CM0133 Internet Computing

Introduction to PHP

Introduction to PHP

- So far we have seen HTML and CSS
- These are enough to create web pages
- However:
  - How can we develop more complex web based applications?
  - How do we process vast amounts of web based data?
  - If you are a business on the internet, how do you deal with thousands of financial transactions?
  - How do you store the results of financial transactions?
  - Where and how do you process these transactions?
- We need a programming language that performs well on the server !

Introduction to PHP

- One server sided programming language is PHP
- PHP is an acronym for *PHP Hypertext Processor* (note this is a recursive acronym)
- PHP is a **free** open-source technology supported by a large community of users. Open source:
  - Provides developers with access to software's source code
  - Means free redistribution rights.
  - Better bugless code
- PHP is platform independent: implementations exist for UNIX,LINUX, Windows, OSX
- PHP supports a large number of database systems, e.g. MySQL and Oracle
- PHP scripts can use many network protocols, e.g. IMAP, NNTP, SMTP, POP3 and HTTP

Introduction to PHP?

- PHP is a scripting language, where scripts run on a web-server as opposed to on the client (e.g JavaScript runs in the browser)

- PHP is web-specific – which can make it more popular than languages such as Perl (although perhaps not as powerful)

- PHP code is typically embedded into a web page, i.e. we mix the PHP code directly with the HTML code (and any JavaScript code too)

- The resulting document is saved with the extension **.php** and uploaded to a server (e.g. put them in **project_html** directory)

Template Systems v CGI

- PHP programming is a non-CGI approach to web-programming
- CGI is an acronym for Common Gateway Interface
- CGI is a protocol for allowing interaction between a client browser and a web server
- If your server supports CGI then you can write programs to run on the server (and interact with the client) in many different programming languages, e.g. **Perl**, C++, Java, Visual Basic

Templating Systems v CGI

- Large websites (e.g. BBC) require programmers, graphical designers, artists and content creators.
- With CGI programming, the script creates the HTML, e.g. the HTML is embedded in the Perl script
- Who is therefore leading the work?
  - The HTML author? The Programmer? The site designer?
  - Who does the design? Is it the programmer because they write the scripts?
  - Who decides what scripts are required? Does the page designer tells the programmer this?

Templating Systems v CGI

PHP is an example of a **templating system**

With templating systems the scripts and HTML are contained in the same file but separable to the extent where they can be developed independently

Therefore:

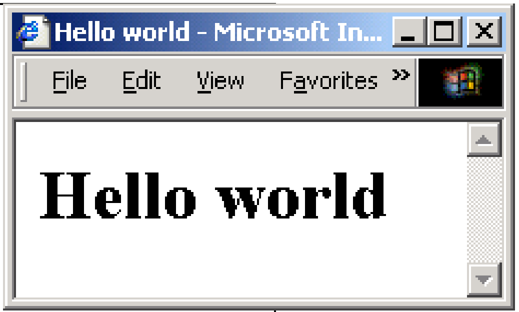
- The HTML author writes the page independently from the PHP author
- The HTML author just writes calls to scripts that the PHP programmer can develop later

What can we do with PHP?

- PHP is a fully functional programming language
- Can be used to develop complex systems
- In this course we will look at:
  - The basics of the language
    - Variables, loops, condition statements, Math, Strings..
  - Handling form data
  - Executing regular expressions
  - File handling
  - Sending Email
  - Cookies and Sessions
  - Interacting with databases

A simple PHP script

```
<html>
 <head>
  <title>Hello world</title>
 </head>
 <body>
  <h1><?php print("Hello world"); ?></h1>
 </body>
<html>
```



Hello world

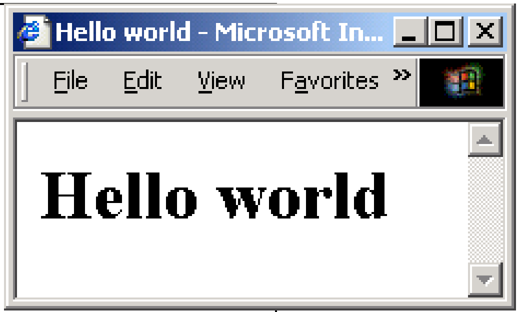
- You can write this using any text editor
- Save it with the extension **.php**
- Place the file on a server which can run php
- In our department you can place your files anywhere in your public web space or anywhere in your **public_html** directory

How it works

- PHP is installed on web server
- Our web server is Apache (just an FYI)
- Server parses files based on extensions (.php)
- Returns plain HTML, no code

A Simple PHP Script

```
<html>
 <head>
  <title>Hello world</title>
 </head>
 <body>
  <h1><?php print("Hello world"); ?></h1>
 </body>
<html>
```



Hello world

The PHP code here is contained within special HTML tags:

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

The print command is used to produce an output

HTML can also be contained **within the print command:**

```
print(“<h1> Hello World </h1>”);
```

You can also use echo instead of print

Including PHP in a web page

There are actually **4 ways** of including PHP in a web page

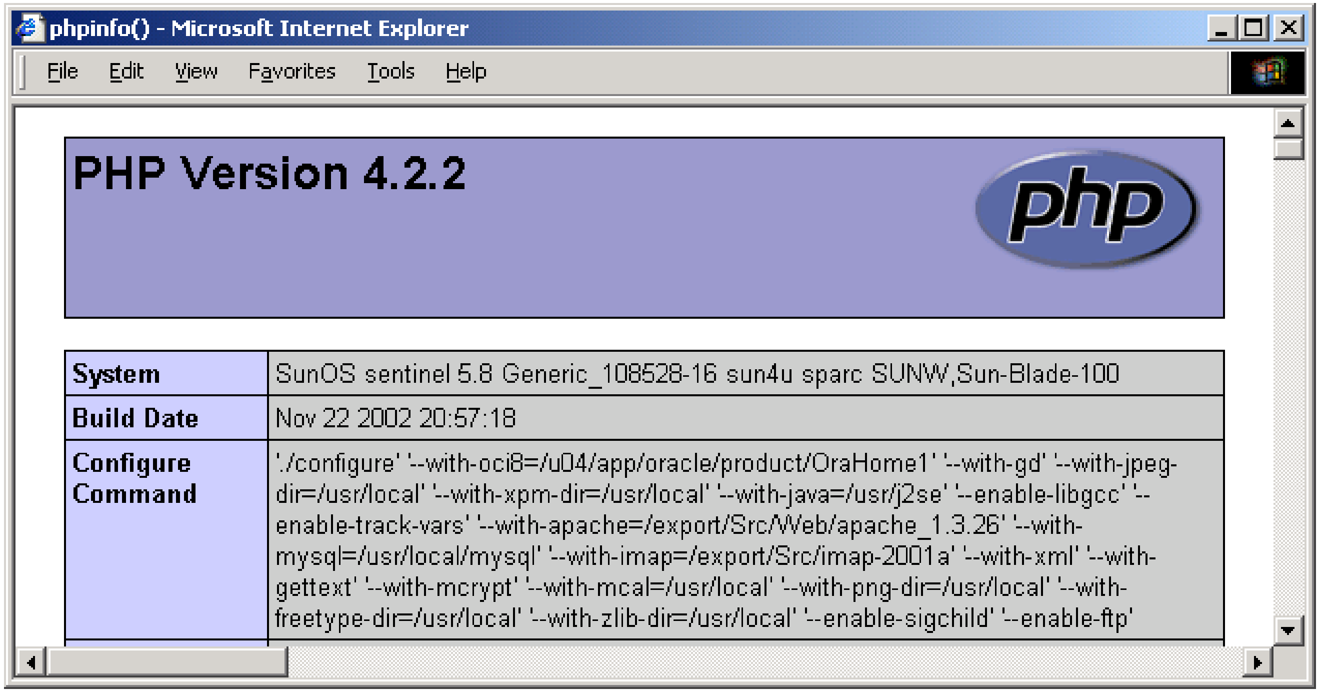
```
1) <?php print("Hello world"); ?>

2) <script language = "php">
     print("Hello world");
   </script>

3) <? print("Hello world"); ?>

4) <% print("Hello world"); %>
```

