PHP

Introduction to PHP

What is PHP?

- PHP stands for **P**HP: **H**ypertext **P**reprocessor
- PHP is a server-side scripting language, like ASP
- PHP scripts are executed on the server
- PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- PHP is an open source software (OSS)
- PHP is free to download and use

What is a PHP File?

- PHP files may contain text, HTML tags and scripts
- PHP files are returned to the browser as plain HTML
- PHP files have a file extension of ".php", ".php3", or ".phtml"

What is MySQL?

- MySQL is a small database server
- MySQL is ideal for small and medium applications
- MySQL supports standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use

PHP + MySQL

- PHP combined with MySQL are cross-platform (means that you can develop in Windows and serve on a Unix platform)

Why PHP?

- PHP runs on different platforms (Windows, Linux, Unix, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP is FREE to download from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

Where to Start?

- Install an Apache server on a Windows or Linux machine
- Install PHP on a Windows or Linux machine
- Install MySQL on a Windows or Linux machine

Basic PHP Syntax

A PHP file normally contains HTML tags, just like an HTML file, and some PHP scripting code.

Below, we have an example of a simple PHP script which sends the text "Hello World" to the browser:

```
<html>
<body>
<?php echo "Hello World"; ?>
</body>
</html>
```

A PHP scripting block always starts with **<?php** and ends with **?>**. A PHP scripting block can be placed anywhere in the document.

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.

There are two basic statements to output text with PHP: **echo** and **print**. In the example above we have used the echo statement to output the text "Hello World".

Variables in PHP

All variables in PHP start with a $ sign symbol. Variables may contain strings, numbers, or arrays.

Below, the PHP script assigns the string "Hello World" to a variable called $txt:

```
<html>
<body>
<?php
$txt="Hello World";
echo $txt;
?>
</body>
</html>
```

To concatenate two or more variables together, use the dot (.) operator:

```
<html>
<body>
<?php
$txt1="Hello World";
$txt2="1234";
echo $txt1 . " " . $txt2 ;
?>
</body>
```

```
</html>
```

The output of the script above will be: "Hello World 1234".

Comments in PHP

In PHP, we use // to make a single-line comment or /* and */ to make a large comment block.

```
<html>
<body>
<?php
//This is a comment
/*
This is
a comment
block
*/
?>
</body>
</html>
```

Operators are used to operate on values.

PHP Operators

This section lists the different operators used in PHP.

Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 x+2	4
-	Subtraction	x=2 5-x	3
*	Multiplication	x=4 x*5	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
!=	is not equal	5!=8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

Logical Operators

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
||	or	x=6 y=3 (x==5 || y==5) returns false
!	not	x=6 y=3 !(x==y) returns true

PHP Conditional Statements

Conditional statements in PHP are used to perform different actions based on different conditions.

Conditional Statements

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

In PHP we have two conditional statements:

- **if (...else) statement** - use this statement if you want to execute a set of code when a condition is true (and another if the condition is not true)
- **switch statement** - use this statement if you want to select one of many sets of lines to execute

The If Statement

If you want to execute some code if a condition is true and another code if a condition is false, use the if....else statement.

Syntax

```
if (condition)
code to be executed if condition is true;
else
code to be executed if condition is false;
```

Example

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

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
echo "Have a nice weekend!";
else
echo "Have a nice day!";
?>
</body>
</html>
```

If more than one line should be executed when a condition is true, the lines should be enclosed within curly braces:

```
<html>
<body>
<?php
$x=10;
if ($x==10)
{
echo "Hello<br />";
echo "Good morning<br />";
}
?>
</body>
</html>
```

The Switch Statement

If you want to select one of many blocks of code to be executed, use the Switch statement.

Syntax

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

Example

This is how it works: First we have a single expression (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use **break** to prevent the code from running into the next case automatically. The default statement is used if none of the cases are true.

```
<html>
<body>
<?php
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";
}
?>
</body>
</html>
```

PHP Looping

Looping statements in PHP are used to execute the same block of code a specified number of times.

Looping

Very often when you write code, you want the same block of code to run a number of times. You can use looping statements in your code to perform this.

In PHP we have the following looping statements:

- **while** - loops through a block of code if and as long as a specified condition is true
- **do...while** - loops through a block of code once, and then repeats the loop as long as a special condition is true
- **for** - loops through a block of code a specified number of times
- **foreach** - loops through a block of code for each element in an array

The while Statement

The while statement will execute a block of code **if and as long as** a condition is true.

Syntax

```
while (condition)
code to be executed;
```

Example

The following example demonstrates a loop that will continue to run as long as the variable i is less than, or equal to 5. i will increase by 1 each time the loop runs:

