Unit-VI PHP Utilities

Introduction to the file input element

- The <input> element with the type="file" allows you to select one or more files from their storage and upload them to the server via the form submission.
- The following shows the file input element:

  <input type="file" id="file" name="file">
- The value of the <input> element will hold the path to the selected file. To upload multiple files, you add the multiple attribute to the <input> element like this:

  <input type="file" id="file" name="file" multiple>
- In this case, the value attribute will hold the path of the first file in the selected file list.
- To allow certain file types to be uploaded, you use the accept attribute. The value of the accept attribute is a unique file type specifier, which can be:
  - A valid case-insensitive file name extension e.g., .jpg, .pdf, .txt
  - A valid MIME type string or a string like image/* (any image file), video/* (any video file), audio/* (any audio file).
- If you use multiple file type specifiers, you need to separate them using a comma (,). For example, the following setting allows you to upload only .png and .jpeg images:

  <input type="file" accept="image/png, image/jpeg" name="file">
- The <form> element that contains the file input element must have the enctype attribute with the value multipart/form-data:

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

File Uploading

- PHP allows you to upload single and multiple files.
- 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 super global variable $_FILES used for file uploading.

$_FILES super global variable

- 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 input type file have fileupload value to the name attribute in html.

#	$_FILES Parameters	Description
1	$_FILES['fileupload']['name']	The original name of the file on the client machine.
2	$_FILES['fileupload']['type']	The mime type of the file, if the browser provided this information. An example would be "image/gif". This mime type is however not checked on the PHP side and therefore don't take its value for granted.
3	$_FILES['fileupload']['size']	The size, in bytes, of the uploaded file.
4	$_FILES['fileupload']['tmp_name']	The temporary filename of the file in which the uploaded file was stored on the server.
5	$_FILES['fileupload']['error']	The error code associated with this file upload.

move_uploaded_file() function

- When a file is uploaded successfully, it is stored in a temporary directory on the server.
- move_uploaded_file() function to move the file from the temporary directory to another one.
- The move_uploaded_file() function accepts two arguments:
  - filename: is the file name of the uploaded file which is $_FILES['file']['tmp_name'].
  - destination: is the destination of the moved file.
- The move_uploaded_file() function returns true if it moves the file successfully; otherwise, it returns false.
- Syntax: **bool move_uploaded_file (string $filename , string $destination)**

Pathinfo() function

- Pathinfo() returns information about a file path
- Syntax: ***pathinfo(string $path, int $flags = PATHINFO_ALL): array|string***
- pathinfo() returns information about path: either an associative array or a string, depending on flags.
- Parameters:
  - Path: The path to be parsed.

- Flags: If present, specifies a specific element to be returned; one of PATHINFO_DIRNAME, PATHINFO_BASENAME, PATHINFO_EXTENSION or PATHINFO_FILENAME. If flags are not specified, returns all available elements.

Important settings for file uploads in the php.ini file:

- file_uploads
  - The file_upload directive should be On to allow file upload. It defaults to On.
- upload_max_filesize
  - The upload_max_filesize specifies the maximum size of the uploaded file. By default, it's 2M (MB). If you get an error saying that the file exceeds upload_max_filesize, you need to increase this value.
- upload_tmp_dir
  - The upload_tmp_dir specifies the directory that stores the uploaded files temporarily.
- post_max_size
  - The post_max_size specifies the maximum size of the POST data. Because you'll upload files with the POST request, you need to make sure that the post_max_size is greater than upload_max_size.
- max_file_uploads
  - The max_file_uploads directive limits the number of files that you can upload at a time.

Getting information from web server

- $_SERVER super global variable is used to get the information of server and execution environment.

#	Elements/Code	Description
1	$_SERVER['PHP_SELF']	Returns the filename of the currently executing script
2	$_SERVER['GATEWAY_INTERFACE']	Returns the version of the Common Gateway Interface (CGI) the server is using
3	$_SERVER['SERVER_ADDR']	Returns the IP address of the host server
4	$_SERVER['SERVER_NAME']	Returns the name of the host server .
5	$_SERVER['SERVER_SOFTWARE']	Returns the server identification string (such as Apache/2.2.24)

6	$_SERVER['SERVER_PROTOCOL']	Returns the name and revision of the information protocol (such as HTTP/1.1)
7	$_SERVER['REQUEST_METHOD']	Returns the request method used to access the page (such as POST)
8	$_SERVER['REQUEST_TIME']	Returns the timestamp of the start of the request (such as 1377687496)
9	$_SERVER['QUERY_STRING']	Returns the query string if the page is accessed via a query string
10	$_SERVER['HTTP_ACCEPT']	Returns the Accept header from the current request
11	$_SERVER['HTTP_ACCEPT_CHARSET']	Returns the Accept_Charset header from the current request (such as utf-8,ISO-8859-1)
12	$_SERVER['HTTP_HOST']	Returns the Host header from the current request
13	$_SERVER['HTTP_REFERER']	Returns the complete URL of the current page (not reliable because not all user-agents support it)
14	$_SERVER['HTTPS']	Is the script queried through a secure HTTP protocol
15	$_SERVER['REMOTE_ADDR']	Returns the IP address from where the user is viewing the current page
16	$_SERVER['REMOTE_HOST']	Returns the Host name from where the user is viewing the current page
17	$_SERVER['REMOTE_PORT']	Returns the port being used on the user's machine to communicate with the web server
18	$_SERVER['SCRIPT_FILENAME']	Returns the absolute pathname of the currently executing script
19	$_SERVER['SERVER_ADMIN']	Returns the value given to the SERVER_ADMIN directive in the web server configuration file (if your script