- Method (1) is clear and unambiguous (recommended)
- Method (2) is useful in environments supporting mixed scripting languages in the same HTML file (most do not)
- Methods (3) and (4) depend on the server configuration

PHP information

- To obtain information about the PHP installation (on the web server), create a file called **`info.php`** containing the single line

```
<?php phpinfo() ?>
```



PHP Version 4.2.2

System	SunOS sentinel 5.8 Generic_108528-16 sun4u sparc SUNW,Sun-Blade-100
Build Date	Nov 22 2002 20:57:18
Configure Command	'./configure' '--with-oci8=/u04/app/oracle/product/OraHome1' '--with-gd' '--with-jpeg-dir=/usr/local' '--with-xpm-dir=/usr/local' '--with-java=/usr/j2se' '--enable-libgcc' '--enable-track-vars' '--with-apache=/export/Src/Web/apache_1.3.26' '--with-mysql=/usr/local/mysql' '--with-imap=/export/Src/imap-2001a' '--with-xml' '--with-gettext' '--with-mcrypt' '--with-mcal=/usr/local' '--with-png-dir=/usr/local' '--with-freetype-dir=/usr/local' '--with-zlib-dir=/usr/local' '--enable-sigchild' '--enable-ftp'

PHP Basics: Variables

- Like in JavaScript, you don’t have to explicitly assign a data type to your variables
- The PHP interpreter works out what the type should be based on what data you put in a variable
- Variables:
  - Can contain mixtures of numbers and letters
  - Are case-sensitive (e.g. **`$fred`** is a different variable to **`$FRED`**)
  - Cannot start with a digit
- All variables begin with a dollar sign **`$`**

PHP Basics: Variables

- Numbers are either Integers or floating point

```
– $positiveInteger = 123;
– $negativeInteger = 65;
– $positiveFloat = 34.3;
– $negativeFloat = -8.547;
```

- Strings may be contained in single or double quotes

```
– $singlequoteeg = ‘This is a string!’;
– $doublequoteeg = “This is also a string!”
```

- NOTE: If you use double quotes, any PHP variables inside the string are replaced by their value

```
– $newstring = “Hello there. $singlequoteeg”;
```

PHP Basics: Variables

To display variable values they may be placed in double quotes as part of string or using a concatenation operator (which is a dot '.')

- Also note the use of comments with //

```
<html>
   <head></head>
   <body>
   <?php
        $start = "Hello ";
        $end = "There";
        $both = $start . $end;
        print("<p>Result of string concatenation</p>");
        print("<p>is : " . $both . "</p>");
        // Can also display result this way
        print("<p>is : $both </p>");
   ?>
   </body>
</html>
```

Common Operators (PHP)

+	Adds numbers/Concatenates strings
-	Subtracts numbers/Reverses sign
*	Multiplies numbers
/	Divides numbers
%	Modulus division (returns remainder from division)
!	Logical NOT
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	True if both operands are equal
!=	True if both operands not equal
&&	Logical AND
||	Logical OR

Note that the ones shown are identical to those in JavaScript and Perl

PHP Basics: Arrays

- Arrays are handled in exactly the same way as JavaScript
- Array indices begin at zero, arrays begin with dollar sign **$**

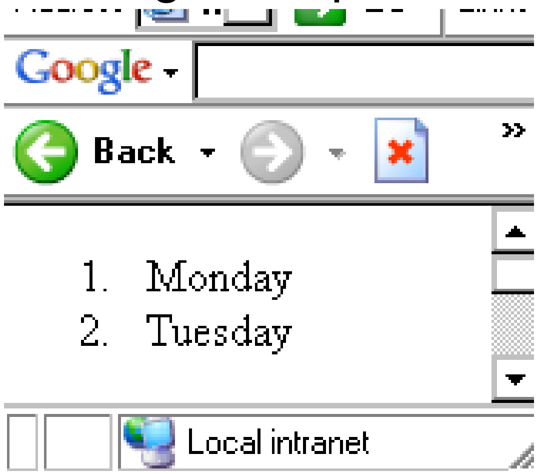
```
<html>
<head></head>
<body>
<?php

$array[0] = "Apple";
$array[1] = "Orange";
/*
*Display the array in a list
*/
print("<ul>");
print("<li> $array[0] </li>");
print("<li> $array[1] </li>");
print("</ul>");
?>
</body>
</html>
```

Note the alternate approach to including comments – this Comment spans multiple lines

Note the combination of HTML and PHP variables

PHP Basics: Associative Arrays

- In an associative array each value is indexed using a unique name (a unique string) rather than a number



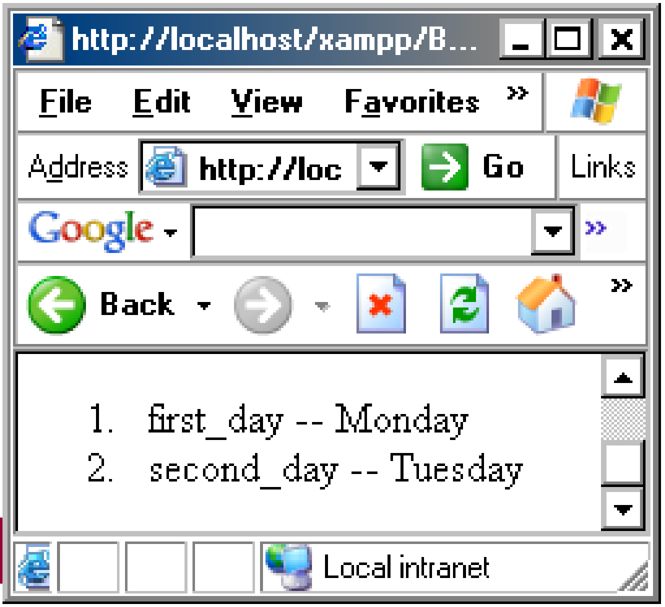
```
1. Monday
2. Tuesday
```

```
<?php

// I might normally do this:
$normalArray[0] = "Monday";
$normalArray[1] = "Tuesday";

//But im using an associative array now..
$associativeArray["first_day"] = "Monday";
$associativeArray["second_day"] = "Tuesday";

print("<ol>");
print("<li>". $associativeArray["first_day"]);
print("<li>". $associativeArray["second_day"]);
print("</ol>");
?>
```

