

# **PHP Tutorial**

# What is PHP?

- PHP == 'PHP: Hypertext Preprocessor' (recursive backronym)
- Open-source, server-side scripting language
- Used to generate dynamic web-pages
- PHP scripts reside between reserved PHP tags
- This allows the programmer to embed PHP scripts within HTML pages

# History of PHP

- PHP began in 1995 when Rasmus Lerdorf developed a Perl/CGI script toolset he called the Personal Home Page or PHP
- PHP 2 released 1997 (PHP now stands for Hypertext Processor). Lerdorf developed it further, using C instead
- PHP 3 released in 1998 (50,000 users)
- PHP 4 released in 2000 (3.6 million domains). Considered debut of functional language and including Perl parsing, with other major features
- PHP 5.0.0 released July 13, 2004 (113 libraries>1,000 functions with extensive object-oriented programming)
- PHP 5.6.31 released July 6, 2017
- PHP 7.0 released December 2015
- Documentation available at: <http://www.php.net/docs.php>
- PHP 7.1 current “stable” release

# What is PHP (cont'd)

- Interpreted language, scripts are parsed at run-time rather than compiled beforehand
- Executed on the server-side
- Source-code not visible by client: e.g. 'View Source' in browsers does not display the PHP code
- Various built-in functions allow for fast development
- Compatible with many popular databases
- Provides
  - + SAPI, Server Application Programming Interface (Direct Module)
  - + ISAPI, The Internet Server Application Programming Interface (IIS)
  - + CGI interface
  - + CLI (Command Line Interface)

# What does PHP code look like?

- Structurally similar to C/C++
- Supports procedural and object-oriented paradigm (to some degree)
- All PHP statements end with a semi-colon
- Each PHP script must be enclosed in the reserved PHP tag

```
<?php  
...  
?>
```

# A Simple Example

- Save as sample.php:

```
<!-- sample.php -->
```

```
<html><body>
```

```
<strong>Hello World!</strong><br />
```

```
<?php
```

```
    echo "<h2>Hello, World</h2>";
```

```
?>
```

```
<?php
```

```
$myvar = "Hello World";
```

```
echo $myvar;
```

```
?>
```

```
</body></html>
```



# Comments in PHP

- Standard C, C++, and shell comment symbols

```
// C++ and Java-style comment  
  
# Shell-style comments  
  
/* C-style comments  
    These can span multiple lines */
```

# Variables in PHP

- PHP variables must begin with a “\$” sign
- The variable name must be followed by a letter or underscore
- Case-sensitive (\$Foo != \$foo != \$fOo)
- Static, Global and locally-scoped variables
  - + Global variables can be used anywhere (declared outside a function)
  - + Local variables restricted to a function or class
  - + Variable declared “static” do not disappear when a function is completed
- Certain variable names reserved by PHP. Examples:
  - + Form variables (\$\_POST, \$\_GET)
  - + Server variables (\$\_SERVER)



# PHP Variables

- Variables are not statically typed
- Integers can become floats, then can become strings
- Variables take the type of the current value
- Variable types include :
  - Boolean
  - Integer
  - Float
  - String
  - Array
  - Object
  - Resource
  - NULL

# PHP Variables

- Assigned by value

```
$foo = "Bob"; $bar = $foo;
```

- Assigned by reference, this links vars

```
$bar = &$foo;
```

- Some variables are pre-assigned, e.g. server and env vars

- For example, there are PHP vars

- + `PHP_SELF`, a variable that returns the current script being executed, including its name and path (Ex:

- `$_SERVER['PHP_SELF']` )

- + `$HTTP_GET_VARS`, an associative array of variables passed to the current script (deprecated by `$_GET`)

# Displaying Variables

- To display a variable with the `echo` statement, pass the variable name to the `echo` statement without enclosing it in quotation marks :

```
$VotingAge = 18;  
echo $VotingAge;
```

- To display both text strings and variables, send them to the `echo` statement as individual arguments, separated by commas :

```
echo "<p>The legal voting age is ", $VotingAge,  
    ".</p>";
```

# Naming Variables

- The following rules and conventions must be followed when naming a variable:
  - Variable names must begin with a dollar sign (\$)
  - Variable names may contain uppercase and lowercase letters, numbers, or underscores (\_). The first character after the dollar sign must be a letter or underscore.
  - Variable names cannot contain spaces
  - Variable names are case sensitive

# PHP Constants

- Constants are special variables that cannot be changed
- Constant names do not begin with a dollar sign (\$)
- Start with letter or underscore (\_) followed by letters, numbers or underscores
- Use them for named items that will not change
- Constant names use all uppercase letters
- Use the **define()** function to create a constant  
`define("CONSTANT_NAME", value);`
- The value you pass to the `define()` function can be a text string, number, or Boolean value
- Constants have global scope

# PHP Operators

- Standard Arithmetic operators

`+, -, *, /` and `%` (modulus)

- String concatenation with a period (`.`)

```
$car = "SEAT" . " Altea";
```

`echo $car;` would output `"SEAT Altea"`

- Basic Boolean comparison with `"=="`
- Using only `=` will overwrite a variable value
- Less than `<` and greater than `>`
- `<=` and `>=` as above but include equality

# PHP Operators (cont'd)

- Assignment (=) and combined assignment

```
$a = 3;
```

```
$a += 5; // sets $a to 8;
```

```
$b = "Hello ";
```

```
$b .= "There!"; // sets $b to "Hello There!";
```

- Bitwise (&, |, ^, ~, <<, >>)

```
$a ^ $b (Xor: Bits that are set in $a or $b but not both  
are set.)
```

```
~ $a (Not: Bits that are set in $a are not set, and  
vice versa.)
```

# Data Types

- PHP is **not** strictly typed
- PHP support 8 “primitive” types:
  - + 4 scalar types: boolean, integer, double (floating-point), string
  - + 2 compound types: array and object
  - + 2 special types; resource and null
- PHP decides what type a variable is
- PHP can use variables in an appropriate way automatically
- E.g.

```
$vat_rate = 0.175; /* VAT Rate is numeric */  
echo $vat_rate * 100 . “%”; //outputs “17.5%”
```
- `$vat_rate` is converted to a string for the purpose of the echo statement



# Numeric Data Types

- PHP supports two numeric data types:
  - An **integer** is a positive or negative number and 0 with no decimal places (-250, 2, 100, 10,000)
  - A **floating-point number** (double) is a number that contains decimal places or that is written in exponential notation (-6.16, 3.17, 2.7541)

