Dynamic Web Development

Lecture 8 – Introduction to PHP 1

PHP

PHP is a programming language.

PHP stands for 'PHP Hypertext Preprocessor'

It is one of the most common server-side scripting languages.

It enables you to create dynamic web pages.

One of the jobs that it can be used for is to enable a website to access a database.

Static Web Pages

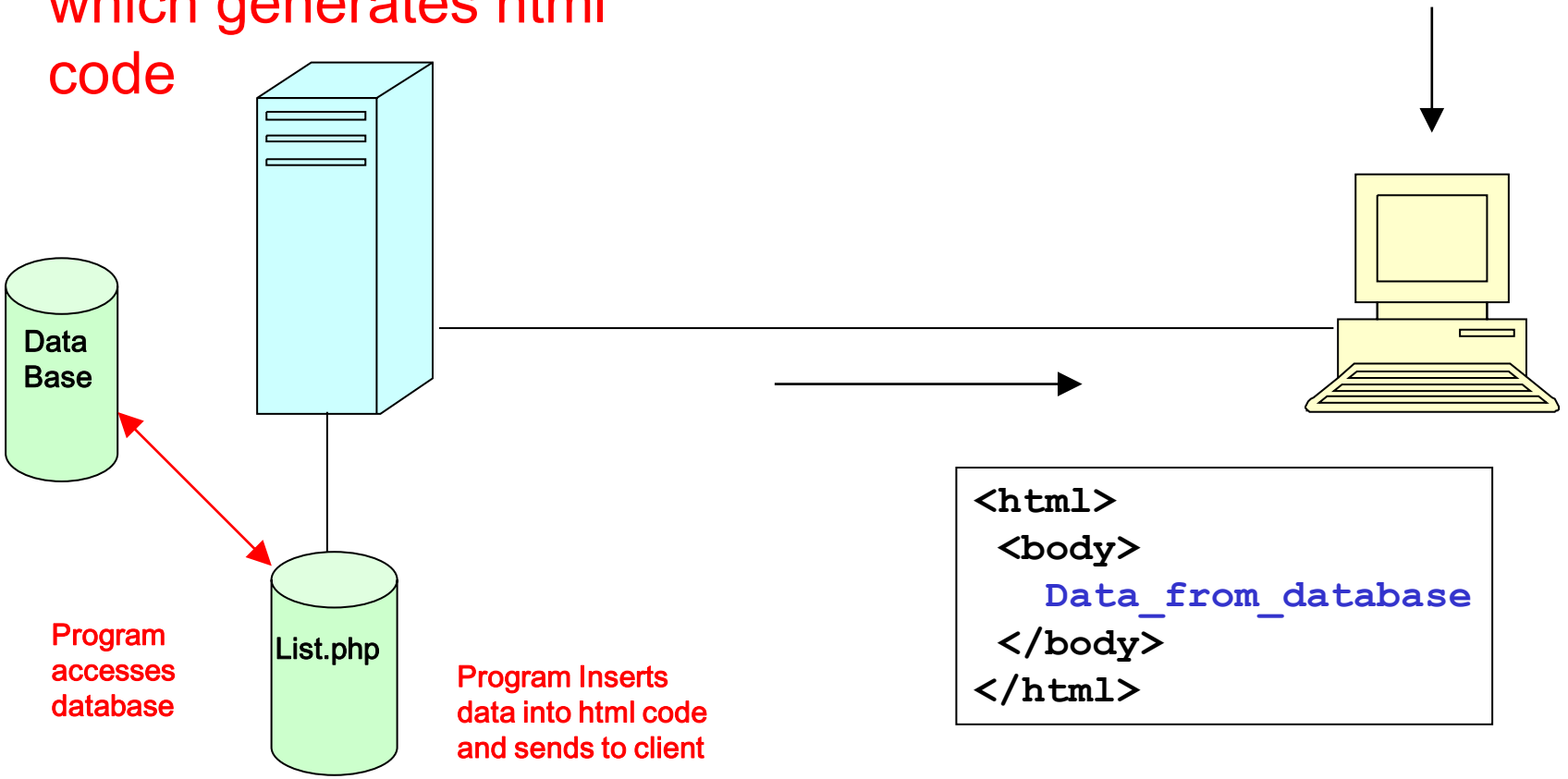
File on server contains fixed html code.

