Subject: Web Technologies (R09) Syllabus:

Author: Bhavsingh Maloth

UNIT-V

Syllabus: PHP Advanced Concepts: Using Cookies, Using HTTP Headers, Using Sessions, Authenticating users, Using Environment and Configuration variables, Working with Date and Time.

PHP Advanced Concepts: Using Cookies:

Chapter-1

Introduction to Cookies

Cookies are text files stored on the client computer and they are kept of use tracking purpose. PHP transparently supports HTTP cookies.

There are three steps involved in identifying returning users:

- Server script sends a set of cookies to the browser. For example name, age, or identification number etc.
- Browser stores this information on local machine for future use.
- When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.

The Anatomy of a Cookie:

Cookies are usually set in an HTTP header:

Setting Cookies with PHP:

- PHP provided setcookie() function to set a cookie. This function requires upto six arguments and should be called before <html> tag. For each cookie this function has to be called separately.

```
setcookie(name, value, expire, path, domain, security);
```

Here is the detail of all the arguments:

1. Name - This sets the name of the cookie and is stored in an environment variable called HTTP_COOKIE_VARS. This variable is used while accessing cookies.
2. Value -This sets the value of the named variable and is the content that you actually want to store.
3. Expiry - This specify a future time in seconds since 00:00:00 GMT on 1st Jan 1970. After this time cookie will become inaccessible. If this parameter is not set then cookie will automatically expire when the Web Browser is closed.
4. Path -This specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.
5. Domain - This can be used to specify the domain name in very large domains and must contain at least two periods to be valid. All cookies are only valid for the host and domain which created them.
6. Security - This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which mean cookie can be sent by regular HTTP.

First Cookie

```php
<?php
$username = 'Harshith';

setcookie('username', $username);
?>
```

7. As this cookie does not have an expiry time, then the cookie will be deleted once the user closes their browser. This can be very useful for keeping a user logged in for an indefinite amount of time, however, as soon as they close their browser (hence, have left the site), the cookie is removed.
8. As this cookie does not have an expiry time, then the cookie will be deleted once the user closes their browser. This can be very useful for keeping a user logged in for an

indefinite amount of time, however, as soon as they close their browser (hence, have left the site), the cookie is removed.

9. There are two very important things to abide by when using cookies. Firstly, there are can be no HTML, text or white-space output before calling the setcookie() function. This is due to a 'protocol restriction' and you will find that header() and session_start() functions must also follow this rule.

Accessing the Cookie:

Access the cookie data

```
<?php

/* Prints the value stored in the username cookie */
echo $_COOKIE['username'];

?>
```

- This code uses the $_COOKIE superglobal to access any cookies set in the current domain. If we were to run this script, it would output *Harshith*
- You can use isset() function to check if a cookie is set or not.

```
<html>
<head>
<title>Accessing Cookies with PHP</title>
</head>
<body>
<?php
 if( isset($_COOKIE["username"]))
   echo "Welcome " . $_COOKIE["username"] . "<br />";
 else
   echo "Sorry... Not recognized" . "<br />";
?>
</body>
</html>
```

Deleting Cookie with PHP

- To delete a cookie, you simply set the value of the cookie to null, and also set the expiring time of the cookie in the past.

- Ideally, one would set the expiry time to about a year in the past, so that if the system time is off by a few minutes or days even, then the cookie will still delete correctly.

```
<?php

setcookie('username', '', time()-60*60*24*365);

?>
```

Practical Cookies: User Logon

- To give a more detailed look into how to use cookies, I am going to show you how to create a little login form so that you can store the username and password in a cookie.
- We will use more complicated cookies now so that you can learn the full use of setcookie().

```
<html>
<head>
<title>User Logon</title>
</head>
<body>
 <h2>User Login </h2>
 <form name="login" method="post" action="login.php">
  Username: <input type="text" name="username"><br>
  Password: <input type="password" name="password"><br>
  Remember Me: <input type="checkbox" name="rememberme" value="1"><br
>
  <input type="submit" name="submit" value="Login!">
  </form>
</body>
</html>
```

Output:

