Introduction to PHP

PHP - What is it

- PHP: **P**HP **H**ypertext **P**re-processor
- Programming language that is interpreted and executed on the server
- PHP is a server-side scripting language
- PHP supports many databases (MySQL, Informix, Oracle, PostgreSQL, Generic ODBC, etc.)
- PHP is open source software
- PHP is free to download and use
- PHP runs on different platforms (Windows, Linux, Unix, etc.)
- PHP is compatible with almost all servers (Apache, IIS, etc.)

PHP - Syntax and Structure

- PHP is similar to C
- All scripts start with **<?php** and end with **?>**
- Line separator: ; (semi-colon)
- Code block: { //code here }
- Comments are created using:

```
// single line quote
# This is also a single-line comment
/* Multiple line block quote */
```

PHP - Variables

- Variables are used for storing values, like text strings, numbers or arrays.
- Variables represented by a dollar ($) sign followed by the name of the variable
- Variable name starts with a letter or underscore

  Example: $author = “Steve Prettyman”;

- No need to define type
- Variable names are case sensitive

  $author and $Author are different

PHP - Example Script

```
<?php
    $author = “Steve Prettyman”;
    $msg = “Hello world!”;
    echo $author , “ says ” , $msg;
?>
```

```
<?php
    $num1 = 1;
    $num2 = 2;
    echo   $num1 + $num2;
 ?>
```

PHP Concatenation

- The concatenation operator (.) is used to put two string values together.

```
<?php
$txt1="Hello World!";
$txt2="What a nice day!";
echo $txt1 . " " . $txt2;
?>
```

```
Hello World! What a nice day!
```

PHP - Operators

- Standard mathematical operators

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

- Basic Boolean comparison with “==”

  Using only = will overwrite a variable value

  Less than < and greater than >

  <= and >= as above but include equality

PHP - Data Types

- A data type is either text or numeric
  -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%”
```

gettype() function returns the type of a variable:

```
$a = 3;
echo gettype($a);       // outputs --  >   integer
```

PHP Conditional Statements

- **if** statement - use this statement to execute some code only if a specified condition is true
- **if...else** statement - use this statement to execute some code if a condition is true and another code if the condition is false
- **if...elseif....else** statement - use this statement to select one of several blocks of code to be executed
- **switch** statement - use this statement to select one of many blocks of code to be executed

PHP Conditional Statements

- The following example will output "Have a nice weekend!" if the current day is Friday:

```
<?php

$day = date("D");
if ($day == "Fri") {
    echo "<p>Have a nice weekend!</p>"; }
?>
```

Get a Date
The required format parameter of the date() function specifies how to format the date (or time).

Here are some characters that are commonly used for dates:
D- represent the short day name (3 letters) like Mon, Tue, Fri.
d - represents the day of the month (01 to 31)
M - represent the short month name (3 letters) Jan
m - represents a month (01 to 12)
Y - represents a year (in four digits)

PHP Conditional Statements

- Use the switch statement to select one of many blocks of code to be executed.

```
switch (n)
{
case label1:
  code to be executed if n=label1;
  break;
case label2:
  code to be executed if n=label2;
  break;
default:
  code to be executed if n is different from both label1 and label2;
}
```

PHP Conditional Statements

```
<?php

$x = 2;
switch ($x) {
    case 1:
        echo "Number 1";
        break;
    case 2:
        echo "Number 2";
        break;
    case 3:
        echo "Number 3";
        break;
    default:
        echo "No number between 1 and 3";
 }
?>
```

PHP For Loop

- A **for loop** is the simplest way to construct a counted loop.

```
for($count=1; $count<=6; $count++ ) {
    echo $count . " times 6 is " . ($count * 6);
    echo "<br>";
}
```

```
1 times 6 is 6
2 times 6 is 12
3 times 6 is 18
4 times 6 is 24
5 times 6 is 30
6 times 6 is 36
```

PHP - embedded language

- Create a file named **hello.php**
- PHP can be placed directly inside HTML: E.g.

