PHP

Module 4

Topics

Working with PHP: Passing information between pages, HTTP GET and POST method, Cookie, Session.

String functions: strlen, strpos, strstr, strcmp, substr, str_replace, string case,

Array constructs: array(),list() and foreach(). Header().

Arrays

- An array is a data structure that stores one or more similar type of values in a single variable.

- There are three different kind of arrays and each array value is accessed using an ID, which is called array index.
  - Indexed or Numeric array
  - Associative array
  - Multidimensional array

Creating arrays

There are 3 main ways to create an array in PHP :

- By assigning a value into one
- By using the array() construct
- By calling a function that happens to return an array as its value.

Creating arrays by direct assignment

- The simplest way to create an array is to act as though a variable is already an array and assign a value into it.
- Eg:

```
<?php
$color[0]=”red”;
$color[1]=”green”;
$color[2]=”blue”;
?>
```

Creating arrays by the array() construct

- The other way to create an array is via array() construct.
- The parameter can be given an index with the => operator using the syntax “index => values” separated by commas.
- When index is omitted, an integer index is automatically generated, starting at 0.

Eg:

```
$color = array();
```

or

```
$color = array(‘red’,’green’,’blue’);
```

Creating arrays by functions returning arrays

- Functions can return value as array.
- Function can be user defined or built in.
- range() is a built in function which takes two integers as arguments and returns an array filled with all the integers between the arguments.

Eg:

```
$nos = range(1,5);          /*  $nos is the array having
                                elements 1,2,3,4,5    */
```

Types of arrays

1. Indexed or Numeric array

- An array with a numeric index.
- Values are stored and accessed in linear fashion.
- These arrays can store numbers, strings and any object but their index will be represented by numbers.
- By default array index starts from zero.

Eg:

```
$fruits = array('apple','orange','banana');
```

2. Associative array

- An array with strings as index.
- This stores element values in association with key values rather than in a strict linear index order.

Eg:

```
$fruits = array(‘red’=> ‘apple’ ,
‘orange’=> ‘orange’ , ‘yellow’=> ‘banana’);
```

3. Multidimensional array

- An array containing one or more arrays and values are accessed using multiple indices.

  Eg:

```
$meals = array(
                ‘Breakfast’ => array(‘idly’,dosa’),
                ‘Lunch’ => array(‘rice’,’biriyani’) );
```

Retrieving values

- The most direct way to retrieve a value is to use its index.

  Eg :

```
$colors = array(‘red’,’green’,’blue’);

echo $colors[1];
```

Array Functions

- **is_array()**

  Takes a single argument of any type and returns a true value if the argument is an array, and false otherwise.

- **count()**

  Takes an array as argument and returns the number of non empty elements in the array.

- **sizeof()**

  Identical to count(). It returns the no.of elements in the array.

- **in_array()**

  Takes two arguments: an element, and an array. Returns true if the element is contained as a value in the array, false otherwise.

- **isset($array[$key])**

  Takes an array[key] form and returns true if the key portion is a valid key for the array.

- **array_merge()**

  Merges one or more arrays into one array.

- **array_pop()**

  Deletes the last element of an array.

- **array_push()**

  Inserts one or more elements to the end of an array.

- **array_reverse()**

  Returns an array in the reverse order.

- **array_search()**

  Searches an array for a given value and returns the key.

- **array_sum()**

  Calculate the sum of values in an array.

- **sort()**

  Sort an array.

- **rsort()**

  Sort an array in reverse order.

- **ksort()**

  Sort an array by key.

- **list()**

  Assign variables as if they were an array.

- **each()**

  Returns the current key and value pair from an array and advances the array cursor.

Iterating through the array using Foreach loop

```
foreach($array as $value)
  {
     // code block
  }
```

- On each loop, the value of the current element is assigned to $value and the internal array pointer is advanced by one.

Eg :

```
$a = array(1,2,3,4);
foreach($a as $v)
    {
        echo $v;
    }
```

```
foreach($array as $key => $value)
  {
     // code block
  }
```

- On each loop, the value of the current element is assigned to $value and current element's key is assigned to $key and the internal array pointer is advanced by one.

Eg :

```
$a = array(“one” => 1, “two” => 2, “three” => 3);
foreach($a as $k => $v)
    {
        echo “ $k => $v “ ;
    }
```

Traversing arrays using list() and each()

- Array traversal can also done by using combination of the list() and each() function.

Eg :

```
$fruits = array("red" => "apple", "yellow" => "banana");

while(list($key,$val) = each($fruits))

    print $val;
```

Strings

- Sequence of characters.
- Strings are enclosed in quotation marks, either in single quote or double quote.