- Now that we have our form, we will create our login script. We must decide what restrictions we are going to place on the cookie.
- I have decided that this will only run on the www.example.com domain and in the /account directory only. Hence,

```
<?php
/* These are our valid username and passwords */
$user = 'Harshith';
$pass = 'maloth';

if (isset($_POST['username']) && isset($_POST['password'])) {

  if (($_POST['username'] == $user) && ($_POST['password'] == $pass)) {

    if (isset($_POST['rememberme'])) {
      /* Set cookie to last 1 year */
      setcookie('username', $_POST['username'], time()+60*60*24*365, '/account', '
www.example.com');
      setcookie('password', md5($_POST['password']), time()+60*60*24*365, '/accou
nt', 'www.example.com');

    } else {
      /* Cookie expires when browser closes */
      setcookie('username', $_POST['username'], false, '/account', 'www.example.co
m');
      setcookie('password', md5($_POST['password']), false, '/account', 'www.exampl
e.com');
```

```
            }
            header('Location: index.php');

        } else {
            echo 'Username/Password Invalid';
        }

    } else {
        echo 'You must supply a username and password.';
    }
    ?>
```

Accessing the Data:

Validating the data:

```
<?php
/* These are our valid username and passwords */
$user = 'Harshith';
$pass = 'maloth';

if (isset($_COOKIE[['username']) && isset($_COOKIE['password'])) {

    if (($_POST['username'] != $user) || ($_POST['password'] != md5($pass))) {
        header('Location: login.html');
    } else {
        echo 'Welcome back ' . $_COOKIE['username'];
    }

} else {
    header('Location: login.html');
}
?>
```

Explanations:

- In this script, we just check that the cookie exists and is valid. If they aren't, then the user is redirected back to the login form.

- Otherwise a welcome message is included. The only important thing to notice is how we have validated the password. Before on the login.php script, we have encrypted our password using md5() and as this encryption cannot be undone,
- we must compare encrypted versions. Hence, we encrypt our preset value and compare it to the already hashed cookie value. This way, there is no chance of the original password becoming available.

Deleting Cookies

- Cookies as very picky about how they are deleted and usually require that they be removed, using the same values as they were set with. Hence, if we have a domain or path specified, then we must specify this domain/path when deleting the cookie.
- To delete a cookie, you simply set the value of the cookie to null, and also set the expiring time of the cookie in the past. Ideally, one would set the expiry time to about a year in the past, so that if the system time is off by a few minutes or days even, then the cookie will still delete correctly.

```
<?php

setcookie('username', '', time()-60*60*24*365);

?>
```

- ●
- User Logout:

```
<?php

setcookie('username', '', time()-60*60*24*365, '/account', 'www.example.com');
setcookie('password', '', time()-60*60*24*365, '/account', 'www.example.com');

header('Location: login.html');

?>
```

Chapter-2

Using HTTP Headers:

This chapter demonstrates how PHP can provide dynamic content according to browser type, randomly generated numbers or User Input. It also demonstrated how the client borwser can be redirected.

Identifying Browser & Platform

- PHP creates some useful environment variables that can be seen in the phpinfo.php page that was used to setup the PHP environment.
- One of the environemnt variables set by PHP is HTTP_USER_AGENT which identifies the user's browser and operating system.
- PHP provides a function getenv() to access the value of all the environment variables. The information contained in the HTTP_USER_AGENT environment variable can be used to create dynamic content appropriate to the borwser.

Following example demonstrates how you can identify a client borwser and operating system.

```
<html>
<body>
<?php
  $viewer = getenv( "HTTP_USER_AGENT" );
  $browser = "An unidentified browser";
  if( preg_match( "/MSIE/i", "$viewer" ) )
  {
    $browser = "Internet Explorer";
  }
  else if(  preg_match( "/Netscape/i", "$viewer" ) )
  {
    $browser = "Netscape";
  }
  else if(  preg_match( "/Mozilla/i", "$viewer" ) )
  {
```

```
    $browser = "Mozilla";
  }
  $platform = "An unidentified OS!";
  if( preg_match( "/Windows/i", "$viewer" ) )
  {
    $platform = "Windows!";
  }
  else if ( preg_match( "/Linux/i", "$viewer" ) )
  {
    $platform = "Linux!";
  }
  echo("You are using $browser on $platform");
?>
</body>
</html>
```

This is producing following result on my machine. This result may be different for your computer depnding on what you are using.

You are using Mozilla! on Windows!

Display Images Randomly

- The PHP rand() function is used to generate a random number.i This function can generate numbers with-in a given range.
- The random number generator should be seeded to prevent a regular pattern of numbers being generated. This is achieved using the srand() function that specifiies the seed number as its argument.