```
<html>
    <head><title>Basic PHP page</title></head>
    <body>
      <h1>
        <?php
            echo “Hello World!”;
         ?>
    </h1>
    </body>
</html>
```

echo and print Statements

- With PHP, there are two basic ways to get output: **echo** and **print**.
- echo and print are more or less the same. They are both used to output data to the screen.
- The differences are small: echo can take multiple parameters, while print can take one argument.

echo and print Statements

```
<html>
<body>

<?php
    echo "<h2>PHP is Fun!</h2>";
    echo "Hello world!<br>";
    echo "I'm about to learn PHP!<br>";
    echo "This ", "string ", "was ", "made ", "with multiple
    parameters.";
?>

</body>
</html>
```

echo and print Statements

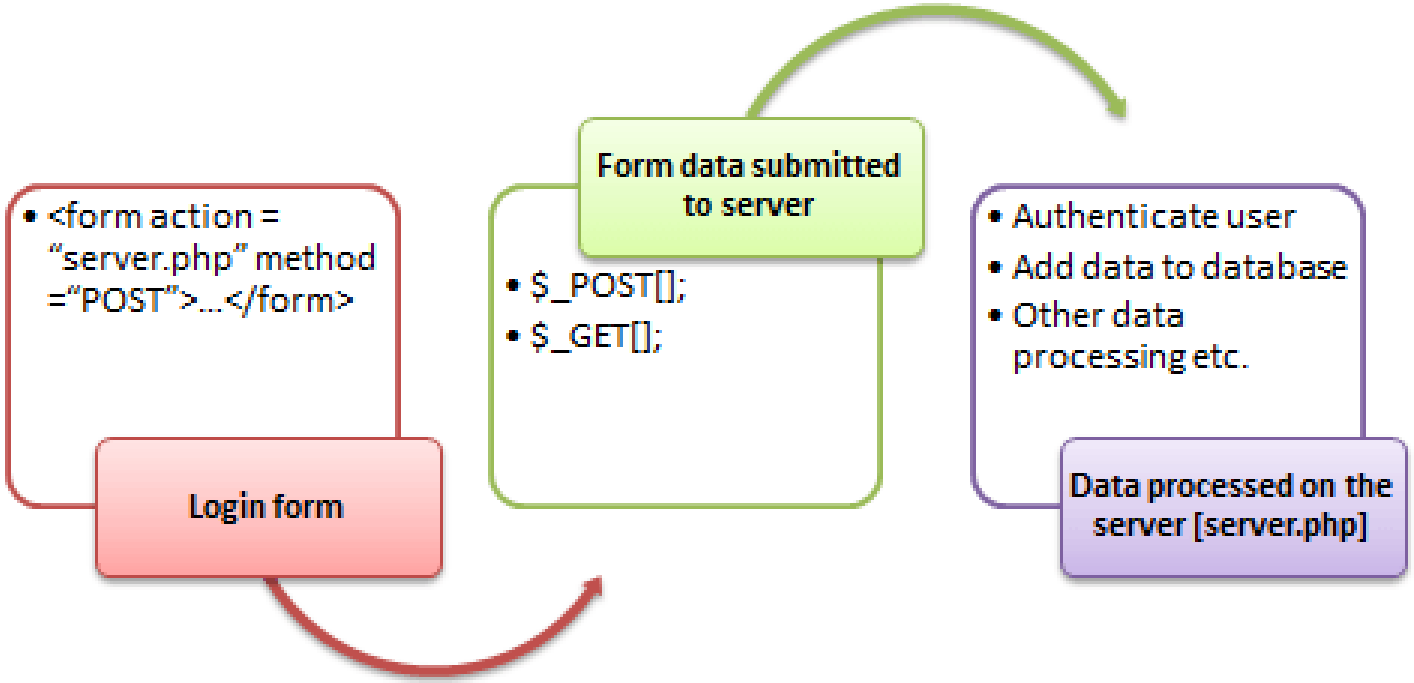
```
<!DOCTYPE html>
<html>
<body>

<?php
	print "<h2>PHP is Fun!</h2>";
	print "Hello world!<br>";
	print "I'm about to learn PHP!";
?>