PHP Basics: Associative Arrays

- We can use a **foreach** to loop over associative arrays

```
<?php
$associativeArray["first_day"] = "Monday";
$associativeArray["second_day"] = "Tuesday";

print("<ol>");
foreach($associativeArray as $key => $val) {
  print("<li>$key -- $val</li>");
}
print("</ol>");
?>
```



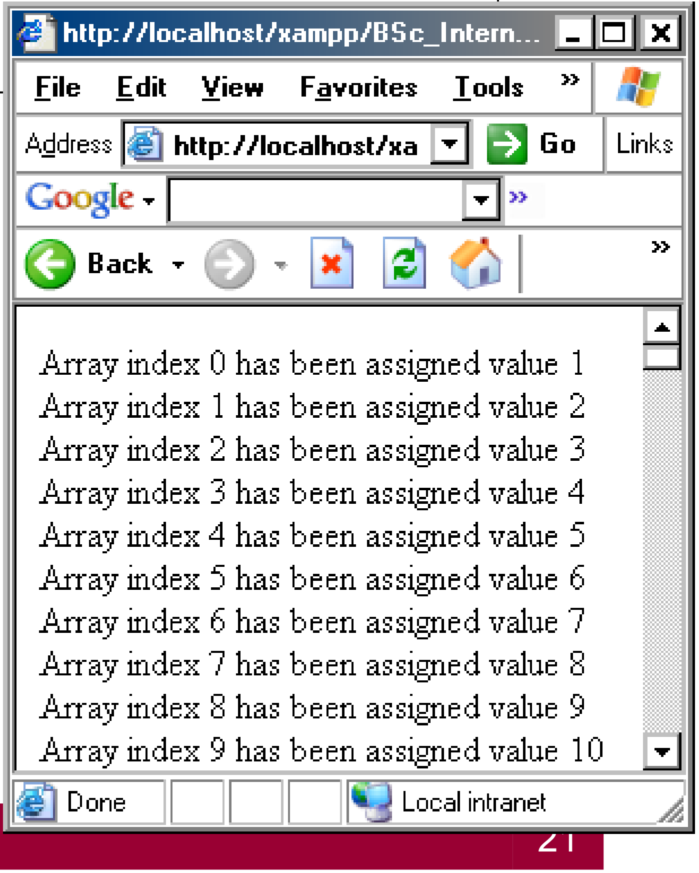
1. first_day -- Monday
2. second_day -- Tuesday

PHP Basics: for loops

- for loops use same structure as in JavaScript, Java and Perl:

```
for (initialise counter ; test condition ; increment)  {
    do something;
}
```

```
<?php
for($i=0; $i < 100; $i++){
$myArray[$i] = $i+1;
print("Array index $i has been ");
print("assigned value $myArray[$i]");
print("<br>");
}
?>
```


http://localhost/xampp/BSc_Intern...
File Edit View Favorites Tools
Address http://localhost/xa
Go
Links
Google
Back
Done
Local intranet

```
Array index 0 has been assigned value 1
Array index 1 has been assigned value 2
Array index 2 has been assigned value 3
Array index 3 has been assigned value 4
Array index 4 has been assigned value 5
Array index 5 has been assigned value 6
Array index 6 has been assigned value 7
Array index 7 has been assigned value 8
Array index 8 has been assigned value 9
Array index 9 has been assigned value 10
```

PHP Basics: **`while`** loops

- Again, same structure as Java, JavaScript, Perl…

`while` (condition is true) {do something }

```
<?php
$i=0;
while($i<100){
$myArray[$i] = $i+1;
print("Array index $i has been ");
print("assigned value $myArray[$i]");
print("<br>");
++$i;
}
?>
```

PHP Basics: Condition Statements

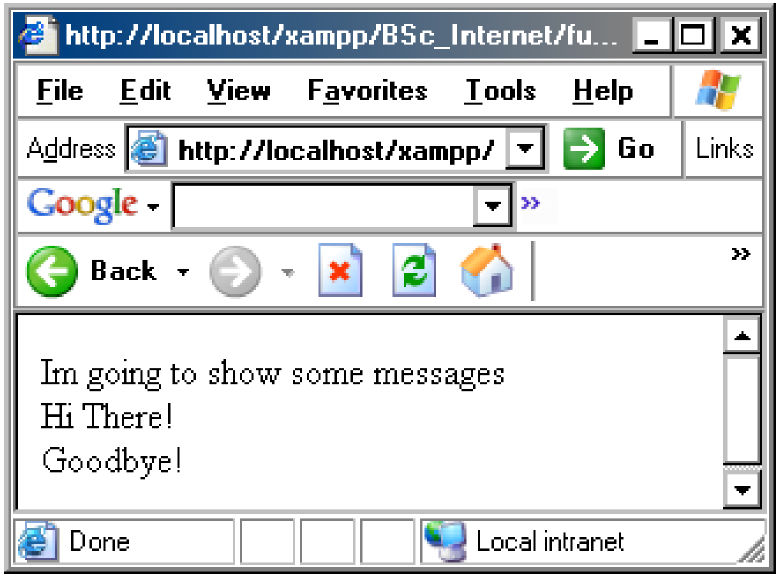
- There are some minor differences to JavaScript (e.g. spacing of **elseif** in JavaScript is **else if**)

```
<?php

if($age>16){
        print("Your over 16");
}elseif($age>18){
        print("Your over 18");
}else{
        print("Your 16 or under..");
}

?>
```

PHP Basics: Functions

- You can define functions wherever you like - structure is the same as JavaScript

```
<head>
<?php
function sayHi(){
print("Hi There! <br>");
}
?>
</head>
<body>
<?php
print("Im going to show some messages<br>");
sayHi();
sayGoodBye();
function sayGoodBye(){
print("Goodbye! <br>");
}
?>
</body>
```


http://localhost/xampp/BSc_Internet/fu...
File Edit View Favorites Tools Help
Address http://localhost/xampp/ Go Links
Google
Back
Done
Local intranet

```
Im going to show some messages
Hi There!
Goodbye!
```

PHP Basics: Scoping

```
<html>
<head>
<?php
$age = 18;$name = “Bob”;
function showStuff($name){
global $age;
print("<br>You are $age");
print("<br>You are $name");
}
?>
</head>
<body>
<?php
print("<br>You are $age");
showStuff($name);
?>
</body>
</html>
```

You can use variables defined outside functions anywhere in the program. e.g. `$age` is used in the top fragment and bottom fragment.