Following example demonstrates how you can display different image each time out of four images:

```
<html>
<body>
<?php
 srand( microtime() * 1000000 );
 $num = rand( 1, 4 );

 switch( $num )
 {
 case 1: $image_file = "/home/images/alfa.jpg";
       break;
```

```
  case 2: $image_file = "/home/images/ferrari.jpg";
        break;
  case 3: $image_file = "/home/images/jaguar.jpg";
        break;
  case 4: $image_file = "/home/images/porsche.jpg";
        break;
  }
  echo "Random Image : <img src=$image_file />";
?>
</body>
</html>
```

Using HTML Forms

- The most important thing to notice when dealing with HTML forms and PHP is that any form element in an HTML page will automatically be available to your PHP scripts.

Try out following example by putting the source code in test.php script.

```
<?php
  if( $_POST["name"] || $_POST["age"] )
  {
    echo "Welcome ". $_POST['name']. "<br />";
    echo "You are ". $_POST['age']. " years old.";
    exit();
  }
?>
<html>
<body>
  <form action="<?php $_PHP_SELF ?>" method="POST">
  Name: <input type="text" name="name" />
  Age: <input type="text" name="age" />
  <input type="submit" />
  </form>
</body>
</html>
```

- The PHP default variable $_PHP_SELF is used for the PHP script name and when you click "submit" button then same PHP script will be called and will produce following result:
- The method = "POST" is used to post user data to the server script. There are two methods of posting data to the server script which are discussed in PHP GET & POST chapter.

Browser Redirection

- The PHP header() function supplies raw HTTP headers to the browser and can be used to redirect it to another location. The redirection script should be at the very top of the page to prevent any other part of the page from loading.
- The target is specified by the Location: header as the argument to the header() function. After calling this function the exit() function can be used to halt parsing of rest of the code.
- Following example demonstrates how you can redirect a browser request to another web page. Try out this example by putting the source code in test.php script.

```
<?php
 if( $_POST["location"] )
 {
    $location = $_POST["location"];
    header( "Location:$location" );
    exit();
 }
?>
<html>
<body>
  <p>Choose a site to visit :</p>
  <form action="<?php $_PHP_SELF ?>" method="POST">
  <select name="location">
     <option value="http://w3c.org">
          World Wise Web Consortium
     </option>
     <option value="http://www.google.com">
          Google Search Page
     </option>
  </select>
  <input type="submit" />
  </form>
</body>
</html>
```

Displaying "File Download" Dialog Box

- Sometime it is desired that you want to give option where a use will click a link and it will pop up a "File Download" box to the user in stead of displaying actual content. This is very easy and will be achived through HTTP header.

- The HTTP header will be different from the actual header where we send Content-Type as text/html\n\n. In this case content type will be application/octet-stream and actual file name will be concatenated alongwith it.
- For example,if you want make a FileName file downloadable from a given link then its syntax will be as follows.

```
#!/usr/bin/perl

# HTTP Header
print "Content-Type:application/octet-stream; name=\"FileName\"\r\n";
print "Content-Disposition: attachment; filename=\"FileName\"\r\n\n";

# Actual File Content
open( FILE, "<FileName" );
while(read(FILE, $buffer, 100) )
{
   print("$buffer");
}
```

Using Sessions:

- An alternative way to make data accessible across the various pages of an entire website is to use a PHP Session.
- A session creates a file in a temporary directory on the server where registered session variables and their values are stored. This data will be available to all pages on the site during that visit.
- The location of the temporary file is determined by a setting in the php.ini file called session.save_path. Bore using any session variable make sure you have setup this path.

When a session is started following things happen:

- PHP first creates a unique identifier for that particular session which is a random string of 32 hexadecimal numbers such as 3c7foj34c3jj973hjkop2fc937e3443.
- A cookie called PHPSESSID is automatically sent to the user's computer to store unique session identification string.

- A file is automatically created on the server in the designated temporary directory and bears the name of the unique identifier prefixed by sess_ ie sess_3c7foj34c3jj973hjkop2fc937e3443.
- When a PHP script wants to retrieve the value from a session variable, PHP automatically gets the unique session identifier string from the PHPSESSID cookie and then looks in its temporary directory for the file bearing that name and a validation can be done by comparing both values.
- A session ends when the user loses the browser or after leaving the site, the server will terminate the session after a predetermined period of time, commonly 30 minutes duration.

Starting a PHP Session:

- A PHP session is easily started by making a call to the session_start() function.This function first checks if a session is already started and if none is started then it starts one. It is recommended to put the call to session_start() at the beginning of the page.
- Session variables are stored in associative array called $_SESSION[]. These variables can be accessed during lifetime of a session.
- The following example starts a session then register a variable called counter that is incremented each time the page is visited during the session.
- Make use of isset() function to check if session variable is already set or not.

