PHP: FORM AND FILE HANDLING

Last time on PHP:

- In the previous PHP lecture we considered the basics:

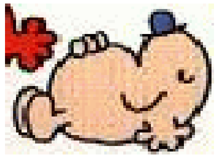
1. The History of PHP
2. The simplest ever Hello World program
3. Opening PHP Code Tags
4. PHP Variables
5. PHP Operators
6. PHP Commands
7. PHP Functions

This morning's action:

- Today we finish the basics and prepare for the real deal:

1. PHP Variable, Array, Functions, Date and Time (chapter 3)
2. Form Handling
3. HTML data Handling and Processing
4. File IO in PHP
5. A Web Counter Example
6. Protecting from hackers
7. The **all important** include statement.

1. Three Reference Slides…



empty	- Determine whether a variable is set
Isset	- same as !empty($a)
gettype	- Get the type of a variable
get_defined_vars	- Returns an array of all defined variables
is_array	- Finds whether a variable is an array
is_bool	- Finds out whether a variable is a boolean
is_float	- Finds whether a variable is a float
is_int	- Find whether a variable is an integer
is_null	- Finds whether a variable is NULL
is_numeric	- is a variable is a number or a numeric string?
is_object	- Finds whether a variable is an object
is_string	- Finds whether a variable is a string

Array Functions (the boring ones)

count	- Count elements in a variable
array_pop	- Pop the element off the end of array
array_push	- Push one or more elements onto the end of array
array_shift	- Shift an element off the beginning of array
array_search	- Searches the array for a given value and returns the corresponding key if successful
sort	- Sort an array into numerical/lexographical order

…er.. The exciting ones?

array_reverse	- Return an array with elements in reverse order
array_flip	- Flip all the values of an array keys become values
array_unique	- Removes duplicate values from an array
array_values	- Return all the values of an array
array_sum	- Calculate the sum of values in an array.
array_rand	- Pick 1 or more random entries from the array
shuffle	– randomly reorders the elements in an array

Time and Date

- **Time()** -- gives you the current UNIX timestamp
- Returns the current time measured in the number of seconds since January 1 1970 00:00:00 GMT.
- Common examples of using Timestamp:
  - Keeping track of login times
  - Time how long processes are taking
  - Keeping attendance time log

- **Date()** outputs the current date and time into whatever format you specify:

Print date("j F Y");	28 June 2009
Print date("jS F Y");	28th June 2009
Print date("F j, Y, g:i a");	June 28, 2009, 9:46 pm
Print date("H:i:s");	09:46:17
Print date("D M j G:i:s T") ;	Mon Jun 28 9:46:17 GMT

Reading Assignment: Chapter 1,2,3

2. OK so what is PHP for?

- Main purpose of PHP is analysing web data – from whole web pages to individual text fields of a form.
- So first of all lets have a quick reminder of what that means in terms of HTML.
- The values of forms can be accessed in PHP using the **$_POST**, **$_GET** and **$_REQUEST** arrays
  - Named elements
  - **Example:** $_POST[“input-element-name”]

HTML Form Handling (form.php)

the form.php (or form.html):

```
<form action="receive.php" method="post">
    Name: <input type="text" name="username"><br>
    <input type="submit">
</form>
```

receive.php - the file that will recieve the data:

```
<?
$username = $_POST["username"];
if($username)
        print "your name is $username <br>";
else
{
?>
        <a href="form.php">please fill the form!</a>
<?
}
?>
```

Combining actions (e.g. form.php)

```
<?
$username = $_POST[“username”];
if($username)
{
    echo "your name is $username !!! <br>";
}
else
{
?>

<form action="<?= $PHP_SELF ?>" method="post">
    Name: <input type="text" name="username"><br>
    <input type="submit">
</form>

<?
}
?>
```

Got that Data… now what

- As we've seen if you create an HTML form and have a php script as a target – the items of that form will **magically appear as variables** in your script.

- You must always perform client-side data validation techniques before receiving data on the server-side.