http://www.bpc.ac.uk/staff/kevin.htm

Dynamic Web Pages

File on server contains a program which generates html code

http://www.bpc.ac.uk/student/list.php




```
<html>
 <body>
   Data_from_database
 </body>
</html>
```

History

Server side scripting used to be done by writing programs in languages like C or Perl

1995 Rasmus Lerdorf created a set of Perl scripting tools called 'Personal Home Page'. Designed to be embedded in HTML files. He releases v2 in 1997.

1998 PHP v3 released. Zeev Suraski and Andi Gutmans completely rewrote the PHP interpreter and made it more extendable. PHP installed on 250,000 website domains.

2000 PHP v4 released. Limited Object Oriented facilities. Suraski and Gutmans rewrite it again and modularise the interpreter into the Zend engine. They set up Zend technologies company to promote use of PHP in corporate environment. PHP installed on 2,500,000 website domains.

2005 PHP v5 released. Improved object oriented features and XML and database interfaces. The reference implementation of PHP is now produced by The PHP Group – there is no formal specification. It is still an open source project.

Mixing HTML with PHP

```
<?php $idcode = "Kevin"; ?>
<html>
<head>
<title>My webpage</title>
</head>
<body>
<p>hello there </p>
<?php echo $idcode; ?>
<p> this is a webpage</p>
<p>with some text</p>
</body>
</html>
```

OR

```
<?php
$idcode = "Kevin";
echo"<html>";
echo "<head>";
echo "<title>My webpage</title>";
echo "</head>";
echo "<body>";
echo "<p>hello there </p>";
echo $idcode;
echo "<p> this is a webpage</p>";
echo "<p>with some text</p>";
echo "</body>";
echo "</html>";
?>
```

Either way, this is sent to the browser

```
<html>
<head>
<title>My webpage</title>
</head>
<body>
<p>hello there </p>
Kevin
<p> this is a webpage</p>
<p>with some text</p>
</body>
</html>
```

And what appears in the browser

hello there

Kevin

this is a webpage

with some text

PHP

PHP is a fully featured programming languages. It has:

- variables
- loops
- if…then…else
- functions

and most of the other features that are shared by all procedural programming languages.

It is very similar to C and Java – some of the syntax is identical.

The www.w3schools.com website has excellent tutorials on PHP, and also some good reference pages on the many built-in functions.

Basic PHP Syntax

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

```
<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 "Kevin".

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>
```

Concatenation

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

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

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>
```

PHP Operators - Assignment

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

PHP Operators - Arithmetic

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

PHP Operators – Comparison 1

Operator		Example
==	is equal to	$x == $y
!=	is not equal	$x != 8
>	is greater than	$x > 8
<	is less than	$y < 8

PHP Operators – Comparison 2

>=	is greater than or equal to	5 >= 8 returns false
<=	is less than or equal to	5 <= 8 returns true
===	equal to AND data types match	4 === "4" returns false
!==	not equal OR data types not the same	"3" !== "4" returns true

If you compare an integer with a string, the string is converted to a number.

If you compare two numerical strings, they are compared as integers.

PHP Operators - Logical

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

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.

If statement syntax

```
if (condition)
    code if condition is true;
else
    code if condition is false;
```

The If Statement

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

Compound Statements

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

if .. else statement

```
<?php
$leapyear = date("L");

if ($leapyear == 1)
  echo "Hooray.  It's a leap year.";
else
  echo "Bad luck.  Not a leap year.";
?>
```

Note the use of the date function. Depending on the argument supplied to it, it can produce many different results.

While function

The while function allows you to create a loop.

You can use it in two ways.
With the test at the top:

```
<?php
$num = 1;
while ($num <= 5)
   {
      echo $num;
      echo "<br>";
      $num = $num + 1;
   }
?>
```

or as a do..while loop, with the test at the bottom:

```
<?php
$num = 1;
do {
     echo $num;
     echo "<br>";
     $num = $num + 1;
   } while ($num <= 5);
?>
```

what is the difference?