</body>
</html>
```

PHP and HTML Forms

- Forms are used to get input from the user and submit it to the web server for processing.
- The diagram below illustrates the form handling process.



PHP and HTML Forms

- A form is an HTML tag that contains graphical user interface items such as input box, check boxes radio buttons, etc.
- The form is defined using the <form> … </form> tags

```
<p>Guessing game...</p>
<form method="post">
   <p><label for="guess">Input Guess</label>
   <input type="text" name="guess" id="guess"/></p>
   <input type="submit"/>
</form>
```

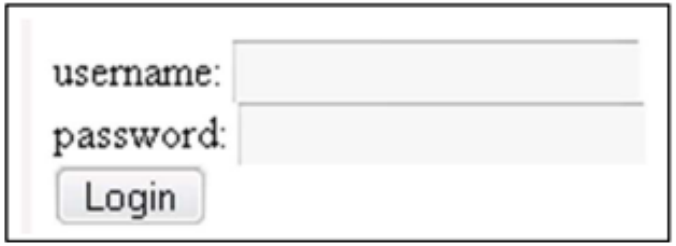
Working with user input

- The user sends data to the server only one way – with HTML Forms

  - They are sets of fields that determine the types of data to be sent.

  - The server receives the filled-in data and sends the HTML response back to the user.

Working with user input



The user enters data and submits the form.
The form has "action" URL to send the data to.

```
<?
echo "Welcome ".$_POST ['username'] ."!";
?>
```

The PHP script receives the data as $_GET and $_POST arrays and runs.

Working with user input

- All form values are placed into an **array**
- Assume a form contains one textbox called “txtName” and the form is submitted using the post method, invoking process.php

```
<form method=“post”   action=“process.php” >
```

- process.php could access the form data using:

```
$_POST[‘txtName’]
```

- If the form used the get method, the form data would be available as:

```
<form method=“get”   action=“process.php” >
$_GET[‘txtName’]
```

$_POST and $_GET

- $_POST and $_GET
  - PHP receives the data in the $_GET and $_POST arrays
    - URL parameters (data from HTML forms with method="GET") go into the $_GET array.
    - Data from HTML forms with method="post" go into the $_POST array.
  - Both arrays are global and can be used anywhere in the requested page.

$_POST

- $_POST is an associative array (key and value pairs)
  – The "name" attribute of form input becomes key in the array

```
<form method="post" action="test.php">
    <input type="text" name="firstname">
    <input type="password" name="pass">
</form>
```

– If in the above form the user fills “Ali" and "mypass“
– test.php will start with built-in array **$_POST**":
•$_POST["firstname"] will be "Ali"
•$_POST["pass"] will be "mypass"

$_GET

- $_GET is also associative array
  – If we open the URL:
  – The test2.php script will start with built-in array $_GET

  $_GET['page'] will be 1
  $_GET['user'] will be “Ali“

http://phpcourse.com/test2.php?page=1&user=Ali

- For example, an HTML form:

```
<form id="showmsg" action="show.php" method="post">
    <input type="text" name="txtMsg" value="Hello World" />
    <input type="submit" id="submit" value="Submit">
</form>
```

- A file called **show.php** would receive the submitted data
- It could output the message, for example:

show.php

```
<h1>
    <?php
        echo $_POST[“txtMsg”];
     ?>
</h1>
```

Hello World

Example

form.htm

```
<html>
<head>
   <title>Simple Form</title>
</head>
<body>
   <h2>Enter Your Information:</h2>
   <form action="process.php" method="post">
      First Name: <input type="text" name="firstname"><br><br>
      Last Name: <input type="text" name="lastname"><br><br>
      Email: <input type="text" name="email"><br><br>
      <input type="submit" value="Submit">
   </form>
</body>
</html>
```

Example

process.php

```
<?php
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $fname = $_POST['firstname'];
    $lname = $_POST['lastname'];
    $email = $_POST['email'];

    echo "<h2>Submitted Information:</h2>";
    echo "First Name: " . $fname . "<br>";
    echo "Last Name: " . $lname . "<br>";
    echo "Email: " . $email;
} else {
    echo "Please submit the form first.";
    echo "<br><a href='form.html'>Go to Form</a>";
}
?>
```