Put this code in a test.php file and load this file many times to see the result:

```
<?php
  session_start();
  if( isset( $_SESSION['counter'] ) )
  {
    $_SESSION['counter'] += 1;
  }
  else
  {
    $_SESSION['counter'] = 1;
  }
  $msg = "You have visited this page ".  $_SESSION['counter'];
  $msg .= "in this session.";
?>
<html>
<head>
<title>Setting up a PHP session</title>
</head>
<body>
<?php  echo ( $msg ); ?>
</body>

</html>
```

Destroying a PHP Session:

- A PHP session can be destroyed by session_destroy() function. This function does not need any argument and a single call can destroy all the session variables. If you want to destroy a single session variable then you can use unset() function to unset a session variable.

Here is the example to unset a single variable:

```
<?php

unset($_SESSION['counter']);
?>
```

Here is the call which will destroy all the session variables:

```
<?php

session_destroy();
?>
```

Turning on Auto Session:

- You don't need to call start_session() function to start a session when a user visits your site if you can set session.auto_start variable to 1 in php.ini file.

Sessions without cookies:

- There may be a case when a user does not allow to store cookies on their machine. So there is another method to send session ID to the browser.
- Alternatively, you can use the constant SID which is defined if the session started. If the client did not send an appropriate session cookie, it has the form session_name=session_id. Otherwise, it expands to an empty string. Thus, you can embed it unconditionally into URLs.
- The following example demonstrates how to register a variable, and how to link correctly to another page using SID.

```
<?php
  session_start();

  if (isset($_SESSION['counter'])) {
    $_SESSION['counter'] = 1;
  } else {
    $_SESSION['counter']++;
  }
?>
  $msg = "You have visited this page ".  $_SESSION['counter'];
  $msg .= "in this session.";
  echo ( $msg );
<p>
To continue  click following link <br />
<a  href="nextpage.php?<?php echo htmlspecialchars(SID); >">

</p>
```

The htmlspecialchars() may be used when printing the SID in order to prevent XSS related attacks.

Chapter-3:

User Authentication:

Example on login page usi ng MySql and PHP:

- In this tutorial we will make a full PHP Login script, what will allow users to register, login and to log out. This system can be easily used on any PHP website. In this tutorial we will deal more with the technical (PHP) part and less with styling it, as you can see it in the bellow example of how the login system will look and work:

- When somebody registers the username and the password is stored in a MySQL table.
- When somebody tries to log in the script will verify if exists an user with the given username and password in the MySQL table. If the login was successful a session is created and a welcome message is displayed. From here the user can log out by destroying the session.

Creating the *users* MySQL table

- As I said, the username and the password of each user will be stored in a MySQL table. I will call this table *user*. We need two columns: *username* and *password*. I`we used the following file to create the table:

1.*create-table.php*

```
<?php
//Connect to MySQL
mysql_connect('host', 'database', 'password') or die (mysql_error());
//Select database
mysql_select_db('database') or die (mysql_error());
//Create the table
mysql_query("create table users(
username varchar(30) NOT NULL,
password varchar(30) NOT NULL,
PRIMARY KEY (username)
)") or die (mysql_error());
//Show "Complete" if everything works
echo "Complete.";
?>
```

Explanations:

- "Username" and "password" is the name of the field.
  "varchar" is the data type of the field - it will contain characters
  "(30)" is the maximum length of the field
  "NOT NULL" means the field can`t be leaved blank
  And we specify the PRIMARY KEY:
- Now just run your *create-table.php* but don`t forget to change the host, database and the password to yours. If everything works you will see the message: "Complete.", otherwise an error message. In this case double check your code

2. Index.php page:

Log in

Username:

Password:

Log in

Don`t have an account?