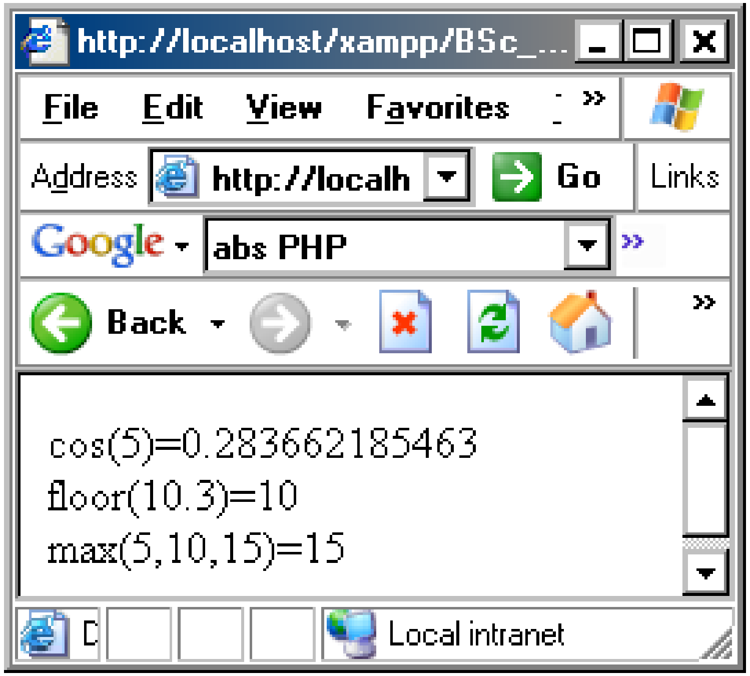
If you want to use a variable declared outside a function within a function you can pass it as an argument to that function or write `global` before it inside the function

E.g. `$name` is passed as an argument to `showStuff`.
`$age` can be used inside `showStuff` because I’ve written `global $age;`

Selected Math Functions

- cos(float), sin(float), tan(float), deg2rad(float)
- abs(number), floor(float), ceil(float), round(float)
- max(arg1, arg2[, argn]), min(arg1, arg2[, argn])

```
<?php
$a = 5;
$b = 10.3;
$c = 15;

print("cos(5)=".cos($a));
$b = floor($b);
print("<br>floor(10.3)=".$b);
$maximum = max($a,$b,$c);
print("<br>max(5,10,15)=".$maximum);
?>
```



```
cos(5)=0.283662185463
floor(10.3)=10
max(5,10,15)=15
```

Processing Form Data

- When studying HTML forms and JavaScript we took some user input and processed it on the client side
- That is, the browser ran the JavaScript code to process the form data and display some feedback
- This is fine for:
  - Running simple programs from form data (e.g. calculators…)
  - Checking that forms have correctly been filled in
- However, JavaScript is not suitable for heavy processing, database access, handling financial transactions, remembering user details, site security..
- PHP is powerful enough to be well suited to all these tasks

Processing Form Data

- Recap: We may use JavaScript to initially check all form fields are filled in before sending data to the server.

```
 <form name="myForm" method="POST" action="processForm.php"
onSubmit="return verifyForm()">
  Name: <input type="text" name="username"><br>
  Address:<input type="text" name="address"><br>
  <input type="submit" value="Send">
 </form>
```

- In this example – when submit is pressed - if the JavaScript function `verifyForm()` returns true, then the form data will be sent to `processForm.php` – i.e. the page defined in the action attribute of the form
- We can actually send the data to any PHP program we like

Processing Form Data

- In this example the data is sent to `processForm.php`
- Whenever we send form data in PHP (v4.1 and above) it gets stored in a PHP global array called: `$_POST` or `$_GET`
- The data will be stored in one of these depending on how you send the form data, i.e. whether or not you set `method = "POST"` or `method = "GET"` in the form

- PHP has other global arrays we can use.
- We will look at `$_COOKIE` and `$_SESSION` later on..

Reading $_POST or $_GET

- It is very simple to access `$_POST` or `$_GET` and retrieve the form data.
- This is what `processForm.php` might look like:

```
<?php

// Extract the form data from $_POST
extract($_POST);

//We now have two variables:
//$username and $address
//We can use these as we like..

print("Username: $username");
print("<br>Address: $address");

?>
```

These variable names Depend on the names given to inputs in the form: e.g. the first text field had `name = “username”`

The_Form.html

```
<body>
  <form method="POST" action="display.php“>
  <h1>Please fill in all fields:</h1> Title:
   <select name = "title">
      <option selected>Mr
      <option>Mrs
      <option>Miss
  </select>
  Age: <input type = "text" name = "age" size=3>
  <br> First Name: <input type = "text" name = "first">
  <font color="red">*</font>
  <br> Last Name:  <input type = "text" name = "second">
  <font color="red">*</font><br>
  <font color="red">* Indicates a required field</font><br>
  <input type="submit" value="Send">
  </form>
</body>
</html>
```