# Boolean Values

- A **Boolean value** is a value of TRUE or FALSE
- It decides which part of a program should execute and which part should compare data
- In PHP programming, you can only use TRUE or FALSE Boolean values
- In other programming languages, you can use integers such as 1 = TRUE, 0 = FALSE

# Variable usage

```
<?php
$foo = 25;           //Numerical variable
$bar = "Hello";      //String variable

$foo = ($foo * 7);    //Multiplies foo by 7
$bar = ($bar * 7);    //Invalid expression
?>
```

# echo example

```
<?php
$foo = 25;           // Numerical variable
$bar = "Hello";      // String variable

echo $bar;           // Outputs Hello
echo $foo, $bar;      // Outputs 25Hello
echo "5x5=", $foo;    // Outputs 5x5=25
echo "5x5=$foo";      // Outputs 5x5=25
echo '5x5=$foo';      // Outputs 5x5=$foo
?>
```

- Notice how echo '5x5=\$foo' outputs \$foo rather than replacing it with 25
- Strings in single quotes ( ' ') are not interpreted or evaluated by PHP
- This is true for both variables and character escape-sequences (such as "\\n" or "\\")

# Arithmetic Operations

```
<?php
    $a=15;
    $b=30;
    $total=$a+$b;
    Print $total;
    Print "<p><h1>$total</h1>";
    // total is 45
?>
```

`$a - $b`      `// subtraction`

`$a * $b`      `// multiplication`

`$a / $b`      `// division`

`$a += 5`      `// $a = $a+5`      - also works for `*=` and `/=`

# Concatenation

- Use a period to join strings into one.

```
<?php
$string1="Hello";
$string2="PHP";
$string3=$string1 . " " . $string2;
Print $string3;
?>
```

Hello PHP

# Escaping the Character

- If the string has a set of double quotation marks that must remain visible, use the \ [backslash] before the quotation marks to ignore and display them.

```
<?php  
$heading=" \"Computer Science\"";  
Print $heading;  
?>
```

```
"Computer Science"
```

# PHP Control Structures

- Control Structures: the structures within a language that allow us to control the flow of execution through a program or script.
- Grouped into conditional / branching structures (e.g. if/else) and repetition structures (e.g. while loops).
- Example if/else if/else statement:

```
if ($foo == 0) {  
    echo 'The variable foo is equal to 0';  
}  
else if (($foo > 0) && ($foo <= 5)) {  
    echo 'The variable foo is between 1 and 5';  
}  
else {  
    echo 'The variable foo is equal to '.$foo;  
}
```



# If ... Else...

```
If (condition)
{
    Statements;
}
Else
{
    Statement;
}
```

```
<?php
If ($user==" John" )
{
    Print "Hello John.";
}
Else
{
    Print "You are not John.";
}
?>
```

No 'Then' in PHP !

# While Loops

General format:

```
While (condition)
{
    Statements;
}
```

```
<?php
$count=0;
While ($count<3)
{
    Print "hello PHP. ";
    $count += 1;
    // $count = $count + 1;
    // or
    // $count++;
}
?>
```

```
hello PHP. hello PHP. hello PHP.
```

# Date Display

```
$datedisplay=date("yyyy/m/d");  
Print $datedisplay;  
# If the date is April 1st, 2012  
# It would display as 2012/4/1
```

**2012/4/1**

```
$datedisplay=date("l, F m, Y"); (see next slide for symbol  
meanings)
```

```
Print $datedisplay;  
# If the date is April 1st, 2012  
# Wednesday, April 1, 2012
```

**Wednesday, April 1, 2012**

# Month, Day & Date Format Symbols

|   |         |
|---|---------|
| M | Jan     |
| F | January |
| m | 01      |
| n | 1       |

|              |   |        |
|--------------|---|--------|
| Day of Month | d | 01     |
| Day of Month | J | 1      |
| Day of Week  | l | Monday |
| Day of Week  | D | Mon    |

# Functions

- Functions MUST be defined before they can be called
- Function headers are of the format

```
function functionName($arg_1, $arg_2, ..., $arg_n)
```

+ Note that no return type is specified

- Unlike variables, function names are not case sensitive

```
(foo (...) == Foo (...) == FoO (...))
```

# Functions example

```
<?php
    // This is a function
    function foo($arg_1, $arg_2)
    {
        $arg_2 = $arg_1 * $arg_2;
        return $arg_2;
    }

    $result_1 = foo(12, 3);      // Store the function
    echo $result_1;             // Outputs 36
    echo foo(12, 3);            // Outputs 36
?>
```

# Include Files

- Include “footer.php” by placing within the HTML the line  
`<?php include 'footer.php'; ?>`

- The file footer.php might look like:

```
<hr SIZE=11 NOSHADE WIDTH="100%">
<i>Copyright © 2010-2012 USC </i></font><br>
<i>ALL RIGHTS RESERVED</i></font><br>
<i>URL: http://www.usc.edu</i></font><br>
```

# Some simple examples

These examples can be found at

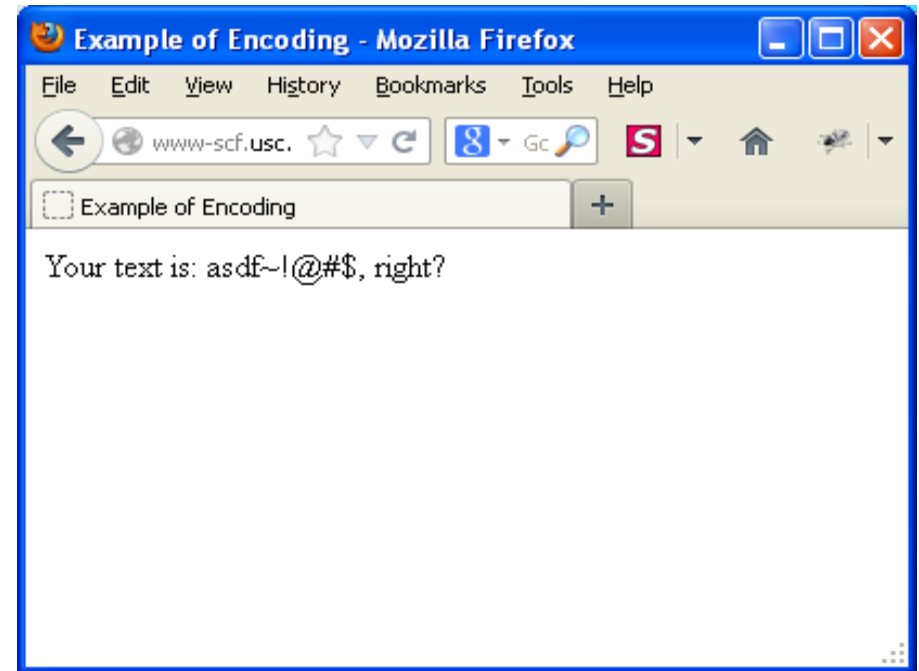
[http://cs-server.usc.edu:45678/examples/php/php\\_ex/translated/index.html](http://cs-server.usc.edu:45678/examples/php/php_ex/translated/index.html)