- But now we've got the data we need to process it… this always causes problems.

- With slashes everywhere, dead whitespace, capitals in the wrong places you always have to tidy it up.

- This Data processing is a good practice if you want to store it in the database

3. Trimming Strings

- **trim()** strips whitespace from the start and end of a string and returns the result.
- It also gets rid of new lines, tabs, carriage returns and the like.
- **ltrim()** and **chop()** do similar things – but only from one end of the string.

Changing Case

- Most people who use your sites *will be Muppets*. *(although your sites will probably only be viewed by members of staff so draw your own conclusions).*
- Whatever you tell them to do, **they won't enter data in the format you want and this will matter.** Login names are classic examples of this.
- Imagine someone types "aStOn VillA" into a field…

Strtoupper()	*Turns string to uppercase*	ASTON VILLA
Strtolower()	*Turns string to lowercase*	aston villa
Ucfirst()	*Capitalises first character*	Aston villa
Ucwords()	*Capitalises every word*	Aston Villa

Adding Slashes

- To do this escaping normally you just add a \ (backslash). This can be very painful to do manually.

  AddSlashes() a function to do it for you.

  StripSlashes() reverses the process.

- So if a user typed in:

  You said to me that "you don't give gaurantees"

- `$userInput = AddSlashes($userInput)`

  You said to me that /"you don/'t give gaurantees/".

Exploding Strings

- explode() returns an array of strings created by breaking the subject string at each occurrence of the separator string.
- Syntax: explode(sep, subject);

```
$email = falak.nawaz@seecs.edu.pk”;
$arr = explode(“@”, $email);
$domains = explode(“.”, $arr[1]);
```

$arr[0] = “falak.nawaz”
$arr[1] = “seecs.edu.pk”

$domain[0] = “seecs”
$domain[1] = “edu”
$domain[2] = “pk”

- **Implode()** reverses the process.

Regular Expressions

- PHP supports 2 styles of regulare expresions syntax : POSIX and PCRE (Perl Compatible)

`Ereg()`	POSIX regular expressions search
`Eregi()`	Turns string to lowercase
`Preg()`	Capitalises first character
`Pregi()`	Capitalises every word

- **Characters and Wildcards:**

```
// prints "Found 'cat'"
if (ereg("cat", "raining cats and dogs"))
        print "Found 'cat'";
```

- To represent any character in a pattern, a period is used as a wildcard. The pattern **C..** matches any three-letter string that begins with a lowercase c. For example, cat, cow, cop, and so on

- **Character List**

```
ereg("p[aeiou]p", $var)
ereg("[0-3][0-9]", "27");
```

Regular Expressions

- **Anchors**

  $match = ereg("^[0-9]", $var); // returns true if $var = "1234567"
- **Optional and Repeating Characters**
- The **?** operator allows zero or one occurrence of a character, so the expression:

  ereg("pe?p", $var)   // matches either "pep" or "pp", but not the string "peep".
- The * operator allows zero or many occurrences of the "o" in the expression:

  ereg("po*p", $var)   // matches "pp", "pop", "poop", "pooop", and so on.
- The **+** operator allows one to many occurrences of "b" in the expression:

  ereg("ab+a", $var) // so while strings such as "aba", "abba", and "abbba" match, "aa" doesn't.

- Email validation Regular Expression:

```
$email = "falak.nawaz@seecs.edu.pk";
if(!eregi("^[a-zA-Z0-9._]+@[a-zA-Z0-9-]+.[a-zA-Z0-9.]+$",$email))
{
    print "that is not a valid email address";
}
```

More on Regular Expression: Chapter 3

4. File Processing

- There are 3 steps to using data in a file

1. Open the file. If the file doesn't already exist create it or catch the error gracefully.
2. Write/Read data from the file.
3. Close the file.

- To open a file in PHP use the **fopen()** function.

- We supply it with a filename, but we also need to set the file mode that tells how we intend to use it.

fopen()

- Fopen expects 2 parameters – the location of the file and the file mode.

```
$fp = fopen("$DOCUMENT_ROOT/orders/orders.txt", "w");
```

- If no path is specified the current directory is used.

