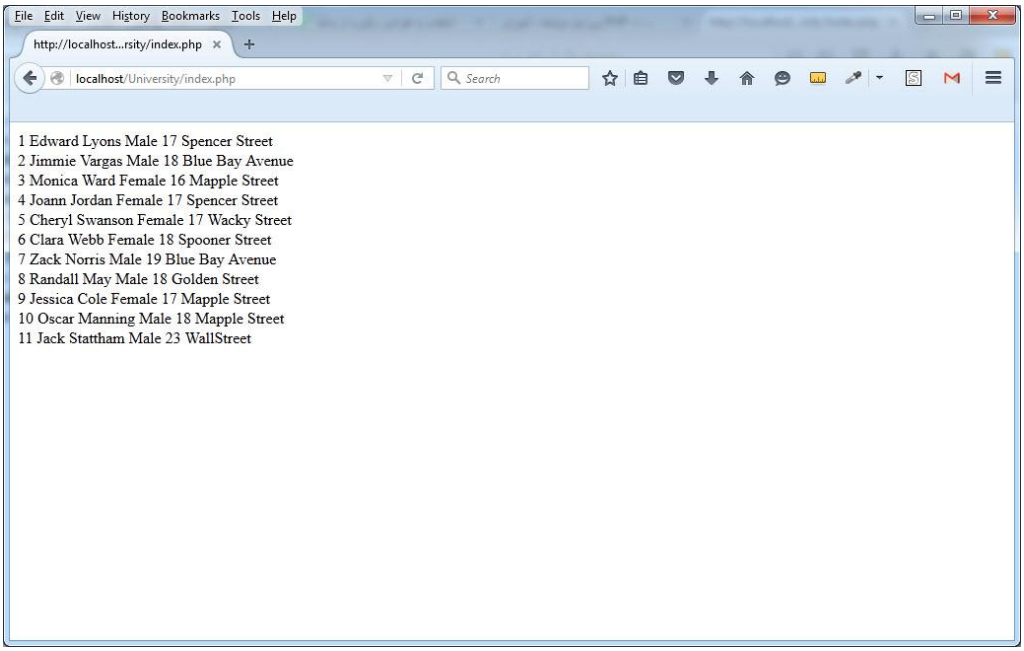
در درس قبل با چگونگی اضافه کردن اطلاعات به بانک **MYSQL** با استفاده از دستورات **PHP** آشنا شدید. در این درس می خواهیم عملیان خواندن اطلاعات از بانک را برای شما توضیح دهیم. این عمل با روش های مختلفی انجام می شود که در این درس ساده ترین نوع را به شما آموزش می دهیم. ابتدا و قبل از هر کاری می خواهیم تمام اطلاعات موجود در بانک را در مرورگر نمایش دهیم .برای این کار از دستور **SELECT** به صورت زیر استفاده می کنیم:

```
SElECT * FROM Table
```

معنای دستور بالا این است که همه اطلاعات را از فلان بانک (بانک مورد نظر ما) انتخاب کن. برای نمایش اطلاعات بانک کد زیر را بعد از دستور درج می نویسیم:

```
1  $select = mysql_query("SElECT * FROM  students");
2
3  if($select && mysql_num_rows($select) > 0)
4  {
5      while($rows = mysql_fetch_assoc($select))
6      {
7          echo $rows['studentID'] .' '. $rows['FirstName'].'
8  '.$rows['LastName'].' '.$rows['Gender'].' '. $rows['Age'].'
9  '.$rows['Address'].'<br/>';
10     }
11 }
```

همانطور که در کد بالا مشاهده می کنید ابتدا در خط **1** با استفاده از تابع **mysql_query()** اطلاعات موجود در جدول **students**را می گیریم و در داخل یک متغیر **(select$)** می ریزیم. در خط **3** با استفاده از دستور **if** چک می کنیم که اگر عمل انتخاب انجام شده باشد و تعداد رکوردهای انتخاب شده بیشتر از **0** با شد یعنی رکوردی وجود داشته باشد دستورات بعدی اجرا شوند (پس متد **mysql_num_rows()** تعداد رکوردهای موجود در بانک را بر می گرداند). در خط **4** و با استفاده از متد **mysql_fetch_assoc()** رکوردهای انتخاب شده را به آرایه ای انجمنی تبدیل می کنیم. و در خط **7** از عنوان ستون های موجود در بانک به عنوان کلیدهای این آرایه استفاده می کنیم. کد را ذخیره و مرورگر را **Refresh** کنید:



از آنجاییکه من از قبل رکوردهایی به بانک اضافه کرده ام جدولم به صورت بالا نمایش داده می شود .