# Example – Encoding Form Data



before



after

# Example – Encoding Form Data Script

```
<HTML>
<HEAD><TITLE>Example of Encoding</TITLE></HEAD>
<BODY>
<?php if($_POST["submit"]): ?>
    Your text is: <?php echo $_POST["input"]; ?>, right?
<?php else: ?>
    <H1>Please enter some text</H1>
    <FORM ACTION="" METHOD=POST>
    Enter text: <INPUT NAME=input><BR>
    <INPUT TYPE=submit name="submit">
    <INPUT TYPE=reset></form>
<?php endif; ?>
</BODY>
</HTML>
```

- **Note:** PHP 5.4+ warns if not doing this: `if (isset($_POST["submit"]))`

# Example – Displaying a Text File

This php program is invoked by clicking on the link

`<a href=http://cs-server.usc.edu:45678/examples/php/php_ex/translated/displayfile.php>`

`Example – Displaying a text file</a>`



Resulting Output

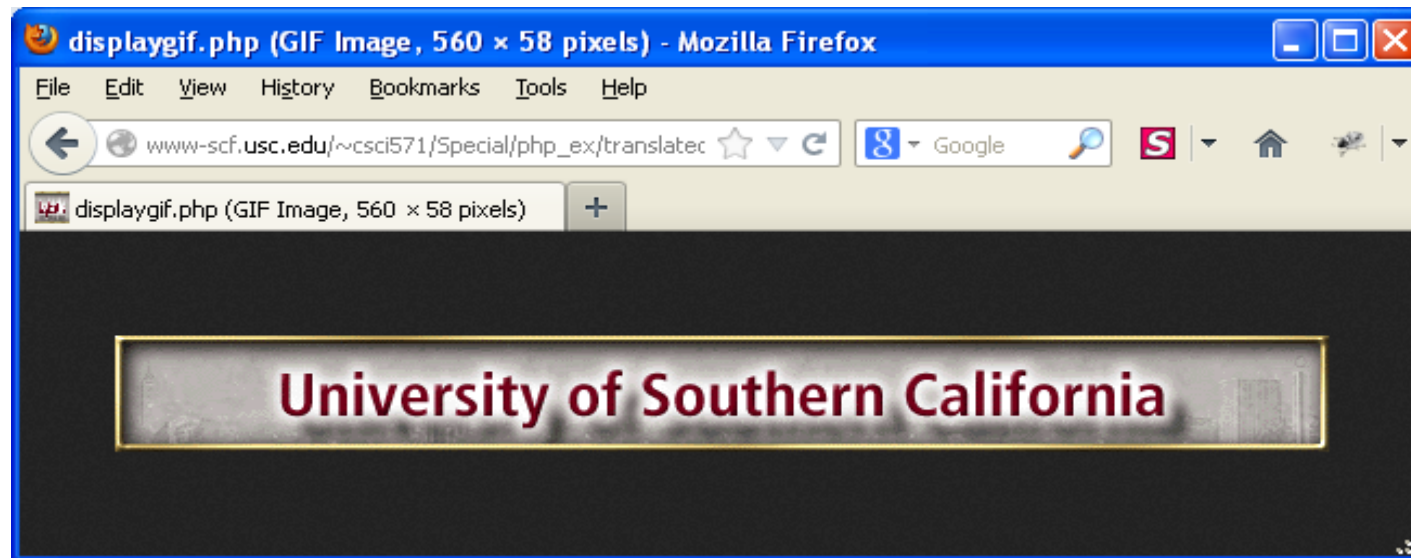
# Example – Source of displayfile.php Script

```
<?php
$ourFileName = "apple.txt";
$ourFileHandle = fopen($ourFileName, "r") or die("can't open
    file");
$file = fread($ourFileHandle, filesize($ourFileName));
fclose($ourFileHandle);
header("Content-type: text/plain");
echo $file;
?>
```

# Example – Returning a GIF Image

This program is also invoked by clicking on a link

`<a href=http://cs-server.usc.edu:45678/examples/php/php_ex/translated/displaygif.php>`  
Example – Returning a GIF Image`</a>`



Resulting Output

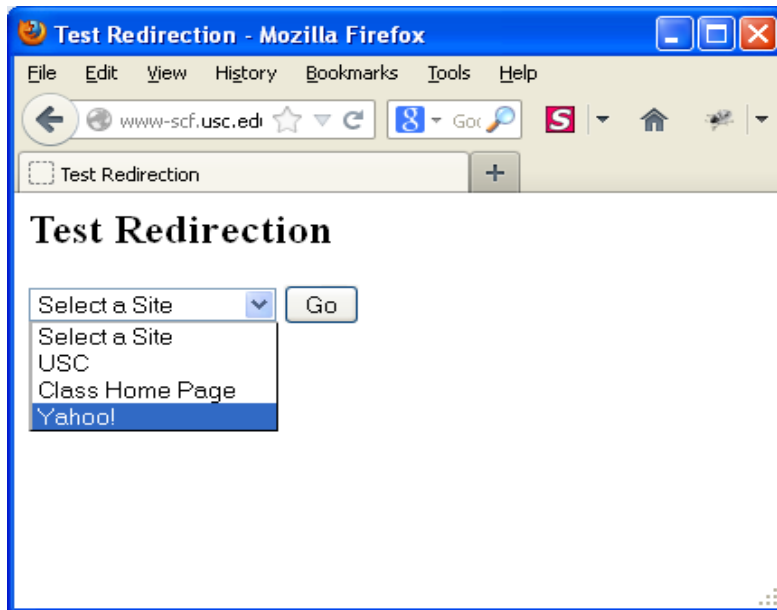
# Example – Source for displaygif.php Script

```
<?php
$ourFileName = "BanS_USC.gif";
$ourFileHandle = fopen($ourFileName, "r") or die("can't
    open file");
$file = fread($ourFileHandle, filesize($ourFileName));
fclose($ourFileHandle);
header("Content-type: image/gif");
echo $file;
?>
```

