PHP Hypertext preprocessor

COURSE MATERIALS

You can access the course materials via this link

http://goo.gl/ev41na

CONTENTS-DAY03

- String Manipulation
- Regular Expressions
- Uploading Files
- Session
- Cookies

FORMATING A STRING FOR DISPLAY

- To trim any excess whitespace from the string:

`chop()`, `rtrim()`, `ltrim()`, **and** `trim()`

```
$name = trim($_POST['name']);
```

- The trim() function strips whitespace from the start and end of a string and returns the resulting string. The characters it strips by default are newlines and carriage returns (\n and \r), horizontal and vertical tabs (\t and \x0B), end-of-string characters (\0), and spaces.
- `chop()` is an alias of `rtrim()`.

FORMATING A STRING FOR DISPLAY

- `nl2br()` function takes a string as a parameter and replaces all the newlines in it with the XHTML
 tag.
- You can apply some more sophisticated formatting using the functions `printf()` and `sprintf()`.

```
string sprintf (string format [, mixed
args...])
```

```
void printf (string format [, mixed args...])
```

CHANGING THE CASE OF A STRING

- `Strtoupper()`, `strtolower()`, `ucfirst()` and `ucwords()`.
- `ucfirst()` Capitalizes first character of string if it's alphabetic.
- `ucwords()` Capitalizes first character of each word in the string that begins with an alphabetic character.

FORMATTING STRINGS FOR STORAGE

- Certain characters are perfectly valid as part of a string but can cause problems, particularly when you are inserting data into a database because the database could interpret these characters as control characters. The problematic ones are quotation marks (single and double), backslashes (\), and the NULL character. You need to escape that using:

`addslashes()` and `stripslashes()`

JOINING AND SPLITTING STRINGS

- Using `explode()`, `implode()`, and `join()`

```
array explode(string separator, string input
[, int limit]);
```

- This function takes a string input and splits it into pieces on a specified separator string. The pieces are returned in an array. You can limit the number of pieces with the optional ***limit*** parameter.
- You can reverse the effects of explode() by using either `implode()` or `join()` .

JOINING AND SPLITTING STRINGS

- Using `strtok()`

```
string strtok(string input, string
separator);
```

- Unlike `explode()`, which breaks a string into all its pieces at one time, `strtok()` gets pieces (called tokens) from a string one at a time.

```
$token = strtok($feedback, " ");

echo $token."<br />";

while ($token != "") {

$token = strtok(" ");

echo $token."<br />";}
```

JOINING AND SPLITTING STRINGS

- Using `substr()`

```
string substr(string string, int start[, int length] );
```

```
$test = 'Your customer service is excellent';
```

```
substr($test, 1);
```

returns our customer service is excellent.

```
substr($test, -9);
```

returns excellent.

```
substr($test, 0, 4);
```

returns the first four characters of the string—namely, Your

COMPARING STRINGS

- Using : `strcmp(), strcasecmp().`

```
int strcmp(string str1, string str2);
```

Returns < 0 if str1 is less than str2; > 0 if str1 is greater than str2, and 0 if they are equal.

```
$string = 'abcdef';
echo $string[0];                    // a
echo $string[3];                    // d
```

GET STRING LENGTH

- You can check the length of a string by using the `strlen()` function. If you pass it a string, this function will return its length. For example, the result of code is 5:

```
echo strlen("hello");
```

SEARCHING STRING

- `strstr()`, `strchr()`, and `stristr()`
- can be used to find a string or character match within a longer string the `strchr()` function is exactly the same as `strstr()`

```
string strstr(string haystack, string needle, [, bool $before_needle = false ] );
```

- You pass the function a haystack to be searched and a needle to be found. If an exact match of the needle is found, the function returns the haystack from the needle onward; otherwise, it returns false.

FINDING THE POSITION OF A SUBSTRING

- `strpos()`, `strrpos()` and `stripos()`
- They return the numerical position of a needle within a haystack.

```
int strpos(string haystack, string needle,
int [offset] );
```

FINDING THE POSITION OF A SUBSTRING

- For example, the following code echoes the value 4 to the browser:

```
$test = "Hello world";

echo strpos($test, "o");
```

- The optional offset parameter specifies a point within the haystack to start searching For example,

```
echo strpos($test, 'o', 5); // 7
```

REPLACING SUBSTRINGS

- `str_replace()` and `substr_replace()`

```
string substr_replace(string string, string
replacement,

int start, int [length] );
```

- This function replaces part of the string string with the string replacement. Which part is replaced depends on the values of the start and optional length parameters.