ویرایش رکوردهای بانک (MySQL)

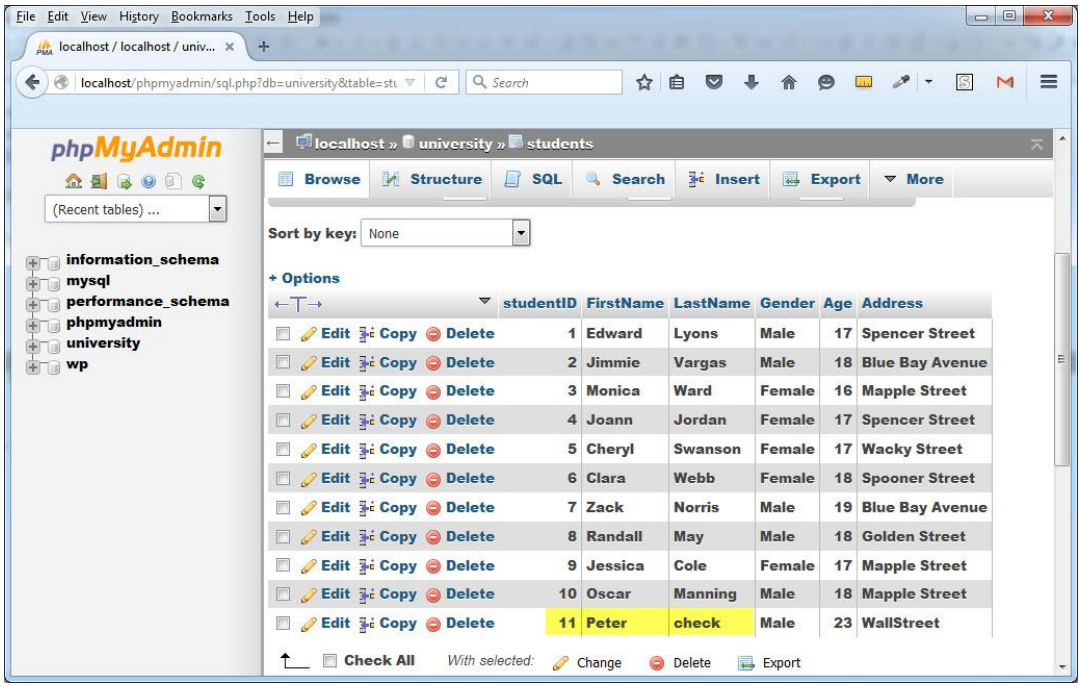
در درس قبل با چگونگی انتخاب رکورد از بانک اطلاعاتی آشنا شدید. در این درس می خواهیم شما را با نحوه ویرایش رکوردها آشنا کنیم. برای ویرایش رکورد از دستور **UPDATE** به صورت زیر استفاده می شود:

```
UPDATE <Table Name> SET <Column> = <Value> WHERE <Condition>;
```

در دستور بالا **Table Name** نام جدول، **Column**نام ستون یا ستون هایی که قرار است ویرایش شوند و **Condition** شرط ویرایش اطلاعات است. در پایین دستور خواندن در درس قبل، کد زیر را بنویسید:

```
mysql_query("UPDATE students SET FirstName = 'Peter', LastName = 'check'
WHERE studentID= 11");
```

با دستور **UPDATE** در درس های قبلی آشنا شدید. در کد بالا ما نام و نام خانوادگی شخصی که **studentID** آن برابر **11** است را به **peter check** تغییر داده ایم. با اجرای برنامه تغییر اعمال می شود و شما می توانید این تغییر را در بانک مشاهده کنید:

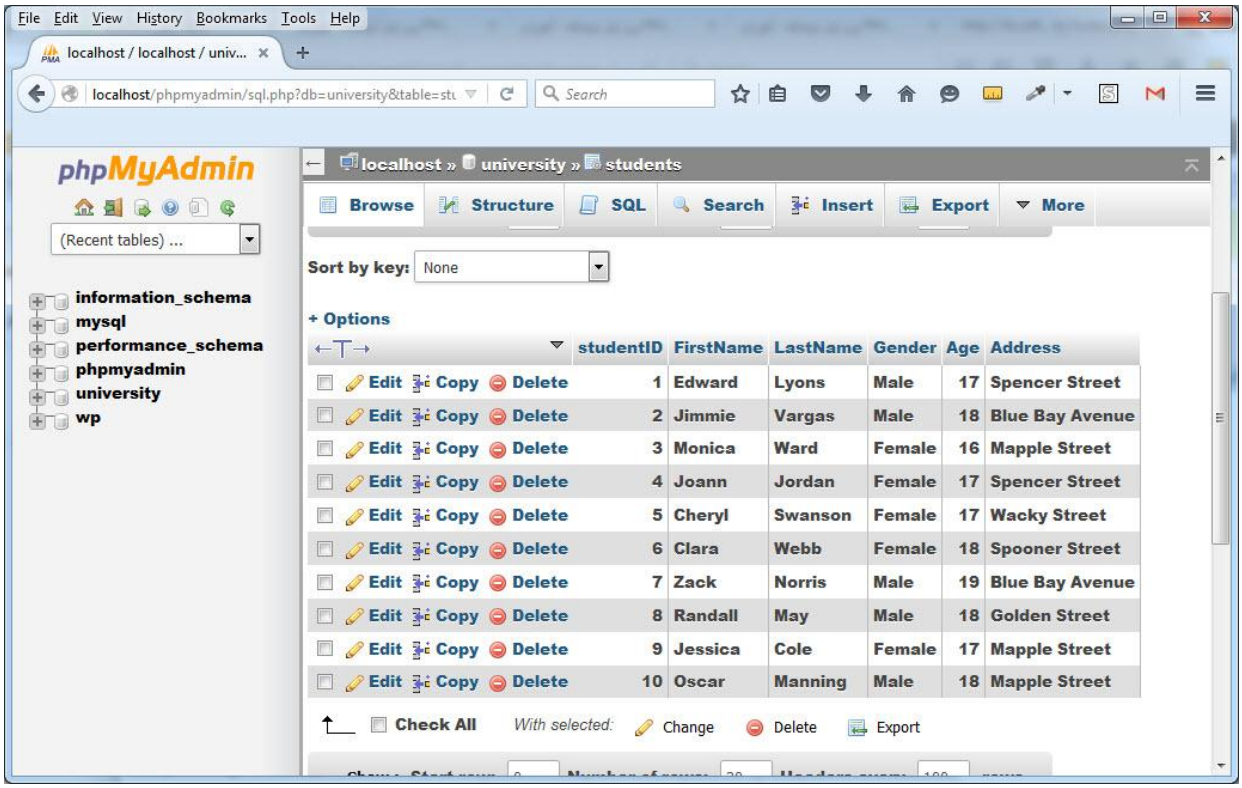


حذف رکورد از بانک (MySQL)

تا کنون با نحوه ذخیره، انتخاب و ویرایش رکوردها آشنا شدید. برای حذف رکورد از بانک اطلاعاتی از دستور زیر استفاده می شود:

```
DELETE FROM <Table Name> WHERE <Condition>;
```

در کد بالا **Table Name** نام جدول و **Condition** هم شرط حذف رکورد می باشد .فرض کنید که می خواهیم رکورد آخر بانک که **studentID** آن برابر **11** است را حذف کنیم. برای این کار کد زیر را می نویسیم:

```
mysql_query("DELETE FROM students WHERE studentID= 11");
```

حال اگر کد بالا را اجرا کنید، مشاهده می کنید که رکورد حذف شده است: