

CT216 Software Engineering Tutorial

Eoin O Fiachain

7th/14th October 2004

1 Web Programming

1.1 Scripting Languages

A *script* is a name typically used for programs written in a *scripting language*. A *scripting language* is a language designed for rapid development by connecting existing components to accomplish a new related task.

Programming languages like C are very low-level and cumbersome to write but provide a high-degree of run-time efficiency. Scripting languages tend to sacrifice run-time efficiency for speed of development.

They tend to be *interpreted*. This means that a special program known as an *interpreter* is required to process the program as it is run.

This contrasts with *compiled* programs (such as those written in the C language) which are translated by a *compiler* into machine-executable code and can run independently.

Scripting languages can be written for a specific domain (such as system administration) or for general-purpose usage.

1.2 Client-Side Scripting

HTML pages contain both data and markup tags describing how to display the data. The browser renders this HTML data to present a view of the web page to the user. Typically this view will not change over time and a web page will remain constant until the user navigates to a new page.

By employing additional programs called *client-side scripts* in a HTML page, greater flexibility is achieved as content can vary in accordance to environmental conditions, time, browser type, operating system type etc.

The web browser can execute these programs in response to particular user events (e.g. when the user moves their mouse over an image, when an image loads etc.) and the view of a web page presented to the user can be changed appropriately.

Not all web browsers support client-side scripting languages. So when using a client-side scripting technology the developer often must strike a balance between browser accessibility and the increased flexibility that client-side scripting enables.

On many web-pages, client-side scripting is used only to provide optional enhancements while the core content is restricted to HTML so that is visible on all web browsers.

The two most common client-side scripting languages are:

- **JavaScript** is an object-based scripting language originally developed by Netscape. It is the most commonly used technology for client-side scripting.

Although it has a similar syntax to Sun Microsystem's Java programming language it is a completely separate technology. Microsoft's version of JavaScript is also known as *JScript*.

It is supported by a variety of web browsers including Microsoft's Internet Explorer and the Mozilla/Netscape suite of web browsers. There are a number of JavaScript standards but different browsers often implement elements of them differently.

- **VBScript** is a client-side scripting language developed by Microsoft. It has a syntax similar to Visual Basic but is only supported by Microsoft's Internet Explorer web browser.

JavaScript can either be embedded within a HTML file or stored in a separate file that the HTML page references. The following example demonstrates the embedded approach:

```

<HEAD>
<TITLE>Some Page</TITLE>

<SCRIPT language="JavaScript">

function new_win() {
    window.open('somefile.html','wind','width=600,height=300');
}

</SCRIPT>

</HEAD>

```

A standard called DOM (Document-Object-Model) is often used by web browsers as a way of representing the HTML data in a structured manner. This standard along with CSS (Cascade Style Sheeting), HTML and JavaScript are often collectively known as *DHTML* (Dynamic HTML).

1.2.1 Client-Side Components

Web pages can also reference objects in a HTML which can be executed when the page is run by the web-browser or a special addition to the web browser known as a *plug-in*. These are known as *components* and would not typically be tightly coupled with the HTML page structure.

Common components include:

- **Java Applets** are programs written in Sun Microsystem's Java language that are designed to be executed within a constrained browser-type environment. Java is a very powerful language that enables very complex applications to be enabled. Java Applet plugins are available for both Microsoft's Internet Explorer and the Mozilla set of browsers.
- **ActiveX Components** are programs written for the Win32/COM (Microsoft Windows) platform that can be designed to be executed in a browser-type environment. They are currently only supported by Microsoft's Internet Explorer.
- **Macromedia Flash** is a technology that enables special types of interactive graphical presentations to be included in web pages. A special Macromedia Flash plugin is available for most web browsers.

1.3 Server-Side Scripting

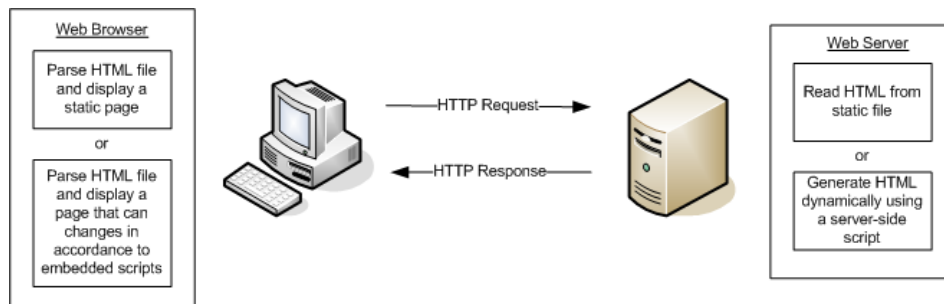


Figure 1: HTTP Request on World Wide Web

Basic web-pages are composed of static HTML files. These are files that contain data and HTML markup that does not typically change unless a user explicitly alters a file's content.

Enterprise web-sites are often quite large and often involve data that varies in response to user input i.e. which user is logged in, what options he selected in a particular form. Rather than always returning a static HTML file regardless of user-input, these web-sites need a flexible approach that can dynamically generate the content of web-pages automatically.

In *server-side scripting*, programs are executed by the web server after receiving a HTTP request. The output of these programs forms the content of the HTTP response that is returned to the web browser.

In *client-side scripting* programs are executed by the web browser after the client receives the HTTP response and HTML page from the server.

Other advantages of using server-side scripting are that repetition of HTML code can be avoided. Content that is used in many web pages can be stored in a single file that is referenced in other pages.

Also, a database server can be employed in order to provide more efficient and flexible data storage. A server-side script queries the database server for the appropriate data and embeds it within HTML code before it is passed back to the client.

1.3.1 Server-Side Execution

A server-side programming language is typically executed in one of two different contexts:

- By executing a separate program through **CGI** (Common Gateway Interface). CGI is a defined standard whereby a web server can start a new process (program), pass parameters to it and retrieve the output of the program. It is language-neutral but it is most commonly used with the PERL language. CGI can be used with either compiled languages (such as C or C++) or with interpreted languages (such as PHP, PERL or Python).
- By executing through a **Web-Server Module**. This module will execute within the web server process. This is significantly more efficient than CGI as there is typically a large CPU overhead associated with each occasion that a new process is started.

Web-Server modules are available for most commonly used server-side scripting languages for the Apache and IIS web servers. The ISAPI standard enables custom modules to be written for IIS.

We shall use the Apache web server with PHP executing as a web-server module for the CT216 Software Engineering Project.

1.3.2 Commonly-Used Server-Side Scripting Languages

- **PERL** is a popular interpreted language created by Larry Wall. Although not designed exclusively for server-side scripting, it provides an effective environment for web development nonetheless through a rich language feature set and extensive library of add-on modules. PERL supports both procedural and object-oriented development.
It is typically used in conjunction with CGI, but can also be used as an Apache module through *mod-perl*.
- **PHP** is an interpreted object-oriented language explicitly designed for server-side web development. Its language syntax is a hybrid between PERL and C. We will examine PHP in greater detail later in the tutorial. PHP supports both procedural and object-oriented development.
- **ASP** is a popular Microsoft technology for server-side scripting on its IIS web server. It can be used with a number of languages including VBScript and JavaScript. Often it can be used in conjunction with compiled ActiveX/COM server-side components.
- **Python** is another object-oriented interpreted language that is used for server-side scripting. Sometimes it used in conjunction with the *Zope* web application server.

1.3.3 Other Popular Server-Side Web Technologies

There are other server-side web technologies designed for larger-scale enterprise web sites. They have moved beyond the constraints of scripting languages to use other languages that enable a stronger-set of object-oriented features, greater run-time efficiency and better integration with other enterprise components.

- **Java Servlets** is a Sun Microsystem's technology that employs the Java programming language and forms part of Sun's JEEE enterprise development environment. Servlets are compiled programs than perform a particular web-site function. Java Servlets are often used together with JSP, which is a complementary declarative language that enables more rapid development.
- **ASP.NET** is a Microsoft technology which combines the features of traditional ASP server-side scripting with Microsoft's .NET framework. ASP.NET programs can be written in any .NET enabled language including VB, C#, C++ and J# (Microsoft's Java dialect).

2 PHP

PHP is a widely-used open source programming language designed primarily for server-side web programming and developing dynamic web content.

A basic PHP program that prints 5 numbered lines to the screen follows:

```
<html>
<head>
<title>Example Program that prints 5 numbered lines</title>
</head>
<body>
<?php
    for ($i=1; $i<=5; $i++)
    {
        echo "Line $i<br>\n";
    }
?>
</body>
</html>
```

The output of this program would be:

```
<html>
<head>
<title>Example Program that prints 5 numbered lines</title>
</head>
<body>
Line 1<br>
Line 2<br>
Line 3<br>
Line 4<br>
Line 5<br>
</body>
</html>
```

2.1 Escaping from HTML

In the preceding example PHP code is embedded within normal HTML markup by the `<?php` and `?>` special tags. There are a number of differ-

ent ways to escape from HTML markup into PHP scripts.

```
<?php echo("Method A"); ?>

<? echo ("Method B"); ?>

<script language="php">
    echo ("Method C");
</script>

<% echo ("Method D (ASP-style tags)"); %>
```

The first method `<?php ... ?>` is the preferred method. The latter two methods may not always be available as they can be enabled or disabled in the PHP configuration file.

The *echo* statement simply writes text to the output. Note that a shortened method for writing `<? echo someexpression; ?>` is `<?=someexpression?>`.

Programmers can escape to and from HTML throughout their PHP code. Generally HTML-mode is preferable where there is a lot of static HTML content, and using *echo* statements is preferable when there is a lot of programming content.

2.2 Types

Like in the C language, PHP variables can belong to a number of different types. The four basic types are:

- **boolean** - TRUE or FALSE
- **integer** - whole number
- **float** - number with decimal points
- **string** - collection of characters

A variable can also belong to a compound type:

- **object** - an object when using Object Oriented Programming
- **array** - similar but slightly different to arrays in C programming.

2.2.1 Arrays

Whereas in C programming an array can only map integers (or indices) to data values, in PHP an array can also map strings to data values.

```
<?php
    $arr = array("foo" => "bar", 12 => true);

    echo $arr["foo"]; // bar
    echo $arr[12];    // 1
?>
```

Variables can also be set to the special *NULL* type which means that they have no value.

2.3 Variables

2.3.1 Variable Naming

Variables in PHP always begin with the \$ character. The rules for naming a variable in PHP are similar to that in C programming. An alphabetic character or underscore must follow the \$ character. After this any combination of alphabetic characters, number characters or underscores may follow.

```
<?php
$CT216 = 67.5;           // Valid
$ct216 = 78;            // Valid
$_ct216 = "tutorial";  // Valid
$216ct = 5;            // Invalid
?>
```

Variable names are case-sensitive so \$CT216 and \$ct216 are considered separate variable names.

2.3.2 Dynamically-Typed

In C programming you must declare a variable with a type before you reference it. This association between a variable and a type at compile-time makes C a *statically-typed* language.

```
int a; /* Variable Declaration */
a = 5; /* Variable Assignment */
```

PHP is a *dynamically-typed* language which means that the type of a variable is not determined until the program is run. This means that you do not have to associate a particular type with a variable when you are writing PHP code. At run-time, the PHP interpreter will determine an appropriate type for the variable based on the value assigned to it.

```
// PHP automatically assigns the string data type at run-time
$var = "NUIG";
```

This automatic type determination can speed-up the development process by reducing code size but it places an extra responsibility on developers to remember the types associated with each variable.

2.3.3 Constants

Constants can also be defined in PHP for the basic data types.

```
<?php
// Valid constant names
define("FOO", "something");
echo FOO; // Outputs "something"
?>
```

Naming rules are similar to variables except that the \$ character is omitted.

2.3.4 Global Scope

Similar to C programming, each variable in PHP has a scope. This is the context in which the variable is defined. This is generally either *local-scope* where the variable is defined within a particular function or *global-scope* where the variable is defined for all functions.

```

<?php
$a = 1; /* global scope */

function Test() {
    $b = 3; /* local scope */
}

Test();
?>

```

Unlike C, functions cannot automatically reference global variables. For example, in the following script the function `Test()` references a variable `$a`. `$a` is considered to be a local-variable called `$a` and not the global variable called `$a`.

```

<?php
$a = 1; /* global scope */

function Test() {
    echo $a; /* reference to local scope variable */
}

Test();
?>

```

The *global* keyword can be used within a function to declare a variable as having global-scope.

```

<?php
$a = 1; /* global scope */

function Test() {
    global $a;
    echo $a; /* reference to global scope variable */
}

Test();
?>

```

Some special predefined PHP global variables automatically have global-scope without using the *global* keyword. These are called *superglobals*.

One such variable is the \$GLOBALS variable which is an array of all the global variables. It provides an alternative means of accessing a global variable \$a from within a function.

```
<?php
$a = 1; /* global scope */

function Test() {
    echo $GLOBALS["a"]; /* reference to global scope variable */
}

Test();
?>
```

2.3.5 Strings

Strings in PHP are similar to strings in C in that they represent a collection of characters. An important difference is that can be expressed using either double-quotes or single-quotes. String contents are treated slightly differently depending on which type of quotes are used.

If expressed using single-quotes then no escape characters will be recognised and variable-parsing will not be employed.

If expressed using double-quotes then escape characters (such as \n and \\) will be recognised in a similar manner to C.

When using double-quotes variable parsing will also be employed. This provides for the automatic translation of a variable name within a string into the variable's contents.

```
<?php
$animal = 'walrus';
$title = "I am a $animal";
echo $title;          // Outputs "I am a walrus"
?>
```

2.4 Language Constructs

2.4.1 Operators

Operators in PHP are mostly similar to the operators used in C programming. Included are arithmetic operators (+ - * / %), bitwise operators (& | ^ ~ << >>), comparison operators (== === != <> !== < > <= >=) assignment operators (= += -= *= /= .= %= &= |= ^= <<= >>=).