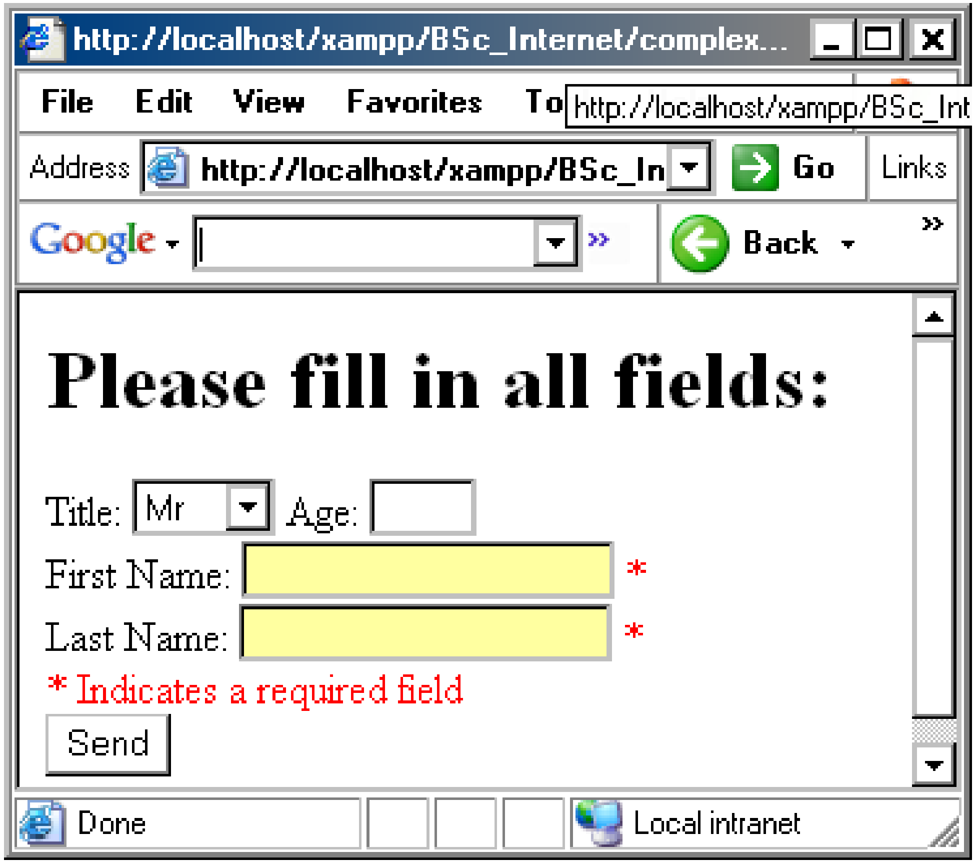
The_Form.html

- The form uses a JavaScript function to check that first/last name fields are filled in

- If they are then form data is sent to `display.php`

- The names given to form inputs are: `first, second, title, age`

- Note how `display.php` mixes PHP fragments and HTML



http://localhost/xampp/BSc_Internet/complex...
File
Edit
View
Favorites
To
http://localhost/xampp/BSc_Int
Address
http://localhost/xampp/BSc_In
Go
Links
Google
Back
Done
Local intranet

Please fill in all fields:

Title: Mr Age:

First Name: *

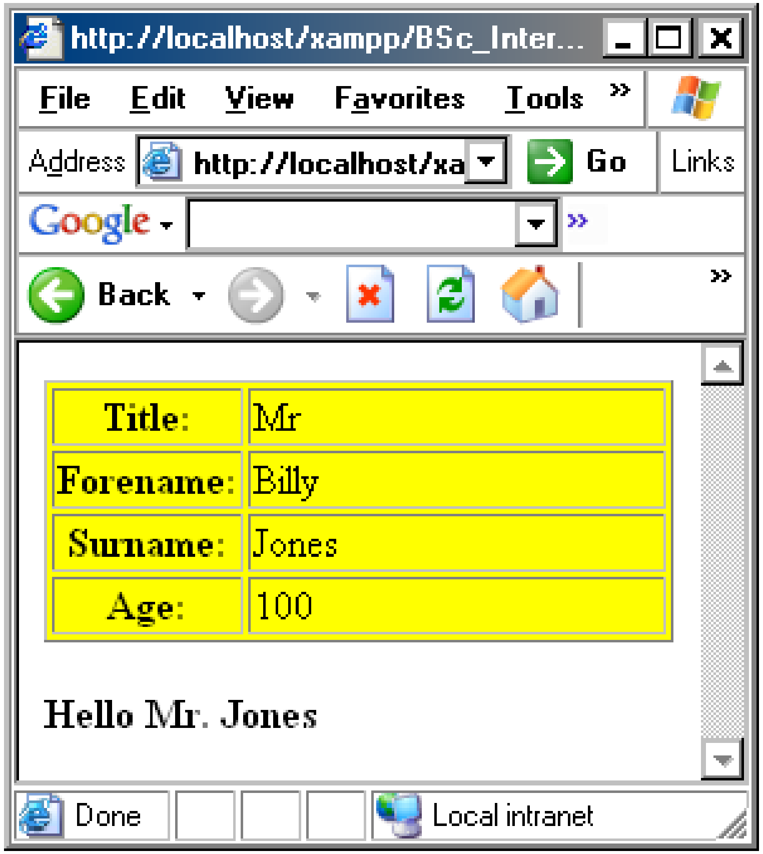
Last Name: *

* Indicates a required field

Send

display.php

```
<?php extract($_POST); ?>
<table width = 250 border=1 bgcolor="yellow">
<tr>
   <th width = "25%">Title:
   <td width = "75%"><?php print($title) ?>
</tr><tr>
   <th>Forename:
   <td><?php print($first) ?>
</tr><tr>
   <th>Surname:
   <td><?php print($second) ?>
</tr><tr>
   <th>Age:
   <td><?php print($age) ?>
</tr>
</table>
<?php
   if($first==“Billy"){
   print("<br><b>Hello $title. $second<b>");
   }
?>
```

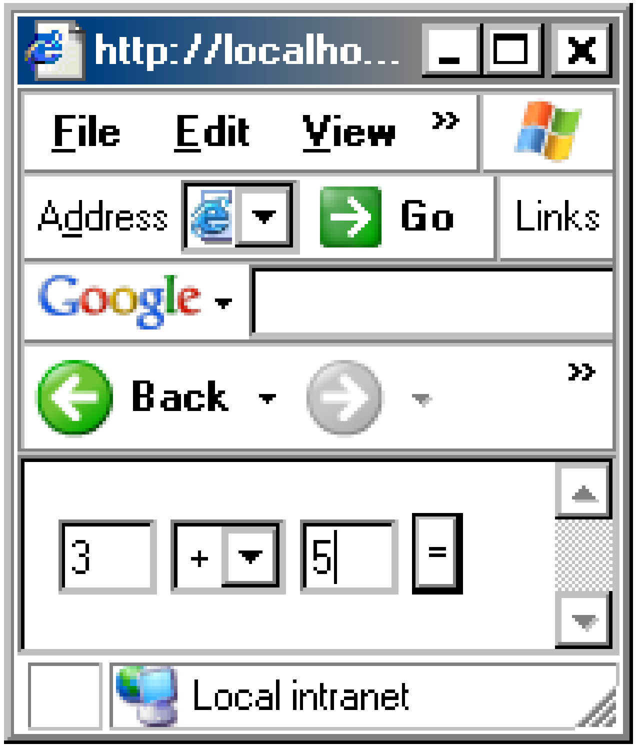

http://localhost/xampp/BSc_Inter...
File Edit View Favorites Tools
Address http://localhost/xa
Go
Links
Google
Back
Done
Local intranet

