



WEB APPLICATION ENGINEERING II

Lecture #6

Umar Ibrahim Enesi



Objectives

- Gain knowledge on how file upload is handled by PHP.



HTTP File Uploads

- Just as text data is sent to the server via webforms, files can also be sent to the server via standard forms too.
- This lecture explains the requirements for and process of handling file uploads.



Client Side (HTML) Requirements

- A typical example:

```
<form action="handler.php" enctype= "multipart/form-data" method=
"POST" >
    <input type="hidden" name="MAX_FILE_SIZE" value="1024" />
    Select File: <input type="file" name="assignment" />
    <input type="submit" name="Submit" />
</form>
```



Client Side (HTML) Requirements

- A typical example:

enctype must be
multipart/form-data

```
<form action="handler.php" enctype= "multipart/form-data" method=
"POST" >
    <input type="hidden" name="MAX_FILE_SIZE" value="1024" />
    Select File: <input type="file" name="assignment" />
    <input type="submit" name="Submit" />
</form>
```



Client Side (HTML) Requirements

- A typical example:

method must
be POST

```
<form action="handler.php" enctype= "multipart/form-data" method=
"POST" >
    <input type="hidden" name="MAX_FILE_SIZE" value="1024" />
    Select File: <input type="file" name="assignment" />
    <input type="submit" name="Submit" />
</form>
```



Client Side (HTML) Requirements

- A typical example:

Optionally specify the maximum size in byte that a client can upload

```
<form action="handler.php" enctype="multipart/form-data" method="POST" >
  <input type="hidden" name="MAX_FILE_SIZE" value="1024" />
  Select File: <input type="file" name="assignment" />
  <input type="submit" name="Submit" />
</form>
```



Client Side (HTML) Requirements

- A typical example:

Use an input of type='file' with a unique name attribute

```
<form action="handler.php" enctype="multipart/form-data" method="POST" >
  <input type="hidden" name="MAX_FILE_SIZE" value="1024" />
  Select File: <input type="file" name="assignment" />
  <input type="submit" name="Submit" />
</form>
```



Handling File Upload at the Server

- When a form submission is received by the server;
 - Encoded file content is decoded and stored in a temporary location
 - Uploaded files together with all corresponding information will be made accessible to handler script.
 - Business logic in the handler script will act upon the uploaded files using the related information.



Accessing File Upload Information

- Uploaded file information is made accessible through `$_FILES` superglobal array.
 - `$_FILES['assignment']['name']` -> Original name of the file from the client machine.
 - `$_FILES['assignment']['type']` -> The mime type of the file.
 - `$_FILES['assignment']['size']` -> The size, in bytes, of the uploaded file.
 - `$_FILES['assignment']['tmp_name']` -> The temporary filename of the file in which the uploaded file was stored on the server.
 - `$_FILES['assignment']['error']` -> The error code associated with this file upload.



Validating Uploaded Files

- With the upload information and various filesystem functions made available to handler script, validation of file upload is made possible.
 - Use `is_file_uploaded()` to determine if the file was actually uploaded via HTTP POST request.
 - Examine `$_FILES['assignment']['size']` to ensure the uploaded file is within an acceptable size range.
 - Examine `$_FILES['assignment']['type']` to ensure the uploaded file is among the set of acceptable mime type.
 - Examine `$_FILES['assignment']['error']` to verify that the error code generated indicate successful upload.



Saving Uploaded Files

- Uploaded files are usually saved in a temporary directory while the script handler executes.
- The file will be deleted from the temporary directory at the end of the request if it has not been moved away or renamed.
- Use `move_uploaded_file()` to move (and rename) an uploaded file to a permanent location.



Error Codes

- Sometimes upload may fail along the line. It is the responsibility of the developer to check for the status of the upload.
- PHP exposes the status of the file upload in `$_FILES['assignment']['error']`
- Possible values are described by these constants:
 - `UPLOAD_ERR_OK (0)`: There is no error, the file uploaded with success.
 - `UPLOAD_ERR_INI_SIZE (1)`: The uploaded file exceeds the `upload_max_filesize` directive in `php.ini`.
 - `UPLOAD_ERR_FORM_SIZE (2)`: The uploaded file exceeds the `MAX_FILE_SIZE` directive that was specified in the HTML form.
 - `UPLOAD_ERR_PARTIAL (3)`: The uploaded file was only partially uploaded.
 - `UPLOAD_ERR_NO_FILE (4)`: No file was uploaded.
 - `UPLOAD_ERR_NO_TMP_DIR (6)`: Missing a temporary folder.
 - `UPLOAD_ERR_CANT_WRITE (7)`: Failed to write file to disk.
 - `UPLOAD_ERR_EXTENSION (8)`: A PHP extension stopped the file upload.



Tweaking The File Upload Settings

- There exists related configuration settings in `php.ini` that affects file upload:
 - `file_uploads`: Indicates whether to allow http file upload or not (*default = on*)
 - `upload_tmp_dir`: Specifies the path to temporary directory for uploaded file
 - `upload_max_filesize`: Specifies the maximum allowed size of uploaded file
 - `max_file_uploads`: Specifies the maximum number of files that can be uploaded in one request.
 - `post_max_size`: Sets max size of post data allowed (files + other form data).
 - `max_input_time`: Sets the maximum time a script is allowed to parse input data.
 - `memory_limit`: Sets the maximum memory a script may consume.
 - `max_execution_time`: This sets the maximum time in seconds a script is allowed to run before it is terminate.



Uploading Multiple Files (method 1)

```
<form action="handler.php" enctype= "multipart/form-data" method=
"POST" >
    Select File1: <input type="file" name="assignment1" />
    Select File2: <input type="file" name="assignment2" />
    <input type="submit" name="Submit" />
</form>
```



Uploading Multiple Files (method 2)

```
<form action="handler.php" enctype= "multipart/form-data" method=
"POST" >
    Select File1: <input type="file" name="assignment[]" />
    Select File2: <input type="file" name="assignment[]" />
    <input type="submit" name="Submit" />
</form>
```




Useful File Upload Utility Functions

- `basename()` : Returns trailing name component of path.
- `pathInfo()` : Returns information about a file path.
- `copy()` : Copies file.
- `delete()` : unlink or unset a file.
- `file_exists()` : Check if a file exists.
- `move_uploaded_file()` : Moves an uploaded file to a new location.
- `is_uploaded_file()` : Tells whether the file was uploaded via HTTP POST



Key Point

- Forms that are meant for file upload must use a method of POST.
- The enctype attribute of file upload form must be “multipart/form-data”
- PHP stores uploaded files in a temporary directory as specified in php.ini
- When a request is completed, uploaded files are deleted if they have not be moved or renamed.
- PHP makes available information about uploaded files via `$_FILES` super global array.
- Multiple files can be uploaded to the server.
- The behaviour of file upload at the server can be configured from php.ini



Reference

- PHP Documentation Manual
- Murach PHP and MySQL