Marwadi University

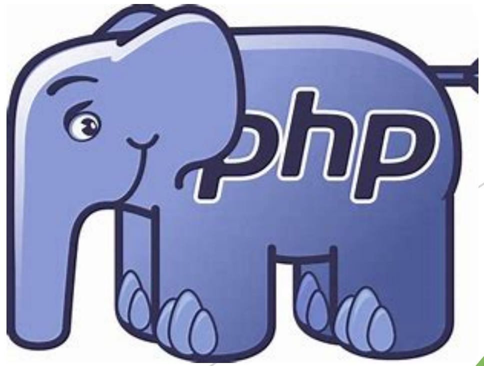
Web Application Development (PHP)

Unit – 3

Working with **Forms**, **Cookies** and User **Sessions**

Prof. Hardik Chavda

FOCA,MU



- **Working with Forms,**
  - Creating a Simple Input Form,
  - Accessing Form Input with User-Defined Arrays,
  - Combining HTML and PHP Code on a Single Page,
  - Using Hidden Fields to Save State,
  - Redirecting the User,
  - Working with File Uploads.
- **Cookies & Sessions**
  - Introducing Cookies,
  - Setting a Cookie with PHP,
  - Deleting a Cookie with PHP
  - Session Function(s) Overview,
  - Starting a Session,
  - Working with Session Variables,
  - Passing Session IDs in the Query String,
  - Destroying Sessions and Un-setting Variables,
  - Using Sessions in an Environment with Registered Users.

Creating a Simple Input Form

-
  ```
  <html>
  <body>
  <form action=“test.php" method="post">

  Name: <input type="text" name="name"><br>
  E-mail: <input type="text" name="email"><br>

  <input type="submit">

  </form>
  </body>
  </html>
  ```
- When the user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with the HTTP POST method.

```
<html>
<body>

Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>

</body>
</html>
```

```
Welcome John
Your email address is john.doe@example.com
```

Get vs Post (Super Globals)

- Both **GET** and **POST** create an array
  - (e.g. array(key1 => value1, key2 => value2, key3 => value3, ...)).
- This array holds **key/value pairs**, where keys are the names of the form controls and values are the input data from the user.
- Both **GET** and **POST** are treated as **$_GET** and **$_POST**. These are **superglobals**, which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.
- **$_GET** is an array of variables passed to the current script via the URL parameters.
- **$_POST** is an array of variables passed to the current script via the HTTP POST method.

- **When to use GET?**
  - Information sent from a form with the GET method is **visible to everyone** (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send. The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.
  - GET may be used for sending non-sensitive data.
  - **Note:** GET should NEVER be used for sending passwords or other sensitive information!
- **When to use 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) and has **no limits** on the amount of information to send.
  - Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server.
  - However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

Accessing Form Input with User-Defined Arrays

- Can be explained with example only.

Using Hidden Fields to Save State

- Can be explained with example only.

Redirecting the User,

- Redirection from one page to another in PHP is commonly achieved using the following two ways:
- Using Header Function in PHP:
- The header() function is an inbuilt function in PHP which is used to send the raw HTTP (Hyper Text Transfer Protocol) header to the client.

- **Syntax:**
  - header($header, $replace, $http_response_code)
- **Parameters: This function accepts three parameters as mentioned above and described below:**
  - **$header**: This parameter is used to hold the header string.
  - **$replace**: This parameter is used to hold the replace parameter which indicates the header should replace a previous similar header, or add a second header of the same type. It is optional parameter.
  - **$http_response_code**: This parameter hold the HTTP response code.
- **Example**

```
<?php
// Redirect browser
header("Location: https://www.hardikchavda.in");
die() or exit;
?>
```

Working with File Uploads

- PHP allows you to upload single and multiple files through few lines of code only.
- PHP file upload features allows you to upload binary and text files both. Moreover, you can have the full control over the file to be uploaded through PHP authentication and file operation functions.
- **PHP $_FILES**
- The PHP global $_FILES contains all the information of file. By the help of $_FILES global, we can get file name, file type, file size, temp file name and errors associated with file.
- Here, we are assuming that file name is ***filename***.
- $_FILES['filename']['name']
  - returns file name.
- $_FILES['filename']['type']
  - returns MIME type of the file.
- $_FILES['filename']['size']
  - returns size of the file (in bytes).
- $_FILES['filename']['tmp_name']
  - returns temporary file name of the file which was stored on the server.
- $_FILES['filename']['error']
  - returns error code associated with this file.