		runs on a virtual host, it will be the value defined for that virtual host)
20	$_SERVER['SERVER_PORT']	Returns the port on the server machine being used by the web server for communication (such as 80)
21	$_SERVER['SERVER_SIGNATURE']	Returns the server version and virtual host name which are added to server-generated pages
22	$_SERVER['PATH_TRANSLATED']	Returns the file system based path to the current script
23	$_SERVER['SCRIPT_NAME']	Returns the path of the current script
24	$_SERVER['SCRIPT_URI']	Returns the URI of the current page

What is AJAX?

- AJAX stands for Asynchronous JavaScript and XML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS and Java Script.
- Conventional web application transmit information to and from the sever using synchronous requests. This means you fill out a form, hit submit, and get directed to a new page with new information from the server.
- With AJAX when submit is pressed, JavaScript will make a request to the server, interpret the results and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.

The XMLHttpRequest Object

- All modern browsers support the XMLHttpRequest object.
- The XMLHttpRequest object can be used to exchange data with a server behind the scenes.
- This means that it is possible to update parts of a web page, without reloading the whole page.

Create an XMLHttpRequest Object

- All modern browsers (Chrome, Firefox, Edge (and IE7+), Safari, Opera) have a built-in XMLHttpRequest object.
- Syntax for creating an XMLHttpRequest object:
    - variable = new XMLHttpRequest();
- Example

- var xhttp = new XMLHttpRequest();

AJAX - Send a Request to a Server

- To send a request to a server, we use the open() and send() methods of the XMLHttpRequest object:

#	Method	Description
1	open(method, url, async)	specifies the type of request. **method:** the type of request: GET or POST **url:** the server (file) location **async:** true (asynchronous) or false (synchronous)
2	send()	Sends the request to the server

AJAX - Server Response

- The readyState property holds the status of the XMLHttpRequest.
- The onreadystatechange property defines a function to be executed when the readyState changes.
- The status property and the statusText property holds the status of the XMLHttpRequest object.

#	Properties	Description
1	onreadystatechange	Defines a function to be called when the readyState property changes
2	readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
3	Status	200: "OK" 403: "Forbidden" 404: "Page not found" For a complete list go to the Http Messages Reference
4	statusText	Returns the status-text (e.g. "OK" or "Not Found")

- The onreadystatechange function is called every time the readyState changes.
- When readyState is 4 and status is 200, the response is ready:

Example: Test.php

```
<html>
  <head>
    <title>PHP MVC Frameworks - Search Engine</title>
    <script type="text/javascript">
      function showName(str)
      {
        if (str.length == 0){ //exit function if nothing has been typed in the textbox
          document.getElementById("txtName").innerHTML=""; //clear previous results
          return;
        }

        if (window.XMLHttpRequest) {// code for IE7+, Firefox, Chrome, Opera, Safari
            xmlhttp=new XMLHttpRequest();
        } else {// code for IE6, IE5
            xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

        }
        xmlhttp.onreadystatechange=function() {
          if (xmlhttp.readyState == 4 && xmlhttp.status == 200){
            document.getElementById("txtName").innerHTML=xmlhttp.responseText;
          }
        }

      xmlhttp.open("GET","frameworks.php?name="+str,true);
      xmlhttp.send();

    }
    </script>
  </head>
  <body>
    <h2>PHP MVC Frameworks - Search Engine</h2>
    <p><b>Type the first letter of the PHP MVC Framework</b></p>
    <form method="POST" action="">
```

```
        <p><input type="text" size="40" id="txtHint"  onkeyup="showName(this.value)"></p>
      </form>
      <p>Matches: <span id="txtName"></span></p>
    </body>
</html>
```

Framework.php

```
<?php
 $frameworks = array ("CodeIgniter","Zend Framework","Cake PHP", "Kohana", "Laravel", "Symfony",
"Laminas Project") ;
   $name = $_GET["name"];
   if (strlen($name) > 0) {
      $match = "";
      for ($i = 0; $i < count($frameworks); $i++) {
         if (strtolower($name) == strtolower(substr($frameworks[$i], 0, strlen($name)))) {
            if ($match == "") {
               $match = $frameworks[$i];
            } else {
               $match = $match . " , " . $frameworks[$i];
            }        }
      }   }
   echo ($match == "") ? 'no match found' : $match;
?>
```