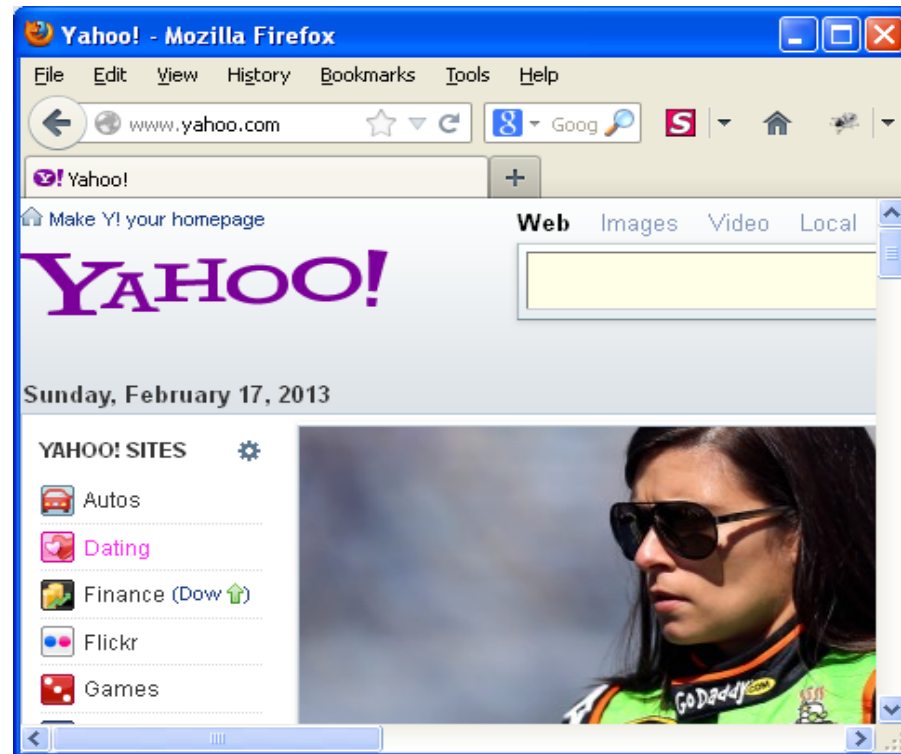
# Example – Redirection in php

This program is also invoked by clicking on a link

`<a href=http://cs-server.usc.edu:45678/examples/php/php_ex/translated/redirection.php>`  
Example – Redirection`</a>`



Before



After

# Example – redirection.php Script

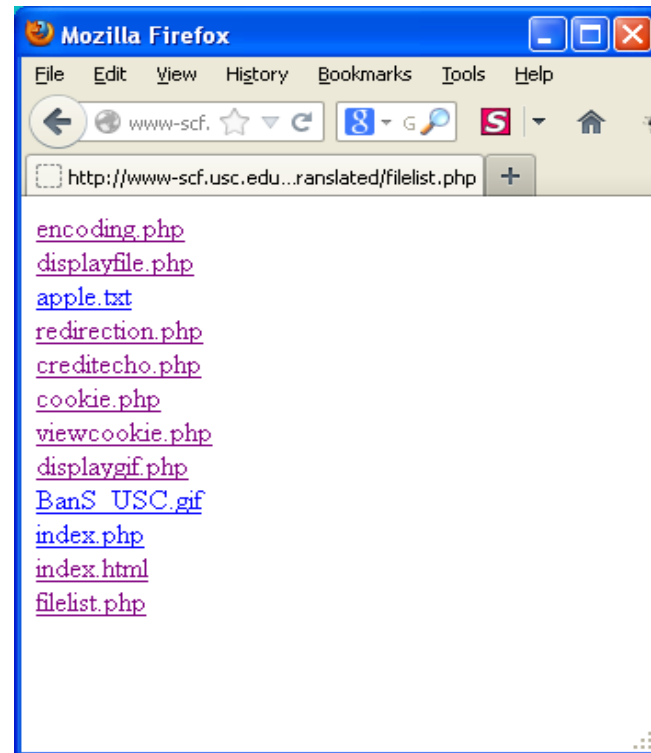
```
<?php if($_POST["submit"]): ?>
<?php header("Location: $_POST[url]"); ?>
<?php else: ?>
<html><head><title>Test Redirection</title></head>
<body>
<h2>Test Redirection</h2>
<form method="post" action="">
<select name="url">
<option selected=selected value="">Select a Site</option>
<option value="http://www.usc.edu">USC</option>
<option value="http://www-scf.usc.edu/~csci571/index.html">Class Home
    Page</option>
<option value="http://www.yahoo.com">Yahoo!</option>
</select> <input type="submit" value="Go" name="submit">
</form></body></html>
<?php endif; ?>
```



# Example - Creating a File List

This program is also invoked by clicking on a link

`<a href=http://cs-server.usc.edu:45678/examples/php/php_ex/translated/filelist.php>`  
Example – Creating a File List`</a>`



Result

# Example – Source for filelist.php Script

```
<?php
//echo getcwd(); get current working directory
if ($handle = opendir('/home/scf-
22/csci571/public_html/Special/php_ex/translated')) {
    while (false !== ($entry = readdir($handle))) {
        if ($entry != "." && $entry != "..") {
            echo "<a href='$entry'>$entry</a><br/>\n";
        }
    }
    closedir($handle);
}
?>
```

# Simple counter example



This example is currently stored at: [http://csci571.herokuapp.com/example\\_counter.php](http://csci571.herokuapp.com/example_counter.php)

# Simple counter script

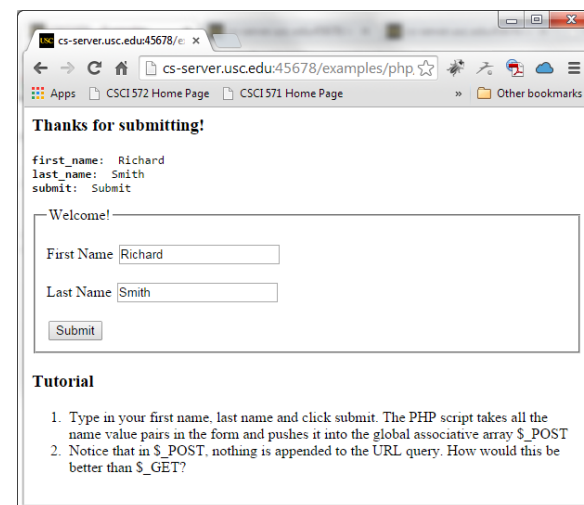
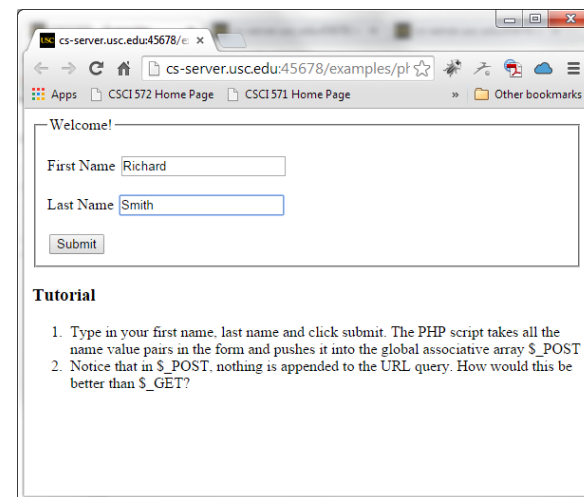
```
<?php
$ourFileName = "counter.txt";
$ourFileHandle = fopen($ourFileName, "r") or die("can't open file");
$num = fread($ourFileHandle, filesize($ourFileName));
fclose($ourFileHandle);
$ourFileHandle = fopen($ourFileName, "w") or die("can't open file");
$num++;
echo "<h1>Welcome Visitor #" . $num . "</h1>";
fwrite($ourFileHandle, $num);
fclose($ourFileHandle);
?>
<h3>Simple Counter PHP Example</h3>
<ol>
<li>The PHP script reads in the txt file and gets the current
    value.</li>
<li>Then, it increments that current value by 1, prints out the visitor
    #, and writes it to the file.</li>
</ol>
```