The . operator (*string concatenation operator*) allows two strings to be joined together).

Similarly the .= (*string concatenating assignment operator*) is an operator used to allow a string to be concatenated with an existing string.

```
<?php
$a = "Hello ";
$b = $a . "World!"; // now $b contains "Hello World!"

$a = "Hello ";
$a .= "World!";    // now $a contains "Hello World!"
?>
```

The === operator checks that two variables are both equal and of the same type.

```
<?php

$c = 3;
$d = "3";

// this will be true
if ($c==$d) { echo "true"; } else { echo "false"; }

// this will be false
if ($c=== $d) { echo "true"; } else { echo "false"; }

?>
```

2.4.2 Control Structures

The control structures in PHP are mostly similar to those involved in C programming.

```
<?php

// If construct
if ($a > $b) {
    echo "a is bigger than b";
} elseif ($a==$b) {
    echo "a is equal to b";
} else {
    echo "a is smaller than b";
}

// While loop
$i = 1;
while ($i <= 10) {
    echo $i++;
}

// Do-while loop
$i = 0;
do {
    echo $i;
} while ($i > 0);

// For loop
for ($i = 1; $i <= 10; $i++) {
    echo $i;
}

?>
```

The *foreach* construct allows for iteration through an array of values without needing to reference indices.

```

<?php

$a = array(
    "one" => 1,
    "two" => 2,
    "three" => 3,
    "seventeen" => 17
);

foreach ($a as $k => $v) {
    echo "\$a[$k] => $v.\n";
}

// Output would be a series of lines:
//     $a[one] => 1.
//     $a[two] => 2.
//     $a[three] => 3.
//     $a[seventeen] => 17.

?>

```

The *include* and *require* constructs include and evaluate another PHP file. In the event of a file access failure, the *include* statement continues on execution whereas the *require* construct will throw a fatal error.

```

<?php
    include 'some_header_file.php';
    require 'some_absolutely_necessary_header_file.php';
?>

```

The *include_once* and *require_once* constructs provide similar functionality while ensuring that no file gets included more than once in a page.

2.4.3 Functions

Functions can be defined in a similar manner to C, except without any data types being necessary. Every function is also preceded by the *function* keyword.

```
<?php

function foo($arg_1, $arg_2) {
    echo "Example function.\n";
    return $retval;
}

?>
```

2.5 Form Processing

2.5.1 HTML Forms

A form in HTML is a means of allowing a user to submit data to a web server. They typically involve a series of user input controls (text boxes, drop down lists etc.) and a submit button.

When the submit button is clicked, the form's data is submitted to a particular URL (specified with the *action* attribute) by either the *GET* method or the *POST* method.

When the *GET* method is used then the data is specified in the query string of the URL that the form is being submitted to. When the *POST* method is used the data is submitted independently of the query string.

A sample HTML form follows which uses the *POST* method to submit to a PHP script identified by *action.php*. It contains two text boxes allowing the user to submit a name and an age, and a submit button.

```
<form action="action.php" method="post">
  <p>Your name: <input type="text" name="name" /></p>
  <p>Your age: <input type="text" name="age" /></p>
  <p><input type="submit" /></p>
</form>
```

2.5.2 When to use GET and POST

Whether to use *GET* or *POST* depends on circumstance.

Typically *POST* would be used for submitting once-off data such as a credit card purchase.

GET would be used in circumstances where the user may refer back to the submitted page again in the future. An example would be after a form is submitted in the google search engine and the results are displayed. As the form parameters are stored in the query string of the results page the URL of the results page can be easily stored for future reference.

<http://www.google.ie/search?hl=en&ie=UTF-8&q=ct216>

Another factor is the maximum size of data allowed to be submitted. There are more severe size limitations associated with *GET* than *POST*.

2.5.3 PHP Form Processing

After a form has been submitted to a PHP script the data elements of the form can be accessed through a number of superglobal arrays. `$_POST` contains the data elements submitted by *POST*. `$_GET` contains the data elements submitted by *GET*. `$_REQUEST` combines both arrays to contain data elements submitted by either method.

For example to retrieve the name and address submitted on the previous form:

```
<?php
    echo "Name is " . $_POST['name'] . "\n";
    echo "Age is " . $_POST['age'] . "\n";
?>
```

2.6 PHP API

PHP contains a large amount of built-in functions to assist programmers in common tasks. Additional functions can be added by installing extra modules for particular tasks.

The PHP Manual is a comprehensive official guide to the available functions (called an API). It also includes a beginners tutorial and a language reference.

It is absolutely essential resource for any PHP developer. An English language version of the manual is available at:

<http://www.php.net/manual/en/>