Register

Username:

Password:

Retype password:

Register

```
<?php
session_start();
?>
<html>
<head>
</head>

<body>
<?php
if(!session_is_registered("username")){?>
<form action="login.php" method="post">
<b style="font-size:150%;">Log in</b><br/>
Username: <input type="text" name="username"/><br/>
Password: <input type="password" name="password"/><br/>
<input type="submit" value="Log in"/>
</form>
Don`t have an account?
<form action="register.php" method="post"><br/>
<b style="font-size:150%;">Register</b><br/>
Username: <input type="text" name="username"/><br/>
Password: <input type="password" name="password"/><br/>
Retype password: <input type="password" name="retype-password"/><br />
<input type="submit" value="Register" />
```

```
</form>
<?php }
else{
echo 'Welcome ' . $_SESSION["username"] . '<br/><a href="logout.php">Log out</a>';
}?>
</body>
</html>
```

Login.php:

```
<?php
session_start();
mysql_connect('host', 'database', 'password')
or die ("Could not connect to mysql because ".mysql_error());
mysql_select_db('database')
or die ("Could not select database because ".mysql_error());
if(mysql_num_rows(mysql_query("SELECT * FROM users WHERE username='".$_POST["username"]."'
and password='".$_POST["password"]."'"))==1){
session_register("username");
header("location:index.php");
}
else {
echo 'Wrong username or password!';
}
?>
```

Register.php:

```
<?php
mysql_connect('host', 'database', 'password') or die (mysql_error());
mysql_select_db('database') or die (mysql_error());
if(mysql_num_rows(mysql_query("SELECT * from users WHERE username='" . $_POST['username'] . "'"))
== 1){
echo "Sorry, this username is already taken!";
}
else if($_POST['password'] != $_POST['retype-password']){
echo "The two passwords don`t match!";
}
else if(strlen($_POST['username']) > 15){
```

```
echo "Username is too long!";
}
else if(strlen($_POST['username']) < 6){
echo "Username is too short!";
}
else if(strlen($_POST['password']) > 15){
echo "Password is too long!";
}
else if(strlen($_POST['password']) < 6){
echo "Password is too short!";
}
else if(preg_match('/[^0-9A-Za-z]/',$_POST['username'])){
echo "Invalid characters in username!";
}
else if(preg_match('/[^0-9A-Za-z]/',$_POST['password'])){
echo "Invalid characters in password!";
}
else{
mysql_query("INSERT into users VALUES ('".$_POST['username']."', '".$_POST['password']."')") or
die(mysql_error());
}
?>
```

Logout.php:

```
<?php
session_start();
session_destroy(); //Delete session
header("location:index.php");
?>
```

Chapter-4:

Using Environment and Configuration variables

Predefined Variables

- PHP provides a large number of predefined variables to all scripts.

- The variables represent everything from external variables to built-in environment variables, last error messages to last retrieved headers.

- Superglobals — Superglobals are built-in variables that are always available in all scopes
- $GLOBALS — References all variables available in global scope
- $_SERVER — Server and execution environment information
- $_GET — HTTP GET variables
- $_POST — HTTP POST variables
- $_FILES — HTTP File Upload variables
- $_REQUEST — HTTP Request variables
- $_SESSION — Session variables
- $_ENV — Environment variables
- $_COOKIE — HTTP Cookies
- $php_errormsg — The previous error message
- $HTTP_RAW_POST_DATA — Raw POST data
- $http_response_header — HTTP response headers
- $argc — The number of arguments passed to script
- $argv — Array of arguments passed to script

$_SERVER: $HTTP_SERVER_VARS [deprecated]

$_SERVER -- $HTTP_SERVER_VARS [deprecated] — Server and execution environment information

Description

- *$_SERVER* is an array containing information such as headers, paths, and script locations. The entries in this array are created by the web server.
- There is no guarantee that every web server will provide any of these; servers may omit some, or provide others not listed here. That said, a large number of these variables are accounted for in the » CGI/1.1 specification, so you should be able to expect those.

Example #1 *$_SERVER* example

```
<?php
echo $_SERVER['SERVER_NAME'];
?>
```

The above example will output something similar to:

```
www.example.com
```

NOTE:

$HTTP_SERVER_VARS contains the same initial information, but is not a superglobal. (Note that *$HTTP_SERVER_VARS* and *$_SERVER* are different variables and that PHP handles them as such)

$GLOBALS:

$GLOBALS — References all variables available in global scope

Description

An associative array containing references to all variables which are currently defined in the global scope of the script. The variable names are the keys of the array.

Example #1 *$GLOBALS* example

```php
<?php
function test() {
    $foo = "local variable";

    echo '$foo in global scope: ' . $GLOBALS["foo"] . "\n";
    echo '$foo in current scope: ' . $foo . "\n";
}

$foo = "Example content";
test();
?>
```

The above example will output something similar to:

```
$foo in global scope: Example content
$foo in current scope: local variable
```

$_ENV: $HTTP_ENV_VARS [deprecated]

$_ENV -- $HTTP_ENV_VARS [deprecated] — Environment variables

- An associative array of variables passed to the current script via the environment method.
- These variables are imported into PHP's global namespace from the environment under which the PHP parser is running. Many are provided by the shell under which PHP is running and different systems are likely running different kinds of shells, a definitive list is impossible
- *$HTTP_ENV_VARS* contains the same initial information, but is not a superglobal. (Note that *$HTTP_ENV_VARS* and *$_ENV* are different variables and that PHP handles them as such)

Example #1 *$_ENV* example

```
<?php
echo 'My username is ' .$_ENV["USER"] . '!';
?>
```

Assuming "bjori" executes this script

The above example will output something similar to:

My username is bjori!