# PHP - Dealing with the client

- `<form method="post" action="file.php" id="frmId" >`
  - + Method specifies how the data will be sent
  - + Action specifies the file to go to. E.g., file.php
  - + id gives the form a unique name
- Post method sends all contents of a form with basically hidden headers (not easily visible to users)
- Get method sends all form input in the URL requested using name=value pairs separated by ampersands (&)
  - + E.g., process.php?name=trevor&number=345
  - + Is visible in the URL shown in the browser

# PHP - Example Using POST

```
<?php
include_once("inc.php");
if(isset($_POST["submit"])):
?>
<h3>Thanks for submitting!</h3>
<pre>
    <?php print_array($_POST); ?>
</pre>
<?php endif; ?>
<form method="POST" action="">
<fieldset>
    <legend>Welcome!</legend>
    <p><label for="first_name">First Name</label>
<input type="text" name="first_name"
value="<?php echo isset($_POST["first_name"]) ?
$_POST["first_name"] : "" ?>"></p>
    <p><label for="last_name">Last Name</label>
    <input type="text" name="last_name" value="<?php echo
isset($_POST["last_name"]) ? $_POST["last_name"] : "" ?>">
    </p>
    <input type="submit" name="submit" value="Submit">
</fieldset></form>
```



# PHP - Dealing with the client

- Summary
  - + Form elements contain input elements
  - + Each input element has an id
  - + If a form is posted, the file stated as the action can use:  
`$_POST["inputid"]`
  - + If a form uses the get method:  
`$_GET["inputid"]`
- Ensure you set all id attributes for form elements and their contents

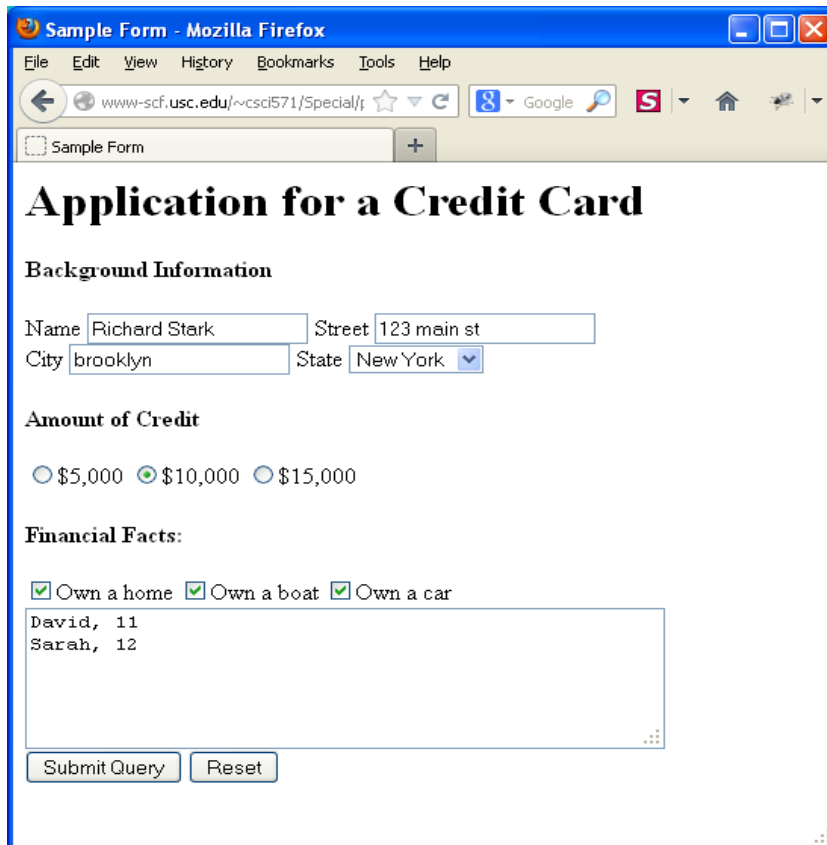
# EXAMPLE - CGI Env. Access

<?php

```
$env["REMOTE_ADDR"] = getenv("REMOTE_ADDR");
$env["SERVER_SOFTWARE"] = getenv("SERVER_SOFTWARE");
$env["SERVER_NAME"] = getenv("SERVER_NAME");
$env["GATEWAY_INTERFACE"] = getenv("GATEWAY_INTERFACE");
$env["SERVER_PROTOCOL"] = getenv("SERVER_PROTOCOL");
$env["SERVER_PORT"] = getenv("SERVER_PORT");
$env["REQUEST_METHOD"] = getenv("REQUEST_METHOD");
$env["PATH_INFO"] = getenv("PATH_INFO");
$env["PATH_TRANSLATED"] = getenv("PATH_TRANSLATED");
$env["DOCUMENT_ROOT"] = getenv("DOCUMENT_ROOT");
$env["SCRIPT_NAME"] = getenv("SCRIPT_NAME");
$env["QUERY_STRING"] = getenv("QUERY_STRING");
$env["REMOTE_HOST"] = getenv("REMOTE_HOST");
$env["AUTH_TYPE"] = getenv("AUTH_TYPE");
$env["REMOTE_USER"] = getenv("REMOTE_USER");
$env["REMOTE_IDENT"] = getenv("REMOTE_IDENT");
$env["CONTENT_TYPE"] = getenv("CONTENT_TYPE");
$env["CONTENT_LENGTH"] = getenv("CONTENT_LENGTH");
$env["HTTP_ACCEPT"] = getenv("HTTP_ACCEPT");
$env["HTTP_HOST"] = getenv("HTTP_HOST");
$env["HTTP_USER_AGENT"] = getenv("HTTP_USER_AGENT");
$env["HTTP_REFERER"] = getenv("HTTP_REFERER");
$env["HTTP_REFERER"] = getenv("HTTP_REFERER");
print "<table border=1 cellspacing=0 cellpadding=2>";
print "<caption>Environment Variables</caption>";
while ( list( $key, $val ) = each( $env ) ) {
    print "<tr><td bgcolor=#CCCCCC><b>$key</b></td><td bgcolor=#EEEEEE><i>$val</i></td></tr>";
}
print "</table>"; }    ?>
```

