CHAPTER 1

Using PHP

To start developing in PHP, create a plain text file with a .php file extension and open it in the editor of your choice – for example Notepad, JEdit, Dreamweaver, Netbeans or PHPEclipse. This PHP file can include any HTML, as well as PHP scripting code. Begin by first entering the following standard HTML elements into the document.

```
<html>
 <head><title>PHP Test</title></head>
 <body></body>
</html>
```

Embedding PHP

PHP code can be embedded anywhere in a web document in one of four different ways. The standard notation is to delimit the code by "`<?php`" and "`?>`". This is called a PHP code block, or just a PHP block.

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

Within a PHP block the engine is said to be in PHP mode, and outside of the block the engine is in HTML mode. In PHP mode everything will be parsed (executed) by the PHP engine, whereas in HTML mode everything will be sent to the generated web page without any execution.

The second notation for switching to PHP mode is a short version of the first where the "php" part is left out. Although this notation is shorter, the longer one is preferable if the PHP code needs to be portable. This is because support for the short delimiter can be disabled in the php.ini configuration file.[1]

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

[1] http://www.php.net/manual/en/configuration.file.php

A third alternative is to embed the PHP code within a HTML script element with the language attribute set to "php". This alternative is always available, but seldom used.

```
<script language="php">...</script>
```

One remaining style you may encounter is when the script is embedded between ASP tags. This style is disabled by default, but can be enabled from the PHP configuration file.

```
<% ... %>
```

The last closing tag in a script file may be omitted if the file ends in PHP mode.

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

Outputting text

Printing text in PHP is done by either typing `echo` or `print` followed by the output. Each statement must end with a semicolon (`;`) in order to separate it from other statements. The semicolon for the last statement in a PHP block is optional.

```
<?php
  echo  "Hello World";
  print "Hello World"
?>
```

Output can also be generated using the "`<?=`" open delimiter. As of PHP 5.4 this syntax is valid, even if the short PHP delimiter is disabled.

```
<?= "Hello World" ?>
<? echo "Hello World" ?>
```

Keep in mind that text output will only be visible on the web page if it is located within the HTML body element.

```
<html>
 <head><title>PHP Test</title></head>
<body>
 <?php echo "Hello World"; ?>
</body>
</html>
```

Installing a web server

To view PHP code in a browser the code first has to be parsed on a web server with the PHP module installed. An easy way to set up a PHP environment is to download and install a distribution of the popular Apache web server called XAMPP,[2] which comes pre-installed with PHP, Perl and MySQL. This will allow you to experiment with PHP on your own computer.

After installing the web server point your browser to "http://localhost" to make sure that the server is online. It should display the file index.php, which by default is located under "C:\xampp\htdocs\index.php" on a Windows machine. Htdocs is the folder that the Apache web server looks to for files to serve on your domain.

Hello world

Continuing from before, the simple "Hello World" PHP web document should look like this.

```
<html>
 <head><title>PHP Test</title></head>
 <body>
  <?php echo "Hello World"; ?>
 </body>
</html>
```

To view this PHP file parsed into HTML, save it to the web server's htdocs folder (the server's root directory) with a name such as "mypage.php". Then point your browser to its path, which is "http://localhost/mypage.php" for a local web server.

When a request is made for the PHP web page the script is parsed on the server and sent to the browser as only HTML. If the source code for the website is viewed it will not show any of the server-side code that generated the page, only the HTML output.

Compile and parse

PHP is an interpreted language, not a compiled language. Every time a visitor arrives at a PHP website the PHP engine compiles the code and parses it into HTML which is then sent to the visitor. The main advantage of this is that the code can be changed easily without having to recompile and redeploy the website. The main disadvantage is that compiling the code at run-time requires more server resources.

For a small website a lack of server resources is seldom an issue. The time it takes to compile the PHP script is also miniscule compared with other factors, such as the time required to execute database queries. However, for a large web application with lots of traffic the server load from compiling PHP files is likely to be significant. For such a site the script compilation overhead can be removed by precompiling the PHP code.

[2] http://www.apachefriends.org/en/xampp.html

This can be done for example with eAccelerator,[3] which caches PHP scripts in their compiled state.

A website that only serves static content (same to all visitors) has another possibility, which is to cache the fully generated HTML pages. This provides all the maintenance benefits of having a dynamic site, with the speed of a static site. One such caching tool is the W3 Total Cache[4] plugin for the WordPress CMS.

Comments

Comments are used to insert notes into the code and will have no effect on the parsing of the script. PHP has the two standard C++ notations for single-line (`//`) and multi-line (`/* */`) comments. The Perl comment notation (#) may also be used to make single-line comments.

```
<?php
  // single-line comment
  #  single-line comment
  /* multi-line
     comment */
?>
```

As in HTML, whitespace characters – such as spaces, tabs and comments – are generally ignored by the PHP engine. This allows you a lot of freedom in how to format your code.

[3] http://www.eaccelerator.net
[4] http://wordpress.org/extend/plugins/w3-total-cache