$_GET: $HTTP_GET_VARS [deprecated]

$_GET -- $HTTP_GET_VARS [deprecated] — HTTP GET variables

Description

- An associative array of variables passed to the current script via the URL parameters.
- *$HTTP_GET_VARS* contains the same initial information, but is not a superglobal. (Note that *$HTTP_GET_VARS* and *$_GET* are different variables and that PHP handles them as such)

Examples

Example #1 *$_GET* example

```
<?php
echo 'Hello ' . htmlspecialchars($_GET["name"]) . '!';
?>
```

Assuming the user entered http://example.com/?name=Hannes

The above example will output something similar to:

```
Hello Hannes!
```

$_POST

$HTTP_POST_VARS [deprecated]

$_POST -- $HTTP_POST_VARS [deprecated] — HTTP POST variables

Description

- An associative array of variables passed to the current script via the HTTP POST method.
- *$HTTP_POST_VARS* contains the same initial information, but is not a superglobal. (Note that *$HTTP_POST_VARS* and *$_POST* are different variables and that PHP handles them as such)

Examples

Example #1 *$_POST* example

```
<?php
echo 'Hello ' . htmlspecialchars($_POST["name"]) . '!';
?>
```

Assuming the user POSTed name=Hannes

The above example will output something similar to:

Hello Hannes!

Chapter-5:

Working with 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():

- The PHP date() function is used to format a time and/or date.
- 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.
- The PHP date() function formats a timestamp to a more readable date and time.

A timestamp is a sequence of characters, denoting the date and/or time at which a certain event occurred.

Syntax:

date(*format*, *timestamp*)

Parameter	Description
format	Required. Specifies the format of the timestamp
timestamp	Optional. Specifies a timestamp. Default is the current date and time

PHP Date() - Format the Date

The required *format* parameter in the date() function specifies how to format the date/time.

Here are some characters that can be used:

- d - Represents the day of the month (01 to 31)
- m - Represents a month (01 to 12)
- Y - Represents a year (in four digits)

Eg:

```
<?php
echo date("Y/m/d") . "<br />";
echo date("Y.m.d") . "<br />";
echo date("Y-m-d");
```

```
?>
```

The output of the code above could be something like this:

```
2009/05/11
2009.05.11
2009-05-11
```

PHP Date() - Adding a Timestamp

- The optional *timestamp* parameter in the date() function specifies a timestamp. If you do not specify a timestamp, the current date and time will be used.
- The mktime() function returns the Unix timestamp for a date.
- The Unix timestamp contains the number of seconds between the Unix Epoch (January 1 1970 00:00:00 GMT) and the time specified.

Syntax for mktime()

```
mktime(hour,minute,second,month,day,year,is_dst)
```

To go one day in the future we simply add one to the day argument of mktime():

```
<?php
$tomorrow = mktime(0,0,0,date("m"),date("d")+1,date("Y"));
echo "Tomorrow is ".date("Y/m/d", $tomorrow);
?>
```

The output of the code above could be something like this:

```
Tomorrow is 2009/05/12
```

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

It will produce following result:

```
948316201
```

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

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

- Now you have complete control over date and time. You can format this date and time in whatever format you wan.

Example:

Try out following example

```
<?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;

?>
```

It will produce following result:

```
seconds = 27
minutes = 25
hours = 11
mday = 12
wday = 6
mon = 5
year = 2007
yday = 131
weekday = Saturday
month = May
0 = 1178994327
Today's date: 12/5/2007
```

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.

```
date(format,timestamp)
```

The date() optionally accepts a time stamp if ommited then current date and time will be used. Any other data you include in the format string passed to date() will be included in the return value.

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

Example:

Try out following example

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

```
print "Today is ";
print date("j of F Y, \a\\t g.i a", time());
?>
```

It will produce following result:

```
01/20/00 13.27:55
Today is 20 of January 2000, at 1.27 pm
```

----------*THE END*----------