move_uploaded_file() function

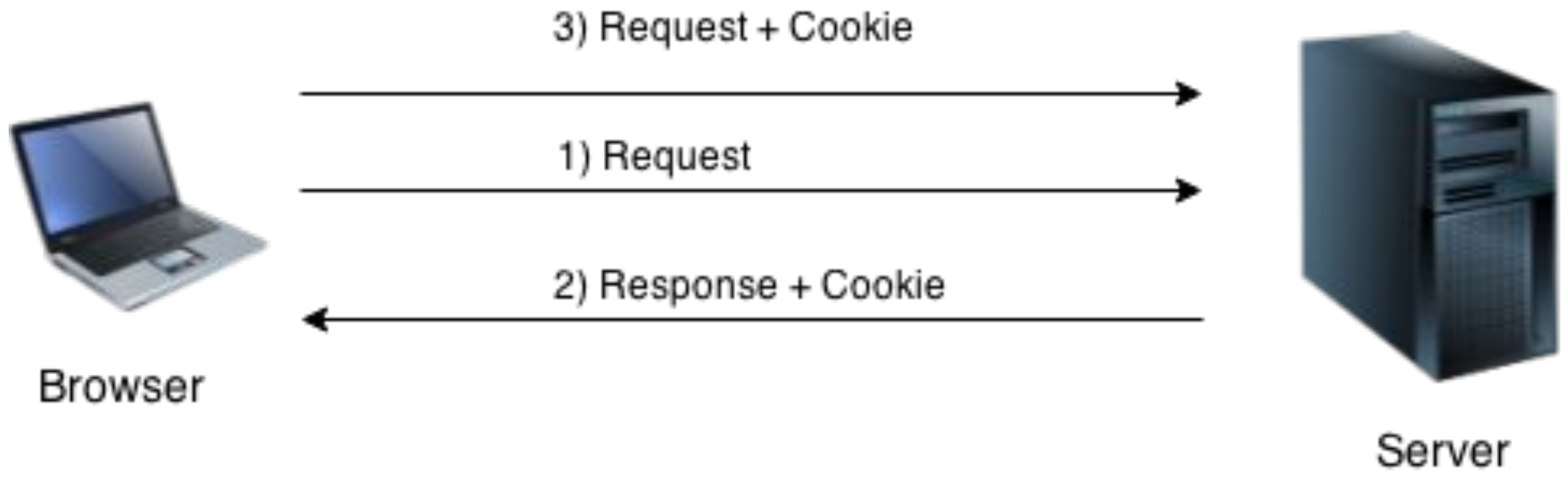
- The move_uploaded_file() function moves the uploaded file to a new location. The move_uploaded_file() function checks internally if the file is uploaded thorough the POST request. It moves the file if it is uploaded through the POST request.
- **Syntax**
- **bool** move_uploaded_file (**string** $filename , **string** $destination)

```
<form action="uploader.php" method="post" enctype="multipart/form-data">
    Select File:
    <input type="file" name="fileToUpload"/>
    <input type="submit" value="Upload Image" name="submit"/>
</form>
-----------------------------------------------------------------------------
<?php
$target_path = “uploads/";  //Location for uploadind files
$target_path = $target_path.basename( $_FILES['fileToUpload']['name']);

if(move_uploaded_file($_FILES['fileToUpload']['tmp_name'], $target_path)) {
    echo "File uploaded successfully!";
} else{
    echo "Sorry, file not uploaded, please try again!";
}
?>
```

Cookies

- 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.


3) Request + Cookie
1) Request
2) Response + Cookie
Browser
Server

PHP setcookie() function

- PHP setcookie() function is used to set cookie with HTTP response. Once cookie is set, you can access it by $_COOKIE superglobal variable.
- **Example**
- setcookie("CookieName", "CookieValue", time()+1*60*60);

//using expiry in 1 hour(1*60*60 seconds or 3600 seconds)

PHP $_COOKIE

- PHP $_COOKIE superglobal variable is used to get cookie.
- **Example**
- $value=$_COOKIE["CookieName"];//returns cookie value

Creating Cookies

```
<?php
setcookie("user", “Hardik“,time()+300);     //Will be deleted after 5 Minutes
?>
<html>
<body>
<?php
    echo "<br/>Cookie Value: " . $_COOKIE["user"];
?>
</body>
</html>
```