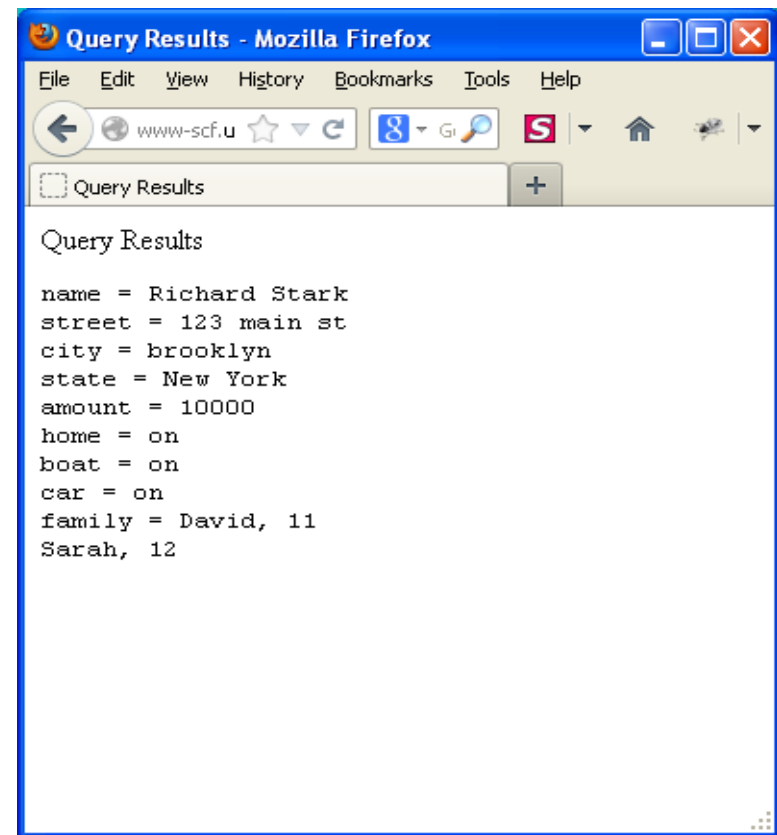


# Example – Echoing Inputs



The screenshot shows a Mozilla Firefox browser window with the title 'Sample Form - Mozilla Firefox'. The address bar shows 'www-scf.usc.edu/~csci571/Special/'. The form is titled 'Application for a Credit Card'. It has sections for 'Background Information' with fields for Name (Richard Stark), Street (123 main st), City (brooklyn), and State (New York). There is an 'Amount of Credit' section with radio buttons for \$5,000, \$10,000 (selected), and \$15,000. A 'Financial Facts' section has checkboxes for 'Own a home', 'Own a boat', and 'Own a car', all of which are checked. Below these are text inputs for 'David, 11' and 'Sarah, 12'. At the bottom are 'Submit Query' and 'Reset' buttons.

Input



The screenshot shows a Mozilla Firefox browser window with the title 'Query Results - Mozilla Firefox'. The address bar shows 'www-scf.u'. The page displays the results of the form submission under the heading 'Query Results'. The results are listed as follows:

```
name = Richard Stark
street = 123 main st
city = brooklyn
state = New York
amount = 10000
home = on
boat = on
car = on
family = David, 11
Sarah, 12
```

Output

For the following slides, see “Translated PHP Examples from Perl”:

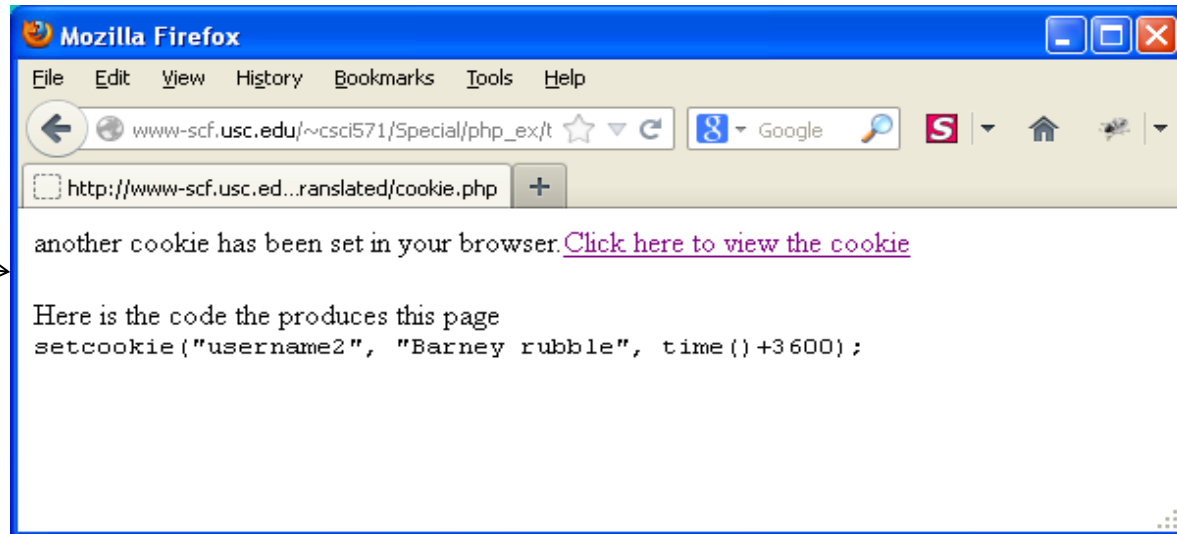
[http://cs-server.usc.edu:45678/examples/php/php\\_ex/translated/index.html](http://cs-server.usc.edu:45678/examples/php/php_ex/translated/index.html)