```
<html>
<body>
<?php
$i=1;
while($i<=5)
{
echo "The number is " . $i . "<br />";
$i++;
}
?>
</body>
</html>
```

The do...while Statement

The do...while statement will execute a block of code **at least once** - it then will repeat the loop **as long as** a condition is true.

Syntax

```
do
{
code to be executed;
}
while (condition);
```

Example

The following example will increment the value of i at least once, and it will continue incrementing the variable i while it has a value of less than 5:

```
<html>
<body>
<?php
$i=0;
do
{
$i++;
echo "The number is " . $i . "<br />";
}
while ($i<5);
?>
</body>
</html>
```

The for Statement

The for statement is used when you know how many times you want to execute a statement or a list of statements.

Syntax

```
for (initialization; condition; increment)
{
code to be executed;
}
```

Note: The for statement has three parameters. The first parameter is for initializing variables, the second parameter holds the condition, and the third parameter contains any increments required to implement the loop. If more than one variable is included in either the initialization or the increment section, then they should be separated by commas. The condition must evaluate to true or false.

Example

The following example prints the text "Hello World!" five times:

```
<html>
<body>
<?php
for ($i=1; $i<=5; $i++)
{
echo "Hello World!<br />";
}
?>
</body>
</html>
```

The foreach Statement

Loops over the array given by the parameter. On each loop, the value of the current element is assigned to $value and the array pointer is advanced by one - so on the next loop, you'll be looking at the next element.

Syntax

```
foreach (array as value)
{
code to be executed;
}
```

Example

The following example demonstrates a loop that will print the values of the given array:

```
<html>
<body>
<?php
$arr=array("one", "two", "three");
foreach ($arr as $value)
{
echo "Value: " . $value . "<br />";
}
?>
</body>
</html>
```

PHP Functions

The real power of PHP comes from its functions. In PHP - there are more than 700 functions available.

PHP Functions

PHP Information

The phpinfo() function is used to output PHP information.

This function is useful for trouble shooting, providing the version of PHP, and how it is configured.

The phpinfo() function options

Name	Description
INFO_GENERAL	The configuration line, php.ini location, build date, Web Server, System and more
INFO_CREDITS	PHP 4 credits
INFO_CONFIGURATION	Local and master values for php directives
INFO_MODULES	Loaded modules
INFO_ENVIRONMENT	Environment variable information
INFO_VARIABLES	All predefined variables from EGPCS (Environment, GET, POST, Cookie, Server)
INFO_LICENSE	PHP license information
INFO_ALL	Shows all of the above. This is the default value

Example

```
<html>
<body>
<?php
// Show all PHP information
phpinfo();
?>
<?php
// Show only the general information
phpinfo(INFO_GENERAL);
?>
</body>
</html>
```

PHP Server Variables

All servers hold information such as which URL the user came from, what's the user's browser, and other information. This information is stored in variables.

In PHP, the $_SERVER is a reserved variable that contains all server information. The $_SERVER is a global variable - which means that it's available in all scopes of a PHP script.

Example

The following example will output which URL the user came from, the user's browser, and the user's IP address:

```
<html>
<body>
<?php
echo "Referer: " . $_SERVER["HTTP_REFERER"] . "<br />";
echo "Browser: " . $_SERVER["HTTP_USER_AGENT"] . "<br />";
echo "User's IP address: " . $_SERVER["REMOTE_ADDR"];
?>
</body>
</html>
```

PHP Header() Function

The header() function is used to send raw HTTP headers over the HTTP protocol.

Note: This function must be called before anything is written to the page!

Example

The following example will redirect the browser to the following URL:

```
<?php
//Redirect browser
header("Location: http://www.w3schools.com/");
?>
<html>
<body>
.....
</body>
</html>
```

Note: This function also takes a second parameter - an optional value of true or false to determine if the header should replace the previous header. Default is TRUE.

However, if you pass in FALSE as the second argument you can FORCE multiple headers of the same type.

Example

```
<?php
header("WWW-Authenticate: Negotiate");
header("WWW-Authenticate: NTLM", FALSE);
?>
<html>
<body>
```

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

PHP More Functions

This section describes file handling in PHP.

Opening a File

The fopen() function is used to open files in PHP.

The first parameter of this function contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened in:

```
<html>
<body>
<?php
$f=fopen("welcome.txt","r");
?>
</body>
</html>
```

The file may be opened in one of the following modes:

File Modes	Description
r	Read only. File pointer at the start of the file
r+	Read/Write. File pointer at the start of the file
w	Write only. Truncates the file (overwriting it). If the file doesn't exist, fopen() will try to create the file
w+	Read/Write. Truncates the file (overwriting it). If the file doesn't exist, fopen() will try to create the file
a	Append. File pointer at the end of the file. If the file doesn't exist, fopen() will try to create the file
a+	Read/Append. File pointer at the end of the file. If the file doesn't exist, fopen() will try to create the file
x	Create and open for write only. File pointer at the beginning of the file. If the file already exists, the fopen() call will fail and generate an error. If the file does not exist, try to create it
x+	Create and open for read/write. File pointer at the beginning of the file. If the file already exists, the fopen() call will fail and generate an error. If the file does not exist, try to create it

Note: If the fopen() function is unable to open the specified file, it returns 0 (false).

Example

The following example generates a message if the fopen() function is unable to open the specified file:

```
<html>
<body>
<?php
if (!($f=fopen("welcome.txt","r")))
exit("Unable to open file!");
?>
</body>
</html>
```

Closing a File

The fclose() function is used to close a file.

```
fclose($f);
```

Reading from a File

The feof() function is used to determine if the end of file is true.

Note: You cannot read from files opened in w, a, and x mode!

```
if (feof($f))
echo "End of file";
```

Reading a Character

The fgetc() function is used to read a single character from a file.

Note: After a call to this function the file pointer has moved to the next character.

Example

The example below reads a file character by character, until the end of file is true:

```
<?php
if (!($f=fopen("welcome.txt","r")))
exit("Unable to open file.");
while (!feof($f))
{
$x=fgetc($f);
echo $x;
}
fclose($f);
?>
```

PHP Forms

A very powerful feature of PHP is the way it handles HTML forms!

PHP Form Handling

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.

Look at the following example of an HTML form:

```
<html>
<body>
<form action="welcome.php" method="POST">
Enter your name: <input type="text" name="name" />
Enter your age: <input type="text" name="age" />
<input type="submit" />
</form>
</body>
</html>
```

The example HTML page above contains two input fields and a submit button. When the user fills in this form and hits the submit button, the "welcome.php" file is called.

The "welcome.php" file looks like this:

```
<html>
<body>
Welcome <?php echo $_POST["name"]; ?>.<br />
You are <?php echo $_POST["age"]; ?> years old!
</body>
</html>
```

A sample output of the above script may be:

```
Welcome John.
You are 28 years old!
```

Here is how it works: The $_POST["name"] and $_POST["age"] variables are automatically set for you by PHP. The $_POST contains all POST data.

Note: If the method attribute of the form is GET, then the form information will be set in $_GET instead of $_POST.

PHP Cookies

A cookie is often used to identify a user.

What is a Cookie?

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests for a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

How to Create a Cookie

The setcookie() function is used to create cookies.

Note: The setcookie() function must appear BEFORE the <html> tag.

Syntax

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

Example

The following example sets a cookie named "uname" - that expires after ten hours.

```
<?php
setcookie("uname", $name, time()+36000);
?>
<html>
<body>
<p>
A cookie was set on this page! The cookie will be active when
the client has sent the cookie back to the server.
</p>
</body>
</html>
```

How to Retrieve a Cookie Value

When a cookie is set, PHP uses the cookie name as a variable.

To access a cookie you just refer to the cookie name as a variable.

Tip: Use the isset() function to find out if a cookie has been set.

Example

The following example tests if the uname cookie has been set, and prints an appropriate message.

```
<html>
<body>
<?php
if (isset($_COOKIE["uname"]))
```

```
echo "Welcome " . $_COOKIE["uname"] . "!<br />";
else
echo "You are not logged in!<br />";
?>
</body>
</html>
```

PHP Include Files (SSI)

Server Side Includes (SSI) are used to create functions, headers, footers, or elements that will be reused on multiple pages.

Server Side Includes

You can insert the content of one file into another file before the server executes it, with the require() function. The require() function is used to create functions, headers, footers, or elements that will be reused on multiple pages.

This can save the developer a considerable amount of time. If all of the pages on your site have a similar header, you can include a single file containing the header into your pages. When the header needs updating, you only update the one page, which is included in all of the pages that use the header.

Example

The following example includes a header that should be used on all pages:

```
<html>
<body>
<?php require("header.htm"); ?>
<p>
Some text
</p>
<p>
Some text
</p>
</body>
</html>
```

PHP The Date() Function

The date() function is used to format a time or a date.

The Date() Function

The date() function is used to format a time or a date.

Syntax