- If you are in a **windows environment** you must use double back slashes.

```
$fp = fopen("$DOCUMENT_ROOT\\orders\\orders.txt", "w");
```

Summary of File Modes

r	Read mode
r+	Reading and writing
w	OverWrite mode – if the file already exists delete it and create a new one
w+	Overwrite and reading mode– if the file already exists delete it and create a new one
a	Append mode
a+	Appending and reading

Checking the file exists

- **Lots of things** can go wrong when you try and open a file.
  - *The file might not exist*
  - *You might not have permission to view it*
  - *It may already be being written to*

- The following code handles this situation:

```
$fp = fopen("orders.txt", "a");
if (!fp)
{
  print "There were problems opening the file";
  exit();
}
```

Writing and Closing

- Writing to a file in PHP is easy. You can either use the function:
  - **fwrite()** …file write
  - **fputs()** …file put sting (an alias to fwrite)

```
$fp = fopen("orders.txt", "a");
fwrite($fp, "adding something to the file");
```

- All that is left is to tidy everything up by closing the file

```
fclose($fp);
```

Reading from a File

- fgets() is the most common function used - It is used to read one line at a fime from a file. In this case below it will read until it encounters a newline character, an EOF or has read 99 bytes from the file.

```
$fp = fopen(“orders.txt”, “a”);
fwrite($fp, “adding something to the file”);
while (!feof($fp))
{
   $order = fgets($fp, 100);
   print $order.”<br>”;
}
```

feof is a really useful function when dealing with files – here we check we are not at the end of the file

- You can also use fread and fgetc.

Other useful file functions

- **File_exists(path)** – whether file exists or not.
- **filesize(path)** – tells you how many bytes the file has.
- **Unlink(path)** - deletes the file given to it as a parameter.
- **Flock(path, option)** – file locking function with options :
  - *1 : Reading lock*
  - *2 : Writing lock*
  - *3 : Release existing lock*

5. A PHP Web Counter

```
<?
  function update_counter()
  {
      $fp = fopen("orders.txt", "r+");
      $hits = fgets($fp, 100);
      rewind($fp);
      $hits++;
      fwrite($fp, $hits);
      return $hits;
  }

  $count = update_counter();
  print "page read ".$count." times";
?>
```

Problems using flat files

- There are a number of problems with working with files:
  - When a file **gets large** it can be very slow to work with.
  - **Searching** through a file is difficult. You tend to have to read everything in to memory.
  - **Concurrent access** can become problematic.
  - Hard to enforce **levels of access** to data

What is the Solution then ???

6. Its good to be Paranoid ☺

- The few people who use your site who aren't muppets *you can assume to be malevolent hackers*. They will try and put javascript and HTML into your form fields.

- To stop them is easy:

```
$str = strip_tags($str)
```

- This simply removes all HTML tags from the string supplied as the parameter – including <SCRIPT> tags.

Crypt

- **crypt()** will encrypt a string that you give it.
- This is especially useful for encrypting items such as passwords.
- This then cannot be reversed – but you can encrypt another string that is entered and then compare it with the stored encrypted string.
- You should always encrypt your users access information if its privacy is essential.

7. Includes

- One of your aims as a good programmer is to write as efficient flexible code as possible. Because of this you will write lots of functions for your PHP that can be used over and over.

- Instead of having tonnes of functions at the top of all your scripts they should be grouped together in a separate file and included in each script.

```
<?
   include(“displayFunctions.fns");
   include_once(“myStringFunctions.fns");
?>
```

- This means NEAT code.
- But also it is EFFICIENT – you don’t have to constantly retype functions.

What we have learnt - Summary

- PHP Variable, Array Functions, Date and Time
- Form Handling
- HTML data Handling and Processing
- File IO in PHP
- A Web Counter Example
- Protecting from hackers
- The **all important** include statement.

- What Next:
  - Playing with DBMS and connecting MySQL with PHP