STRING MANIPULATION

```
int ord ( string $string );

string md5 ( string $str);

string str_repeat ( string $input , int
$multiplier );

string str_shuffle ( string $str );
```

REGULAR EXPRESSIONS

- PHP supports two styles of regular expression syntax: POSIX and Perl.
- A regular expression is a way of describing a pattern in a piece of text.
- we will cover the simpler POSIX style here.

WILD CHARACTER

. (DOT) matches single character.

ex .at will match cat rat hat … or #at

to match a character class we use

[a-z]

ex [a-z]at

will match letters only as hat rat cat NOT #at or $at

CHARACTER CLASSES

- Get all words that contain any vowel

[aeiou]

[a-zA-Z]

[^a-z] Not contain

[a-z]* can be repeated zero or more times.

[a-z]+ can be repeated one or more times.

(very)*large

(very){1, 3}

^[A-Z] beginning with

[A-Z]$ Ending by

cat|rat|hat OR

CHARACTER CLASSES

Class	Matches
`[[:alnum:]]`	Alphanumeric characters
`[[:alpha:]]`	Alphabetic characters
`[[:lower:]]`	Lowercase letters
`[[:upper:]]`	Uppercase letters
`[[:digit:]]`	Decimal digits
`[[:xdigit:]]`	Hexadecimal digits
`[[:punct:]]`	Punctuation
`[[:blank:]]`	Tabs and spaces
`[[:space:]]`	Whitespace characters
`[[:cntrl:]]`	Control characters
`[[:print:]]`	All printable characters
`[[:graph:]]`	All printable characters except for space

CHARACTERS OUTSIDE THE SQURE BRACKETS

Character	Meaning
\	Escape character
^	Match at start of string
$	Match at end of string
.	Match any character except newline (\n)
|	Start of alternative branch (read as OR)
(	Start subpattern
)	End subpattern
*	Repeat zero or more times
+	Repeat one or more times
{	Start min/max quantifier
}	End min/max quantifier
?	Mark a subpattern as optional

CHARACTERS OUTSIDE THE SQURE BRACKETS

Character	Meaning
\	Escape character
^	NOT, only if used in initial position
–	Used to specify character ranges

FINDING SUBSTRINGS WITH REGEX

- `preg_match()` function can be used to parse string using regular expression. The parts of expression enclosed in parenthesis are called subpatterns and with them you can pick individual parts of the string.

```
$str = "<a href=\"http://example.org\">My
Link</a>";

$pattern = "/<a href=\"(.*)\">(.*)<\/a>/";

$result = preg_match($pattern, $str,
$matches);

if($result === 1) {

    // The string matches the expression

    print_r($matches);

}
```

FINDING SUBSTRINGS WITH REGEX

- `preg_match_all()` returns all matching results in the subject string (in contrast to `preg_match()`, which only returns the first one).
- To split a string using REGEX use: `preg_split()`

```
$string = "0| PHP 1| CSS 2| HTML 3| AJAX 4|
JSON";

$array = preg_split("/[0-9]+\|/", $string);
```

UPLOADING FILES

```
<form action="upload.php" method="post"
enctype="multipart/form-data"/>

<div>

<input type="hidden" name="MAX_FILE_SIZE"
value="1000000" />

<label for="userfile">Upload a file:</label>

<input type="file" name="userfile"
id="userfile"/>

<input type="submit" value="Send File"/>

</div>

</form>
```

UPLOADING FILES

- In the <form> tag, you must set the attribute `enctype="multipart/form-data"` to let the server know that a file is coming along with the regular information.
- You may have a form field that sets the maximum size file that can be uploaded. This is a hidden field and is shown here as

```
<input type="hidden" name="MAX_FILE_SIZE"
value=" 1000000">
```

- Note that the `MAX_FILE_SIZE` form field is optional, as this value can also be set server-side in bytes.

UPLOADING FILES

- The data you need to handle in your PHP script is stored in the superglobal array `$_FILES`.
- `$_FILES['userfile']['tmp_name']` is the place where the file has been temporarily stored on the web server.
- `$_FILES['userfile']['name']` is the file's name on the user's system.
- `$_FILES['userfile']['size']` is the size of the file in bytes.
- `$_FILES['userfile']['type']` is the MIME type of the file—for example, text/plain or image/gif.
- `$_FILES['userfile']['error']` will give you any error codes associated with the file upload .