Eg:

```
$str1 = “Good Morning” ;

$str2 = ‘Good Evening’ ;
```

String Functions

strlen()

- Returns the length of the string on success, and 0 if the string is empty.
- Syntax :

```
strlen($string);
```

- Eg :

```
$str = ‘abcdef’ ;
echo strlen($str);    // returns 6
```

strpos()

- It searches for a specific text within a string. If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.
- Syntax :

```
strpos( $str1 , $str2 [ , $offset = 0 ] ) ;
```

Where,

$str1 : The string to search in.

$str2 : Item to be searched.

$offset : It specify which character in str1 to start searching.

- Eg :

```
<?php
$str1 = "Hello world!" ;
$str2 = "world" ;
echo strpos($str1 , str2);              // outputs 6
?>
```

strstr()

- It searches for the first occurrence of a string inside another string.
- Syntax :

```
strstr(str1 , str2 [ ,$pos = false ] )
```

Where,

str1 : Specifies the string to search.

str2 : Specifies the string to search for.

$pos : Optional. A boolean value whose default is "false".

If set to "true", it returns the part of the string before the first occurrence of the *search* parameter.

Eg :

```
<?php
echo strstr("This is an example program","example" );
?>
```

This will output “example program”

strcmp()

- It compares two strings.
- The comparison is Case-sensitive.
- Syntax :

    strcmp(*str1,str2*)

    It returns,

        <0 if str1 is less than str2.

        >0 if str1 is greater than str2 and,

        0 if they are equal.

Eg :

```php
<?php
echo strcmp("Hello world!","Hello world!");      //output 0
echo strcmp("Anu","Sruthi");                     //output -ve
echo strcmp("Vimal","Swetha");                   //output +ve
?>
```

substr()

- It returns a part of a string specified by the start and length parameter.
- Syntax :

  substr(string,start[,length])

- String : Required. Specifies the string to return a part of
- Start : Required. Specifies where to start in the string
- Length : Optional. Specifies the length of the returned string.

  Default is to the end of the string.

Eg :

```php
<?php
echo substr("Hello world",10);          //  result = d
echo substr("Hello world",1);           //  result = ello world
echo substr("Hello world",-8);          //  result = lo world
echo substr("Hello world",-4);          //  result = orld
?>
```

Str_replace

- It replaces some characters with some other characters in a string.
- Syntax :

```
str_replace(find,replace,string,count);
```

- Find : Required. Specifies the value to find
- Replace : Required. Specifies the value to replace the value in find
- String : Required. Specifies the string to be searched
- Count : Optional. A variable that counts the number of replacements

Eg :

```php
<?php
echo str_replace("world","Peter","Hello world!");
?>
```

Output : Hello Peter

string case

1. strtolower()

- Convert all characters to lowercase.
- Eg :

```
<?php
echo strtolower("Hello WORLD");
?>


//    Output : hello world
```

2. strtoupper()

- Convert all characters to uppercase:
- Eg :

```
<?php
echo strtoupper("Hello WORLD");
?>

//    Output : HELLO WORLD
```

Passing information between pages

There are two ways the browser client can send information to the web server.

- The GET Method
- The POST Method

The GET Method

- It passes arguments from one page to the next as part of the URL query string.

- Before the browser sends the information, it encodes it using a scheme called URL encoding. In this scheme, name/value pairs are joined with equal signs and different pairs are separated by the ampersand(&).

- Spaces are removed and replaced with the + character and any other non alphanumeric characters are replaced with a hexadecimal values. After the information is encoded it is sent to the server.

- The page and the encoded information are separated by the ? character.

- Eg :

  http://www.test.com/index.htm?name1=value1&name2=value2

- Information sent from a form with the GET method is visible to everyone (all variable names and values are displayed in the URL)
- GET should NEVER be used for sending passwords or other sensitive information.

- GET also has limits on the amount of information to send. The limitation is about 2000 characters.
- $_GET is a PHP super global variable which is used to collect form data after submitting an HTML form with method="get".

```
<html> <body>
<form action="" method="get">
<input type="text" name="number" />
<input type="submit" />
</form></body> </html>
<?php
$num = $_GET[ 'number' ];
echo $num;
?>
```

The POST Method

- $_POST is a PHP super global variable which is used to collect form data after submitting an HTML form with method="post".

- Information sent from a form with the POST method is invisible to others (all names/values are embedded within the body of the HTTP request)

- The POST method does not have any restriction on data size to be sent.

