Desain dan Pemrograman Web

12. PHP:Time, Include, Require, Form.

M. Udin Harun Al Rasyid, S.Kom, Ph.D
http://lecturer.eepis-its.edu/~udinharun
udinharun@eepis-its.edu

Table of Contents

PHP Date and Time

- Dates are so much part of everyday life that it becomes easy to work with them without thinking.
- PHP also provides powerful tools for date arithmetic that make manipulating dates easy.

Getting the Time Stamp with time():

- PHP's **time()** function gives you all the information that you need about the current date and time. It requires no arguments but returns an integer.

- The integer returned by **time() represents the number of seconds elapsed since midnight GMT on January 1, 1970**. This moment is known as the UNIX epoch, and the number of seconds that have elapsed since then is referred to as a time stamp.

- This is something difficult to understand. But PHP offers excellent tools to convert a time stamp into a form that humans are comfortable with.

```
<?php
print time();
?>
```

```
1354513904
```

Converting a Time Stamp with getdate()

- The function **getdate()** optionally accepts a time stamp and returns an associative array containing information about the date.

- If you omit the time stamp, it works with the current time stamp as returned by time().

- Following table lists the elements contained in the array returned by getdate().

Key	Description	Example
seconds	Seconds past the minutes (0-59)	20
minutes	Minutes past the hour (0 - 59)	29
hours	Hours of the day (0 - 23)	22
mday	Day of the month (1 - 31)	11
wday	Day of the week (0 - 6)	4
mon	Month of the year (1 - 12)	7
year	Year (4 digits)	1997
yday	Day of year (0 - 365)	19
weekday	Day of the week	Thursday
month	Month of the year	January
0	Timestamp	948370048

Example

```php
<?php
$date_array = getdate();
foreach ( $date_array as $key => $val )
{
   print "$key = $val<br />";
}
$formated_date  = "Today's date: ";
$formated_date .= $date_array[mday] . "/";
$formated_date .= $date_array[mon] . "/";
$formated_date .= $date_array[year];

print $formated_date;
?>
```

seconds = 3
minutes = 58
hours = 6
mday = 3
wday = 1
mon = 12
year = 2012
yday = 337
weekday = Monday
month = December
0 = 1354514283
Today's date: 3/12/2012

Converting a Time Stamp with date():

- The **date()** function returns a formatted string representing a date.
- You can exercise an enormous amount of control over the format that date() returns with a string argument that you must pass to it.
- Syntax: date(format,timestamp)

- Following table lists the codes that a format string can contain:

Format	Description	Example
a	'am' or 'pm' lowercase	pm
A	'AM' or 'PM' uppercase	PM
d	Day of month, a number with leading zeroes	20
D	Day of week (three letters)	Thu
F	Month name	January
h	Hour (12-hour format - leading zeroes)	12
H	Hour (24-hour format - leading zeroes)	22
g	Hour (12-hour format - no leading zeroes)	12
G	Hour (24-hour format - no leading zeroes)	22
i	Minutes (0 - 59)	23
j	Day of the month (no leading zeroes	20
l (Lower 'L')	Day of the week	Thursday
L	Leap year ('1' for yes, '0' for no)	1
m	Month of year (number - leading zeroes)	1
M	Month of year (three letters)	Jan
r	The RFC 2822 formatted date	Thu, 21 Dec 2000 16:01:07 +0200
n	Month of year (number - no leading zeroes)	2
s	Seconds of hour	20
U	Time stamp	948372444
y	Year (two digits)	06
Y	Year (four digits)	2006
z	Day of year (0 - 365)	206
Z	Offset in seconds from GMT	+5

```php
<?php
print date("m/d/y G.i:s <br>", time());
print "Today is ";
print date("j F Y", time());
?>
```

PHP include and require Statements

- In PHP, you can insert the content of one PHP file into another PHP file before the server executes it.

- The include and require statements are used to insert useful codes written in other files, in the flow of execution.

- **Include and require are identical, except upon failure:**
  - require will produce a fatal error (E_COMPILE_ERROR) and stop the script
  - include will only produce a warning (E_WARNING) and the script will continue

- So, if you want the execution to go on and show users the output, even if the include file is missing, **use include**.

- Otherwise, in case of FrameWork, CMS or a complex PHP application coding, always **use require** to include a key file to the flow of execution. This will help avoid compromising your application's security and integrity, just in-case one key file is accidentally missing.

- Including files saves a lot of work. This means that you can create a standard header, footer, or menu file for all your web pages. Then, when the header needs to be updated, you can only update the header include file.

- **Syntax**

```
include 'filename';

or

require 'filename';
```

PHP include and require Statement

Basic Example

- Assume that you have a standard header file, called "header.php". To include the header file in a page, use include/require:

```
<html>
<body>

<?php include 'header.php'; ?>
<h1>Welcome to my home page!</h1>
<p>Some text.</p>

</body>
</html>
```

Example 2

- Assume we have a standard menu file that should be used on all pages. "menu.php":

```
echo '<a href="/default.php">Home</a>
<a href="/tutorials.php">Tutorials</a>
<a href="/references.php">References</a>
<a href="/examples.php">Examples</a>
<a href="/about.php">About Us</a>
<a href="/contact.php">Contact Us</a>';
```

- All pages in the Web site should include this menu file. Here is how it can be done:

```
<html>
<body>

<div class="leftmenu">
<?php include 'menu.php'; ?>
</div>

<h1>Welcome to my home page.</h1>
<p>Some text.</p>

</body>
</html>
```

Example 3

- Assume we have an include file with some variables defined ("vars.php"):

```
<?php
$color='red';
$car='BMW';
?>
```

- Then the variables can be used in the calling file:

```
<html>
<body>

<h1>Welcome to my home page.</h1>
<?php include 'vars.php';
echo "I have a $color $car"; // I have a red BMW
?>

</body>
</html>
```

PHP form

- In HTML, a form is begins and ends with a <form> tag. As an example, let's start creating a form in a file named *myform.php*.

```
<form action="myform.php" method="post">
        <!-- form fields go here -->
</form>
```

- The **"action"** specifies what page to submit the form to. Many times, the action will be the same page as the form. The **"method"** indicates how the form is submitted. There are two methods: **"get"** and **"post"**.

- Now let's add some inputs to the form. Let's add a text field that asks for your **favorite movie** and a **submit button** to submit the form.

```
<form action="myform.php" method="post">
Which is your favorite movie?
<input type="text" name="formMovie" maxlength="50">
<input type="submit" name="formSubmit" value="Submit">
</form>
```

Getting the form data

- **The input of type "text"** is just a single line field to type in some text.
- We give it a name of "**formMovie**" which we will use later during processing. **Maxlength** just indicates that the browser shouldn't let the user type more than 50 characters into the text box.
- Let's add some PHP to process this form:

```
<?php
  if($_POST['formSubmit'] == "Submit")
  {
    $varMovie = $_POST['formMovie'];
  }
?>

<form action="myform.php" method="post">
Which is your favorite movie?
<input type="text" name="formMovie" maxlength="50">
<input type="submit" name="formSubmit" value="Submit">
</form>
```

- Let's add another input

```php
<?php
  if($_POST['formSubmit'] == "Submit")
  {
    $varMovie = $_POST['formMovie'];
    $varName = $_POST['formName'];
  }
?>

<form action="myform.php" method="post">
    Which is your favorite movie?
    <input type="text" name="formMovie" maxlength="50" value="
<?=$varMovie;?>">
    What is your name?
    <input type="text" name="formName" maxlength="50" value="
<?=$varName;?>">
    <input type="submit" name="formSubmit" value="Submit">
</form>
```

Finish