UPLOADING FILES

```
if ($_FILES[‘userfile’][‘error’] > 0)
{
        echo ‘Problem: ‘;
        switch ($_FILES[‘userfile’][‘error’])
        {
        case 1: echo ‘File exceeded upload_max_filesize’;
        break;
        case 2: echo ‘File exceeded max_file_size’;
        break;
        case 3: echo ‘File only partially uploaded’;
        break;
        case 4: echo ‘No file uploaded’;
        break;
        case 6: echo 'Cannot upload file: No temp directory specified';
        break;
        case 7: echo 'Upload failed: Cannot write to disk';
        break;
        }
        exit;
}
```

UPLOADING FILES

```
// put the file where we'd like it
$upfile = '/uploads/'.$_FILES['userfile']['name'] ;
// Does the file have the right MIME type?
if ($_FILES['userfile']['type'] != 'text/plain')
{
echo 'Problem: file is not plain text';
exit;
}
if (is_uploaded_file($_FILES['userfile']['tmp_name']))
{
        if (!move_uploaded_file($_FILES['userfile']['tmp_name'], $upfile))
        {
        echo 'Problem: Could not move file to destination directory';
        exit; }
else
{
echo 'Problem: Possible file upload attack. Filename: ';
echo $_FILES['userfile']['name'];
exit;
}
echo 'File uploaded successfully<br><br>';
```

UPLOADING FILES

File Upload Configuration Settings in php.ini

Directive	Description	Default Value
`file_uploads`	Controls whether HTTP file uploads are allowed. Values are On or Off.	On
`upload_tmp_dir`	Indicates the directory where uploaded files will temporarily be stored while they are waiting to be processed. If this value is not set, the system default will be used.	NULL
`upload_max_filesize`	Controls the maximum allowed size for uploaded files. If a file is larger than this value, PHP will write a 0 byte placeholder file instead.	2M
`post_max_size`	Controls the maximum size of POST data that PHP will accept. This value must be greater than the value for the `upload_max_filesize` directive, since it is the size for all of the post data, including any files to be uploaded.	8M

SESSION CONTROL

- ***HTTP is a stateless protocol***. This means that the protocol has no built-in way of maintaining state between two transactions. When a user requests one page, followed by another, HTTP does not provide a way for you to tell that both requests came from the same user.
- The idea of session control is to be able to track a user during a single session on a website. If you can do this, you can easily support logging in a user and showing content according to her authorization level or personal preferences.

SESSION CONTROL

- Sessions in PHP are driven by a **unique session ID**, a cryptographically random number.
- This session ID is generated by PHP and stored on the server side for the lifetime of a session.
- It can be either stored on a user's computer in a **cookie** or passed along through URLs.
- **Cookies** are a different solution to the problem of preserving state across a number of transactions while still having a clean-looking URL.

SESSION CONTROL

- A ***cookie*** is a small piece of information that scripts can store on a client-side machine.
- You can manually set cookies in PHP using the setcookie() function. It has the following prototype:

```
bool setcookie (string name [, string value
[, int expire [, string path[, string domain
[, int secure]]]]])
```

SESSION CONTROL

- If you set a cookie as

```
setcookie ('mycookie', 'value');
```

- you will can access the cookie via `$_COOKIE['mycookie'];`
- You can delete a cookie by calling `setcookie()` again with the same cookie name and an expiry time in the past.

SESSION CONTROL

- You can use a dual cookie/URL method:

```
session_set_cookie_params($lifetime, $path,
$domain [, $secure]);
```

- to see the contents of the cookie set by session control:

```
Array session_get_cookie_params();
```

SESSION CONTROL

- You can use a dual cookie/URL method:

```
session_set_cookie_params($lifetime, $path,
$domain [, $secure]);
```

- to see the contents of the cookie set by session control:

```
Array session_get_cookie_params();
```

- The other method it can use is adding the **session ID** to the URL. You can set this to happen automatically if you set the `session.use_trans_sid` directive in the php.ini file.

SESSION CONTROL

- The basic steps of using sessions are

1. Starting a session
2. Registering session variables
3. Using session variables
4. Deregistering variables and destroying the session

SESSION CONTROL

- Before you can use session functionality, you need to actually begin a session.

```
session_start();
```

- It's essential to call `session_start()` at the start of all your scripts that use sessions. If this function is not called, anything stored in the session will not be available to this script.
- you can begin a session is to set PHP to start one automatically when someone comes to your site. You can do this by using the `session.auto_start` option in your php.ini file.

SESSION CONTROL

- To create a session variable, you simply set an element in this array, as follows:

```
$_SESSION['myvar'] = 5;
```

- When you are finished with a session variable, you can unset it. You can do this directly by unsetting the appropriate element of the `$_SESSION` array, as in this example:

```
unset($_SESSION['myvar']);
```

SESSION CONTROL

- You should not try to unset the whole `$_SESSION` array because doing so will effectively disable sessions. To unset all the session variables at once, use

```
$_SESSION = array();
```

- When you are finished with a session, you should first unset all the variables and then call

```
session_destroy();
```

- to clean up the session ID.