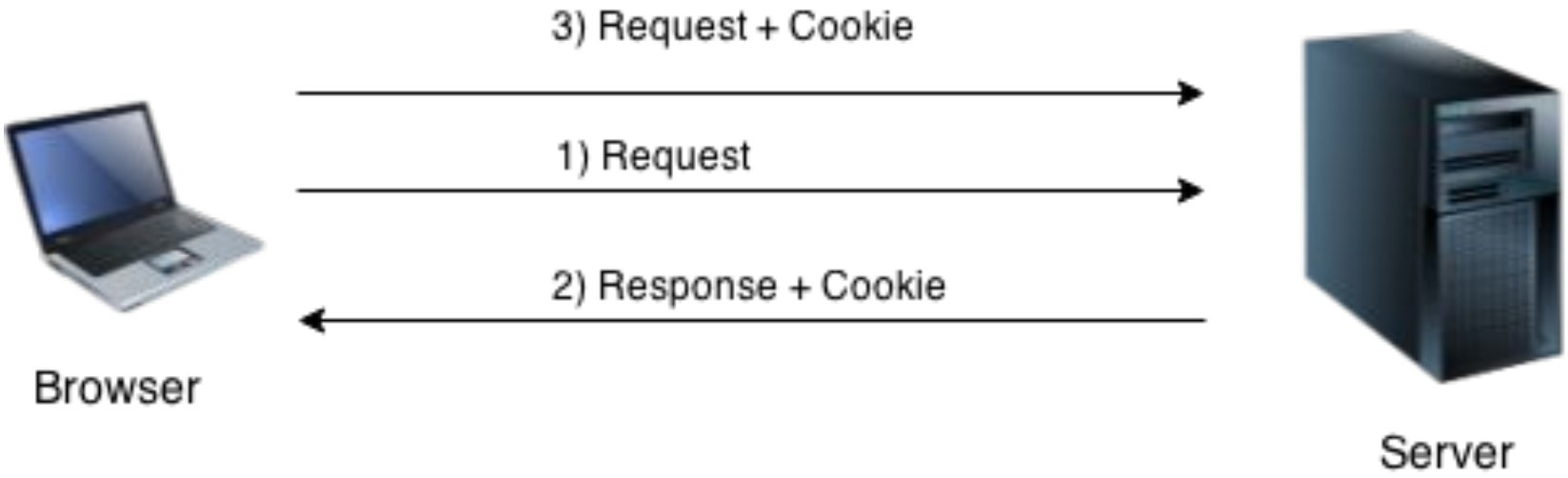
- The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.

```
<html> <body>
<form action="" method="post">
<input type="text" name="number" />
<input type="submit" />
</form></body> </html>
<?php
$num = $_POST[ 'number' ];
echo $num;
?>
```

PHP Cookie

- PHP cookie is a small piece of information which is stored at client browser. It is used to recognize the user.

- Cookie is created at server side and saved to client browser. Each time when client sends request to the server, cookie is embedded with request. Such way, cookie can be received at the server side.



- To set a cookie in PHP,the **setcookie()** function is used.

  setcookie(name, value, expire, path, domain, security);

The setcookie() function requires six arguments in general which are:

- **Name :** It is used to set the name of the cookie.
- **Value :** It is used to set the value of the cookie.
- **Expire :** It is used to set the expiry timestamp of the cookie after which the cookie can't be accessed.
- **Path :** It is used to specify the path on the server for which the cookie will be available.
- **Domain :** It is used to specify the domain for which the cookie is available.
- **Security :** It is used to indicate that the cookie should be sent only if a secure HTTPS connection exists.

- $_COOKIE super global variable is used to get cookie.

  Eg:

  ```
  $value =  $_COOKIE["CookieName"];
  ```

- For deleting a cookie, the setcookie() function is called by passing the same arguments, except the value, which should be set to an empty string.

PHP Sessions

- PHP session is used to store and pass information from one page to another temporarily (until user close the website).

- A PHP session is used to store data on a server rather than the computer of the user.

- Session identifiers or SID is a unique number which is used to identify every user in a session based environment.

Different steps involved in PHP sessions:

- **Starting a PHP Session**: The first step is to start up a session. After a session is started, session variables can be created to store information. The PHP **session_start()** function is used to begin a new session.It also creates a new session ID for the user.

- **Storing and Accessing Session Data**: Store all Session data in key-value pairs using the $_SESSION superglobal array. The stored data can be accessed during lifetime of a session.

Eg:

```
$_SESSION["username"] = "Ajay";          //Store information
echo $_SESSION["username"];              //Get information
```

- **PHP Destroying Session**: PHP session_destroy() function is used to destroy all session variables completely.

The header() function

- The header() function sends a raw HTTP header to a client.

- header() function must be called before any actual output is sent.

- Syntax :

  header(string, replace, http_response_code)

  - **String** : Specifies the header string to send
  - **Replace** : Optional. Indicates whether the header should replace a previous similar header or add a new header of the same type.
  - **Http_response_code** : Optional. Forces the HTTP response code to the specified value

The End

Thank You

Teacher : Jishna K

College of Applied Science, Thamarassery.