```
string date (date_format[,int timestamp])
```

This function returns a string formatted according to the specified format.

Date Formats

The table below shows the characters that may be used in the format string:

Character	Description
a	"am" or "pm"
A	"AM" or "PM"
B	Swatch Internet time (000-999)
d	Day of the month with a leading zero (01-31)
D	Three characters that represents the day of the week (Mon-Sun)
F	The full name of the month (January-December)
g	The hour in 12-hour format without a leading zero (1-12)
G	The hour in 24-hour format without a leading zero (0-23)
h	The hour in 12-hour format with a leading zero (01-12)
H	The hour in 24-hour format with a leading zero (00-23)
i	The minutes with a leading zero (00-59)
I	"1" if the date is in daylights savings time, otherwise "0"
j	Day of the month without a leading zero (1-31)
l	The full name of the day (Monday-Sunday)
L	"1" if the year is a leap year, otherwise "0"
m	The month as a number, with a leading zero (01-12)
M	Three letters that represents the name of the month (Jan-Dec)
n	The month as a number without a leading zero (1-12)
O	The difference to Greenwich time (GMT) in hours
r	An RFC 822 formatted date (e.g. "Tue, 10 Apr 2005 18:34:07 +0300")
s	The seconds with a leading zero (00-59)
S	The English ordinal suffix for the day of the month (st, nd, rd or th)
t	The number of days in the given month (28-31)
T	The local time zone (e.g. "GMT")
U	The number of seconds since the Unix Epoch (January 1 1970 00:00:00 GMT)
w	The day of the week as a number (0-6, 0=Sunday)
W	ISO-8601 week number of year, weeks starting on Monday
Y	The year as a 4-digit number (e.g. 2003)
y	The year as a 2-digit number (e.g. 03)
z	The day of the year as a number (0-366)

Examples

```
<?php
//Prints something like: Monday
echo date("l");
```

```
//Prints something like: Monday 15th of January 2003 05:51:38 AM
echo date("l dS of F Y h:i:s A");
//Prints something like: Monday the 15th
echo date("l \\t\h\e jS");
?>
```

PHP Database ODBC

ODBC is an Application Programming Interface (API) that allows you to connect to a data source (e.g. an MS Access database).

Create an ODBC Connection

With an ODBC connection, you can connect to any database, on any computer in your network, as long as an ODBC connection is available.

Here is how to create an ODBC connection to a MS Access Database:

1. Open the **Administrative Tools** icon in your Control Panel.
2. Double-click on the **Data Sources (ODBC)** icon inside.
3. Choose the **System DSN** tab.
4. Click on **Add** in the System DSN tab.
5. Select the **Microsoft Access Driver**. Click **Finish.**
6. In the next screen, click **Select** to locate the database.
7. Give the database a **Data Source Name (DSN)**.
8. Click **OK**.

Note that this configuration has to be done on the computer where your web site is located. If you are running Internet Information Server (IIS) on your own computer, the instructions above will work, but if your web site is located on a remote server, you have to have physical access to that server, or ask your web host to to set up a DSN for you to use.

Connecting to an ODBC

The odbc_connect() function is used to connect to an ODBC data source. The function takes four parameters: the data source name, username, password, and an optional cursor type.

The odbc_exec() function is used to execute an SQL statement.

Example

The following example creates a connection to a DSN called northwind, with no username and no password. It then creates an SQL and executes it:

```
$conn=odbc_connect('northwind',",");
```

```
$sql="SELECT * FROM customers";
$rs=odbc_exec($conn,$sql);
```

Retrieving Records

The odbc_fetch_rows() function is used to return records from the result-set. This function returns true if it is able to return rows, otherwise false.

The function takes two parameters: the ODBC result identifier and an optional row number:

```
odbc_fetch_row($rs)
```

Retrieving Fields from a Record

The odbc_result() function is used to read fields from a record. This function takes two parameters: the ODBC result identifier and a field number or name.

The code line below returns the value of the first field from the record:

```
$compname=odbc_result($rs,1);
```

The code line below returns the value of a field called "CompanyName":

```
$compname=odbc_result($rs,"CompanyName");
```

Closing an ODBC Connection

The odbc_close() function is used to close an ODBC connection.

```
odbc_close($conn);
```

An ODBC Example

The following example shows how to first create a database connection, then a result-set, and then display the data in an HTML table.

```
<html>
<body>
<?php
$conn=odbc_connect('northwind',",");
if (!$conn)
{
exit("Connection Failed: " . $conn);
}
$sql="SELECT * FROM customers";
```

```
$rs=odbc_exec($conn,$sql);
if (!$rs)
{
exit("Error in SQL");
}
echo "<table><tr>";
echo "<th>Companyname</th>";
echo "<th>Contactname</th></tr>";
while (odbc_fetch_row($rs))
{
$compname=odbc_result($rs,"CompanyName");
$conname=odbc_result($rs,"ContactName");
echo "<tr><td>$compname</td>";
echo "<td>$conname</td></tr>";
}
odbc_close($conn);
echo "</table>";
?>
</body>
</html>
```

@@@@@