Title:	Mr
Forename:	Billy
Surname:	Jones
Age:	100

Hello Mr. Jones

Simple PHP Calculator – the form

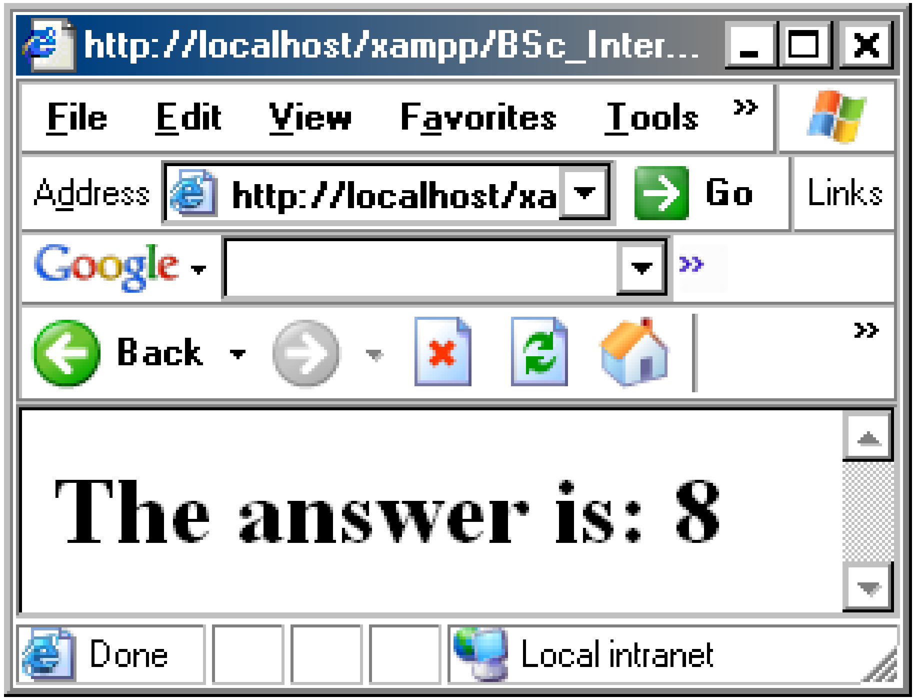
```
<form method="POST" action="calc.php">
   <input type="text" name="num1" size=1>
   <select name = "operation">
   <option>+
   <option>-
   </select>
   <input type="text" name="num2" size=1>
   <input type="submit" value = "=">
</form>
```



3 + 5 =

Simple PHP Calculator – calc.php

```
<?php
extract($_POST);
if($operation=="+"){
   $answer = $num1 + $num2;
}else{
   $answer = $num1 - $num2;
}
?>
<h1>The answer is: <?php print($answer) ?> </h1>
```



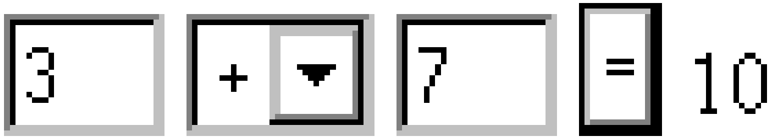
The answer is: 8

Self Referencing

- We don’t have to send Form data to a new PHP program
- You can have the action of the form self-reference the page that created the form
  - Keeps all form processing in one page
  - Good if PHP scripts are small
  - Good if not too many PHP fragments in one page
- The advanced calculator sends the form variables back to its self – its much neater than the last version
- We set `action="<?php $_SERVER['PHP_SELF'] ?>">` to self reference the page

Advanced Calculator

```
<?php extract($_POST);
if($operation=="+"){
   $answer = $n1 + $n2;
}else{
   $answer = $n1 - $n2;
}
?>
<form method="POST" action="<?php $_SERVER['PHP_SELF'] ?>">
<input type="text" name="n1" size=1 value="<?php print($n1); ?>">
<select name = "operation">
   <option>+
   <option>-
</select>
<input type="text" name="n2" size=1 value="<?php print($n2); ?>">
<input type="submit" value = "=">
<?php print($answer); ?>
</form>
```



3 + 7 = 10

Mixing HTML and PHP

- You can mix PHP and HTML to make you pages more dynamic

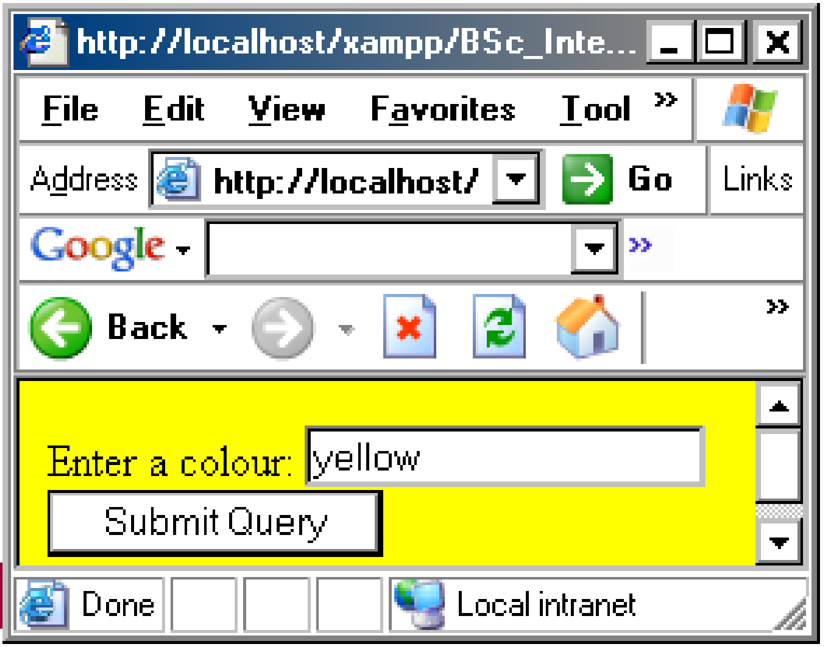
- In the following example the web pages body colour is determined by the value of the PHP string `$colour`

- You can set any HTML attribute values you like in this way: hyperlinks, image sources, table sizes etc

Mixing HTML and PHP

```
<?php
extract($_POST);
?>
<body bgcolor=<?php print($colour) ?>>
<form action = "<?php $_SERVER['PHP_SELF']; ?>" method="POST">
Enter a colour:
<input type="text" name="colour">
<br><input type = "submit">
</form>
</body>
```

Note the inclusion of the php fragment as a value for the HTML attribute


http://localhost/xampp/BSc_Inte...
File Edit View Favorites Tool
Address http://localhost/ Go Links
Google
Back
Done
Local intranet

Enter a colour: yellow

Submit Query

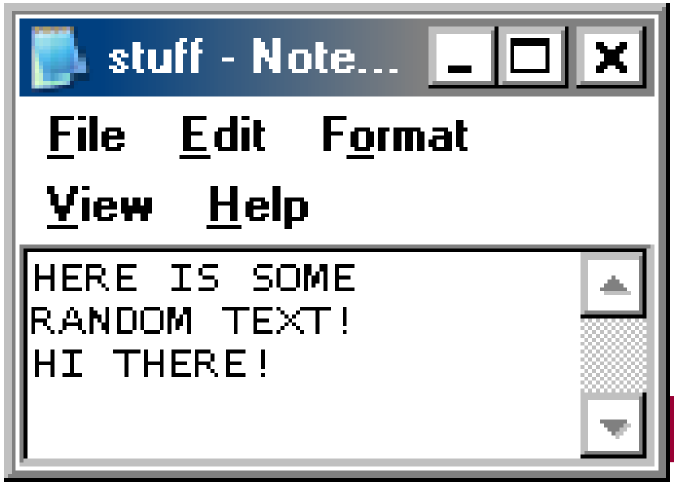
File Handling with PHP

- At some point you will want to store or access some permanent data regarding your website/site users
- You could do this by incorporating a database
- However, databases are designed to store large volumes of data
- If you have a low-volume site, then using simple files can be a better alternative
- In the long run, files are not as powerful or flexible as databases. However they are simple and quick to use.

