Advanced Web Programming

Advanced Web Programming

- Web technologies Server-side Scripting Concepts
- PHP

PHP

- **What is PHP?**
- PHP stands for **P**HP: **H**ypertext **P**reprocessor
- PHP is a widely-used, open source scripting language
- PHP scripts are executed on the server
- PHP is free to download and use

What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies

What Can PHP Do?

- PHP can add, delete, modify data in your database
- PHP can restrict users to access some pages on your website
- PHP can encrypt data
- With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

Why PHP?

- PHP runs on different platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP has support for a wide range of databases
- PHP is free. Download it from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

What is a PHP File?

- PHP files can contain text, HTML, JavaScript code, and PHP code
- PHP code are executed on the server, and the result is returned to the browser as plain HTML
- PHP files have a default file extension of ".php“
- A PHP script can be placed anywhere in the document.
- A PHP script starts with **<?php** and ends with **?>**:

Hello.PHP

- Basically, a PHP file is a HTML file with some PHP code inside PHP-delimiters:

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

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>
```

Hello.PHP

- Each code line in PHP must end with a semicolon. The semicolon is a separator and is used to distinguish one set of instructions from another.
- With PHP, there are two basic statements to output text in the browser: **echo** and **print**.

Comments

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

<?php
//This is a PHP comment line

/*
This is
a PHP comment
block
*/
?>

</body>
</html>
```

Getting Started

- **What Do I Need?**
  - To start using PHP, you can:
  - Find a web host with PHP and MySQL support
  - Install a web server on your own PC, and then install PHP and MySQL
- **Use a Web Host With PHP Support**
  - If your server has activated support for PHP you do not need to do anything.
  - Just create some .php files, place them in your web directory, and the server will automatically parse them for you.
  - You do not need to compile anything or install any extra tools.
  - Because PHP is free, most web hosts offer PHP support.

Set Up PHP on Your Own PC

- install a web server
- install PHP
- install a database, such as MySQL
- Use XAMP
  - XAMPP is an easy to install Apache distribution containing MySQL, PHP and Perl.
  - It's Free …
  - http://www.apachefriends.org/en/xampp.html

XAMPP and PHP

- Create Hello.php file in the htdocs directory
- Can use Notepad ++ .. Etc
- Call Hello.php on the URL in your web browser
- NB- XAMPP – Apache must be running
- Create a folder in C:\xampp\htdocs
- Copy Hello.php into folder, call on browser, use localhost
- MMServer V's local Machine

Rules for PHP variables:

- PHP has no command for declaring a variable.
- A variable is created the moment you first assign a value to it:
- $txt="Hello world!"; $x=5;
- After the execution of the statements above, the variable **txt** will hold the value **Hello world!**, and the variable **x** will hold the value **5**.

PHP is a Loosely Typed Language

- In the example above, notice that we did not have to tell PHP which data type the variable is.
- PHP automatically converts the variable to the correct data type, depending on its value.
- In a strongly typed programming language, we will have to declare (define) the type and name of the variable before using it.

Global Scope

- A variable that is defined outside of any function, has a global scope.
- Global variables can be accessed from any part of the script, EXCEPT from within a function.
- To access a global variable from within a function, use the **global** keyword:

Global Scope

Example

```
<?php
$x=5; // global scope
$y=10; // global scope

function myTest()
{
global $x,$y;
$y=$x+$y;
}

myTest();
echo $y; // outputs 15
?>
```

Global Scope

- PHP also stores all global variables in an array called $GLOBALS[*index*]. The *index* holds the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.

Example

```
<?php
$x=5;
$y=10;

function myTest()
{
$GLOBALS['y']=$GLOBALS['x']+$GLOBALS['y'];
}

myTest();
echo $y;
?>
```

Local Scope

- The scope is confined to the function

```
<!DOCTYPE html>
<body>
        <h1>Local Scope</h1>
<?php
        function myTest()
        {       $x=0;
                echo $x;
                $x++;
        }
        myTest();
        myTest();
        myTest();
?>
</body>
</html>
```

Static Scope

- When a function is completed, all of its variables are normally deleted. However, sometimes you want a local variable to not be deleted.
- To do this, use the **static** keyword when you first declare the variable:

Example

```
<?php

function myTest()
{
static $x=0;
echo $x;
$x++;
}

myTest();
myTest();
myTest();

?>
```

Parameter Scope

Parameter Scope

A parameter is a local variable whose value is passed to the function by the calling code.

Parameters are declared in a parameter list as part of the function declaration:

Example

```
<?php

function myTest($x)
{
echo $x;
}

myTest(5);

?>
```