# Example – Echoing Inputs Scripts

```
<?php if($_GET["submit"]): ?>
<HTML><HEAD><TITLE>Query Results</TITLE></HEAD><BODY>
Query Results<pre>
<?php
foreach($_GET as $key => $value) {
    if ($key !== "submit") {
        echo $key . " = " . $value . "\n";
    }
}
?>
</pre></BODY></HTML>
<?php else: ?>
<HTML><HEAD><TITLE>Sample Form</TITLE></HEAD><BODY>
<H1>Application for a Credit Card</H1>
<FORM METHOD="GET" ACTION="">
<H4>Background Information</H4>
Name <INPUT name=name> Street <INPUT name=street><BR>
City <INPUT name=city> State <SELECT name=state>
<OPTION> Alabama <OPTION> California <OPTION> New York<OPTION> Wisconsin </SELECT>
<H4>Amount of Credit</H4> <INPUT type=radio name=amount value=5000>$5,000
<INPUT type=radio name=amount value=10000>$10,000
<INPUT type=radio name=amount value=15000>$15,000
<H4>Financial Facts:</H4> <INPUT type=checkbox name=home>Own a home
<INPUT type=checkbox name=boat>Own a boat
<INPUT type=checkbox name=car>Own a car<BR>
<TEXTAREA rows=5 cols=50 name=family>
Please describe here the names and ages of people in your family and the number of cards
you are requesting. </TEXTAREA>
<INPUT type=submit name="submit"> <INPUT type=reset></FORM></BODY>
</HTML>
<?php endif; ?>
```

# Example - Setting a Cookie

before



after



# Set / View cookie

Set cookie:

```
<?php
setcookie("username2", "Barney Rubble", time() + 3600 );
?>
```

View cookie:

```
<?php
if(isset($_COOKIE["username2"])) {
    echo "The new cookie <b>username2</b> contains the
    value " . $_COOKIE["username2"];
}
?>
```

# Why PHP Sessions?

- Whenever you want to create a website that allows you to store and display information about a user, determine which user-groups a person belongs to, or utilize permissions on your website, **PHP Sessions** are vital to each of these features.
- PHP has a set of functions that can achieve the same results of Cookies without storing information on the user's computer. **PHP Sessions** store the information on the *web server* in a location that you chose in special files. These files are connected to the user's web browser via the server and a special ID called a "Session ID". This is virtually invisible to the user.

# PHP Sessions

- Sessions store their identifier in a cookie in the client's browser
- Every page that uses session data must be preceded by the `session_start()` function
- Session variables are then set and retrieved by accessing the global `$_SESSION[]`
- Sample session.php code:

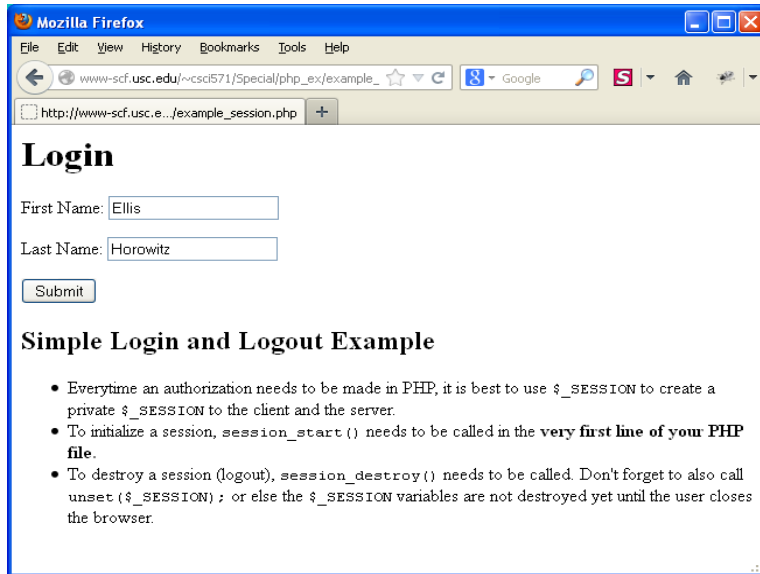
```
<?php
```

```
    session_start();  
    if (!$_SESSION["count"])  
        $_SESSION["count"] = 0;  
    if ($_GET["count"] == "yes")  
        $_SESSION["count"] = $_SESSION["count"] + 1;  
    echo "<h1>".$_SESSION["count"]."</h1>";
```

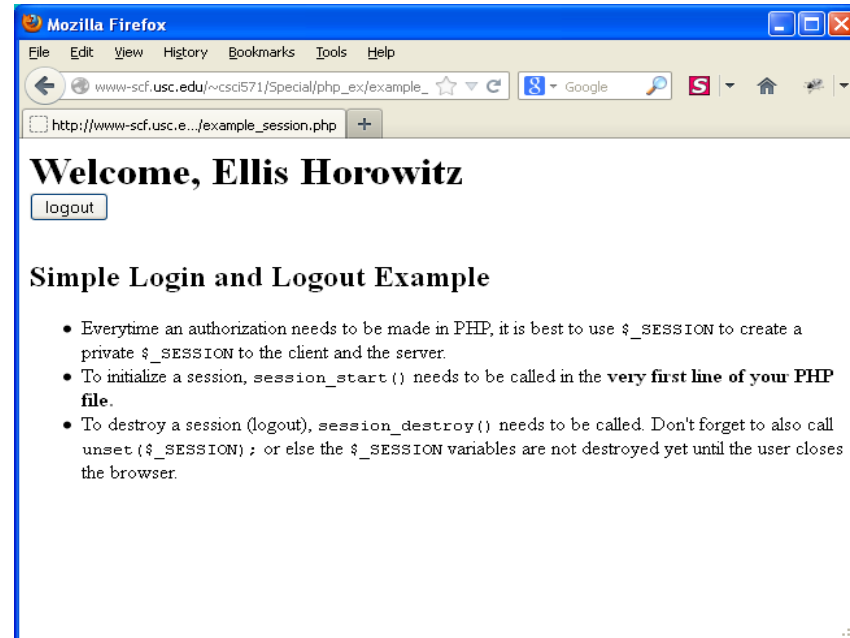
```
?>
```

```
<a href="session.php?count=yes">Click here to count</a>
```

# EXAMPLE – SESSIONS



Initial Login Screen



Resulting Screen

`$_SESSION` creates a private ID; `session_start()` and `session_destroy()`

# example-session.php

```
<?php
    session_start();
    if(isset($_POST["submit"])) {
        $_SESSION["fname"] = $_POST["fname"];
        $_SESSION["lname"] = $_POST["lname"];
    }
    if(isset($_POST["logout"])) {
        session_destroy();
        unset($_SESSION);
    }
?>
<?php if(!(isset($_SESSION["fname"],$_SESSION["lname"]))): ?>
<h1>Login</h1>
<form method="POST">
<p><label for="username">First Name:</label>
    <input type="text" name="fname" />
</p> <p>
    <label for="lname">Last Name:</label>
    <input type="text" name="lname" />
</p> <p><input type="submit" name="submit" value="Submit"></p>
</form>
```