OP:

```
Cookie Value: Hardik
```

Editing Cookies

```
<?php
?>
<html>
<body>
<?php
      setcookie("user", “Developer“, time()+300);  //Will be deleted after 5 Minutes
      echo "<br/>Cookie Value: " . $_COOKIE["user"];
?>
</body>
</html>
```

OP:

```
Cookie Value: Developer
```

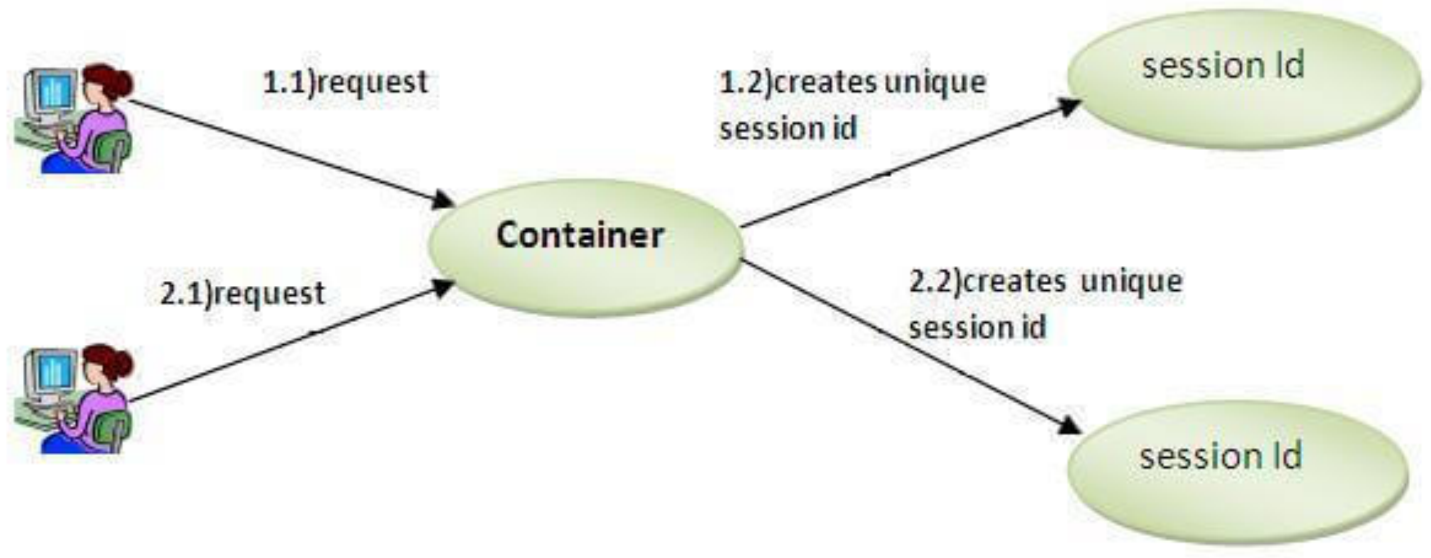
Deleting Cookies

```
<?php
?>
<html>
<body>
<?php
    setcookie("user", “Developer“, time()-300);  //Negative value will delete this cookie
    echo "<br/>Cookie Value: " . $_COOKIE["user"];
?>
</body>
</html>
```

OP:
Will Generate Error

Sessions

- PHP session is used to store and pass information from one page to another temporarily (until user close the website).
- PHP session creates unique user id for each browser to recognize the user and avoid conflict between multiple browsers.



- **session_start()**
- PHP session_start() function is used to start the session. It starts a new or resumes existing session. It returns existing session if session is created already. If session is not available, it creates and returns new session.

- **$_SESSION**
- PHP $_SESSION is an associative array that contains all session variables. It is used to set and get session variable values.

Session1.php

```
<?php
session_start();  //Compulsory for every page
?>
<html>
<body>
<?php
$_SESSION[“username”] = “HARDIK”;
echo "Session information is stored successfully.<br/>";
?>
<a href=“Session2.php">Visit next page</a>
</body>
</html>
```

Session2.php

```
<?php
session_start();   //Compulsory for every page
?>
<html>
<body>
<?php
echo "User is: ".$_SESSION[" username "];
?>
</body>
</html>
```

```
<?php
session_start();
session_destroy();
?>
```

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

```
<?php
session_start();
session_destroy();
?>
```