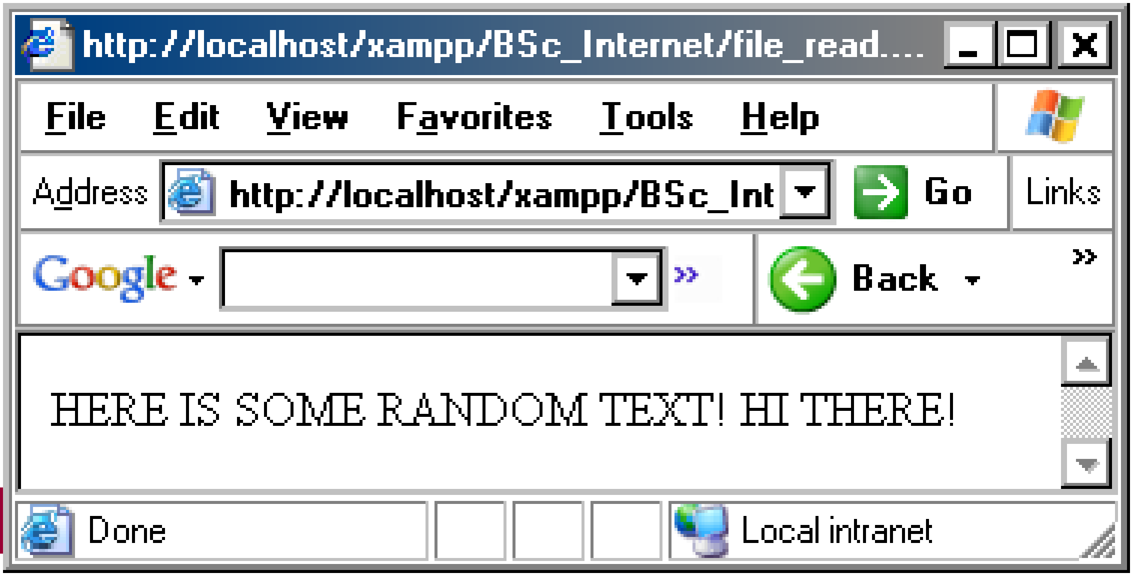
Reading files: `file_get_contents()`

- Note there are several methods to read and write files in PHP: we will only look at one
- To read files we can use `file_get_contents()`
- Reads file contents into a string, e.g:

```
<?php
$filename = "stuff.txt";
$contents = file_get_contents($filename);
print $contents;
?>
```



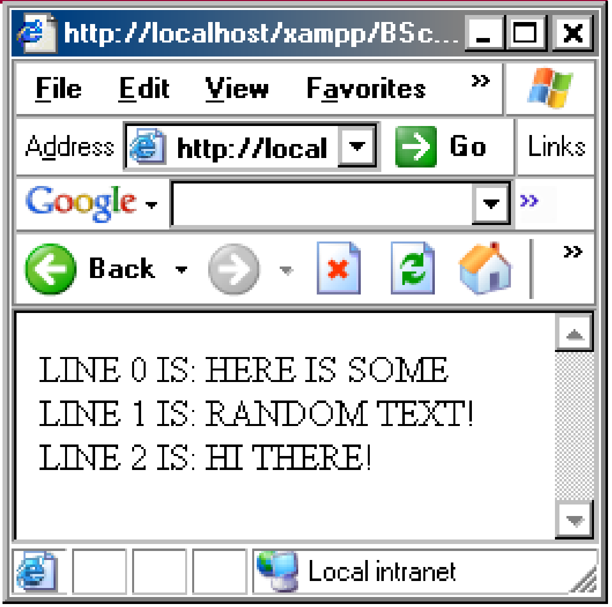
```
HERE IS SOME
RANDOM TEXT!
HI THERE!
```


http://localhost/xampp/BSc_Internet/file_read....
File Edit View Favorites Tools Help
Address http://localhost/xampp/BSc_Int
Go
Links
Google
Back
Done
Local intranet

HERE IS SOME RANDOM TEXT! HI THERE!

Reading files: `file_get_contents()`

- We can also read file contents into an array
- `\n` is a new line character (it represents a line break in a text file)
- The array is formed using the line breaks



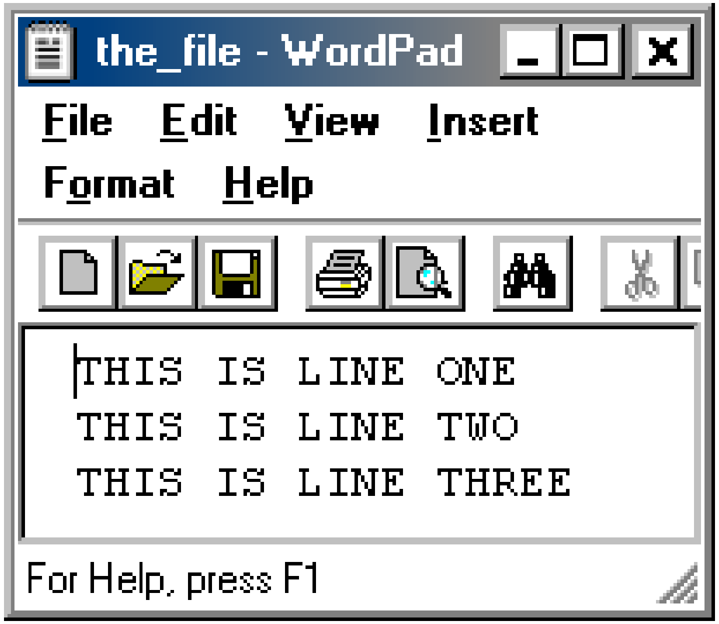
```
LINE 0 IS: HERE IS SOME
LINE 1 IS: RANDOM TEXT!
LINE 2 IS: HI THERE!
```

```
<?php
    $filename = "stuff.txt";
    $contents = file_get_contents($filename);
    $filearray = explode("\n", $contents);
    $array_length = sizeof($filearray);

    for($i=0;$i<$array_length;$i++){
       print "LINE $i IS: $filearray[$i] <br>";
    }
?>
```