# example-session.php (cont'd)

```
<?php    else:    ?>
<h1>Welcome, <?php echo ucwords($_SESSION["fname"] . " " . $_SESSION["lname"]);
?>
<form method="POST">
    <input type="submit" name="logout" value="logout">
</form>
<?php endif; ?>

<h2>Simple Login and Logout Example</h2>
<ul>
<li>Everytime an authorization needs to be made in PHP, it is best to use
<code>$_SESSION</code> to create a private <code>$_SESSION</code> to the client
    and the server.</li>
<li>To initialize a session, <code>session_start()</code> needs to be
    called in the <b>very first line of your PHP file.</b></li>
<li>To destroy a session (logout), <code>session_destroy()</code> needs
    to be called. Don't forget to also call <code>unset($_SESSION);</code> or else
    the <code>$_SESSION</code> variables are not destroyed yet until the user closes
    the browser.</li>
</ul>
```

Note: The `ucwords()` function converts the first character of each word in a string to uppercase.

# Connecting to a Database

- PHP Has a set of functions that can be used with MySQL
- First step is to setup a link to the desired database
- See “MySQL Functions” docs at:  
<http://php.net/manual/en/ref.mysql.php>
- See MySQL API docs at:  
<http://php.net/manual/en/book.mysql.php>

```
$link = mysql_connect($host, $user, $password);  
Mysql_select_db($database);
```

# Making a Query

- Once a link is established, querying is easy
- Errors for any given function are returned by `mysql_error()`

```
$query = "SELECT * FROM $table;";  
$result = mysql_query($query);
```

# MySQL Result Set

- The value returned by `mysql_query()` is a reference to an internal data structure
- It can be parsed by various functions

```
$row = mysql_fetch_array($result);  
$num_rows = mysql_num_rows($result);  
$affected = mysql_affected_rows($result);
```

# Associative Arrays

- An array that can be indexed by a string
- A set of key->value pairs
- Very similar to a hash in Perl

```
$row['id'] = $id;  
$row = array('id' => $id);  
echo $row['id'];
```

# PHP foreach

- Similar to the Perl equivalent
- Allows iterating through each element of an array

```
foreach($arr as $item) {}  
foreach($row as $key=>$value) {}
```

# Put it All Together

```
<html><body>
<?php
    // setup a query
    $link = mysql_connect($host, $user, $password);
    mysql_select_db($database);
    $query = "SELECT * FROM $table WHERE id = '$id'";
    $result = mysql_query($query);
>
```

## Put it All Together (cont'd)

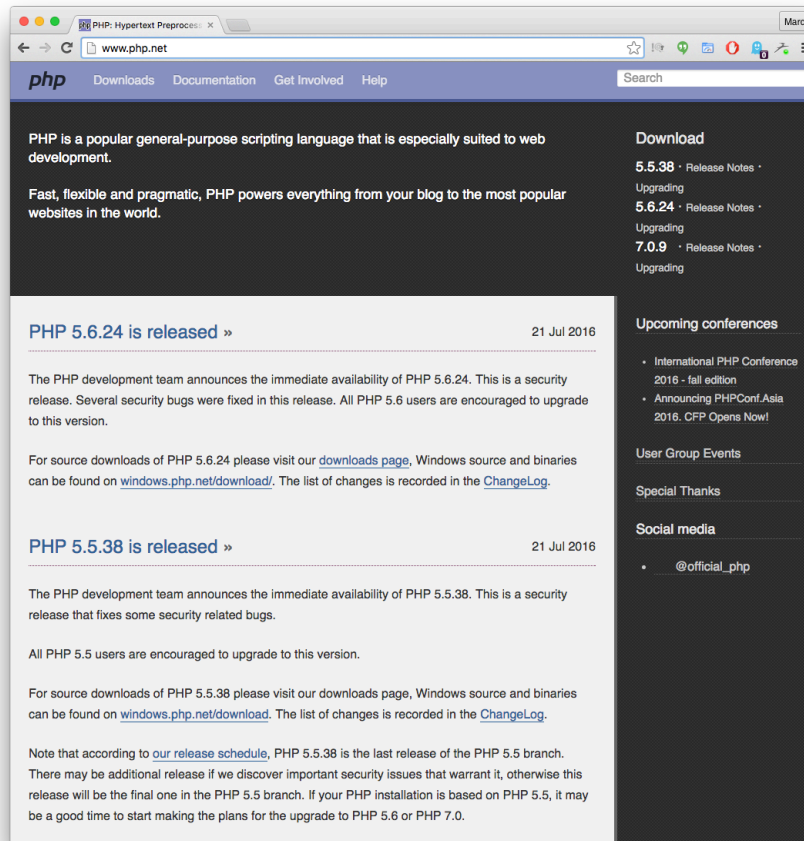
```
//Display a table
echo "<table>";
While ($row = mysql_fetch_array($result)) {
    echo "<tr>";
        foreach ($row as $key => $value) {
            echo "<td>$value</td>";
        }
    echo "</tr>";
}
echo "</table>";
mysql_close($link);
?>
</body></html>
```



# The Results

| <b>ID</b>       | <b>Time</b>         | <b>New Rain</b> | <b>Total Rain</b> |
|-----------------|---------------------|-----------------|-------------------|
| 382422089522600 | 2007-08-13 03:00:00 | 0.19            | 11.76             |
| 382422089522600 | 2007-08-13 03:15:00 | 0.01            | 11.77             |
| 382422089522600 | 2007-08-19 13:30:00 | 0.05            | 11.82             |
| 382422089522600 | 2007-08-19 13:45:00 | 0.01            | 11.83             |
| 382422089522600 | 2007-08-20 13:30:00 | 0.01            | 11.84             |
| 382422089522600 | 2007-08-20 13:45:00 | 0.01            | 11.85             |
| 382422089522600 | 2007-08-20 17:30:00 | 0.01            | 11.86             |
| 382422089522600 | 2007-08-24 13:15:00 | 0.01            | 11.87             |
| 382422089522600 | 2007-08-24 15:00:00 | 0.11            | 11.98             |

# www.php.net



www.php.net is the main destination for PHP. The site includes all source/binary releases of PHP plus tutorials and discussion groups. Another excellent site for PHP is [www.phphelp.com](http://www.phphelp.com).