Writing files: `file_put_contents()`

- The following code writes the array **`$my_array`** to the text file **`the_file.txt`**
- **`implode()`** makes each entry in the array a new line in the output file
- **`implode()`** adds line breaks at the end of each line

```
<?php
    $filename = "the_file.txt";
    $my_array[0] = "THIS IS LINE ONE";
    $my_array[1] = "THIS IS LINE TWO";
    $my_array[2] = "THIS IS LINE THREE";
    $mystring = implode("\n", $my_array);
    $numbytes = file_put_contents($filename, $mystring);
    if($numbytes){
    print("$numbytes bytes written.");
    }else{
    print("Error writing file.");
    }
?>
```



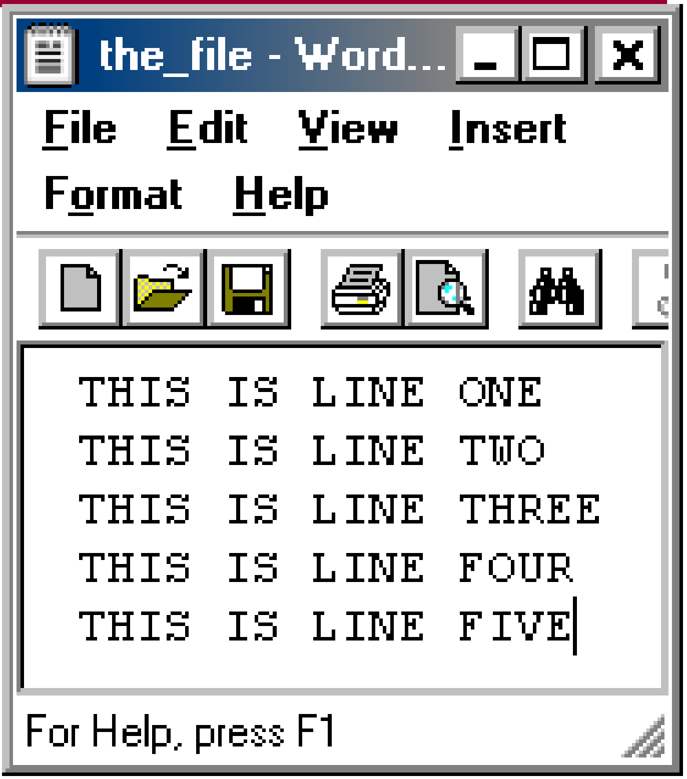
```
THIS IS LINE ONE
THIS IS LINE TWO
THIS IS LINE THREE
```

Writing files: `file_put_contents()`

- We can also *append* files, i.e. we can add to existing files
- We can simply include the argument **`FILE_APPEND`**

Ensures writing begins on a new line

```
<?php
  $filename = "the_file.txt";
  $my_array[0] = “\nTHIS IS LINE FOUR";
  $my_array[1] = "THIS IS LINE FIVE";
  $mystring = implode("\n", $my_array);
  $numbytes = file_put_contents($filename,$mystring,FILE_APPEND);
  if($numbytes){
  print("$numbytes bytes written.");
  }else{
  print("Error writing file.");
  }
?>
```



```
THIS IS LINE ONE
THIS IS LINE TWO
THIS IS LINE THREE
THIS IS LINE FOUR
THIS IS LINE FIVE
```

Reading Directory Contents

- The logical progression to working with files is working with directories – this is very straightforward
- The following program takes a directory name as a string (relative or absolute) and lists each file in the directory
- The three main functions are **`opendir(), readdir()`** and **`closedir()`**
- The directory name being read is called **`Stuff`**
- On each iteration, the name of the current file is stored in the string **`$file_name`**

Reading Directory Contents

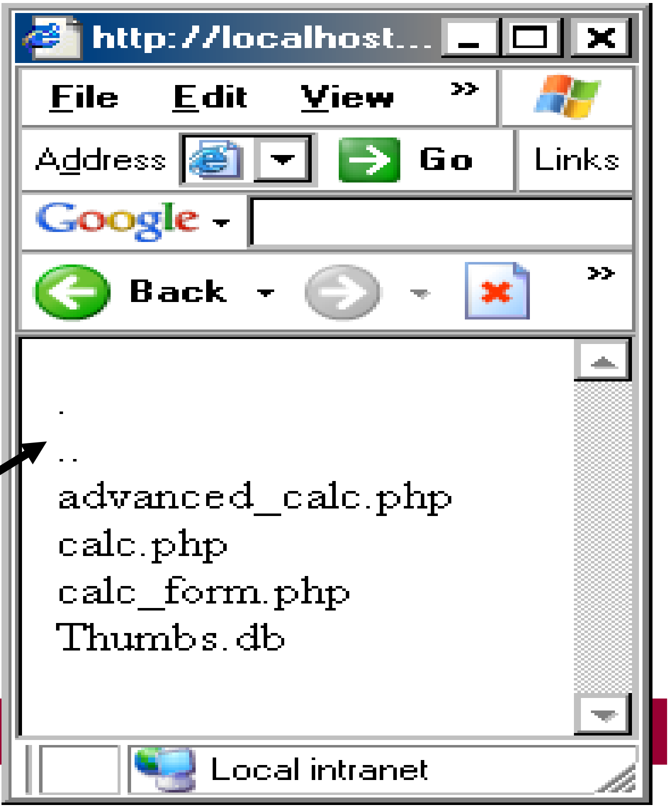
- **`opendir()`** returns a **handle** to the directory which we store in the variable **`$handle`** – we use this to reference the directory for later use
- **`readdir()`** takes the directory handle as an argument
- Each time **`readdir($handle)`** is called it returns the next file in the directory

```
while (false !== ($file = readdir($handle)))
```

- This line says: while **`readdir($handle)`** is still returning files, execute the code contained in the block
- **`!==`** means 'not equal and not the same type as'
- We use this in case **`($file = readdir($handle))`** is false, i.e. it is possible that the filename itself may evaluate to false!
- **`closedir()`** just closes the directory connection and cleans up

Reading Directory Contents

```
<?php
    $handle = opendir('Stuff');
    if($handle) {
       while(false !== ($file = readdir($handle))){
       print "$file <br>";
   }
   closedir($handle);
  }
?>
```



```
.
..
advanced_calc.php
calc.php
calc_form.php
Thumbs.db
```

Note that we may want to list only certain file types – we also may want to Remove the ‘dots’..

More to come ...

- String Manipulation
- Regular Expressions
- Mail
- Object Oriented PHP
- Databases
- State Management – Cookies & Sessions
- Parsing – XML
- AJAX & PHP

Literature

- **http://www.php.net**
- http://library.cf.ac.uk - search for PHP - programming
- http://www.adaptivepath.com/ideas/essays/archives/000385.php