Lab Homework 8: File Uploads

In Lab Homework 7 we began building a file storing system, **2300 Plop Box** in **box.php**. In lab, you'll complete the implementation of the Plop Box by programming the support for file uploads.

1. Learning Objectives

- Learn how to handle file uploads for an HTML form.
- Practice using reference documentation.

2. Deadline

Lab Homework	Deadline	Slip Days	Credit	Solution
All Parts	Sun 4/19, 11:59pm ET	Max: 2 days	20 points (completion)	Provided

3. Instructions

1. **Clone** your lab repository.

   Clone the following URL:
   `git@github.coecis.cornell.edu:info2300-2020sp/YOUR_GITHUB_USERNAME-lab08.git`
   Replace **YOUR_GITHUB_USERNAME** in the URL with **your Cornell GitHub username**.

2. **Work together.**

   Feel free to work with your peers to complete this lab. Use your section specific chat rooms. Organize a Zoom hangout to work on the assignment together. Take this as an opportunity for some *virtual* human contact!

   Note: You are encouraged to work together so long as you do your own work and you don't give away answers.

3. **Ask questions** or **say hi** during your registered section live Zoom Q & A.

   Your section leaders will hold a Zoom Q & A during your registered section time. Feel free to pop in and **say hi** or **ask a question**! Again, use this as another opportunity to keep up with your fellow Cornell community members!

4. **Submit.**

   When you're finished, follow the instructions in **submit.md** to submit your assignment.

Part I: Plop Box File Upload

In the previous lab, we set up the database table for our Plop Box. Now it's time to implement the file upload for our file upload service.

Before moving on, remember that **the database DOES NOT store the files** (although you could do this with BLOB, but this is generally not considered best practice). The database will be utilized to store information we will later use to construct the link to the file's location in the *uploads* directory. The files themselves are not stored in the database but rather on the server's file system.

1. HTML Form for File Upload

The first step in enabling users to upload files, is to implement a form for uploading files.

1. We must you a POST request.
2. We must set the `enctype` attribute on the `<form>` element.

   Read the HTML4 Specifications on Form content types to determine the appropriate content type our form needs.

   - `enctype="application/x-www-form-urlencoded"` - *default* form content type
   - `enctype="multipart/form-data"`
3. We need an input component for file uploads.

   Review the input type of `file` documentation: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/file

   Note: When setting the name for the file input, you want to avoid "file" as the naming is too generic but use something like "box_file".
4. We want to set a maximum file size to 1 MB which is 1,000,000 bytes.

   - This is a reasonable amount of space for most files. If we do not constrain the file size, this allows the potential for our website server to be overloaded. If we allow users to upload very large files, we run the risk of running out of space very quickly.
   - We therefore want to set this as a hidden input. You can learn how to do this on by looking at example 1 in this reference documentation: http://php.net/manual/en/features.file-upload.post-method.php.

   Note: The hidden input MUST come before the file input.

Objective: Review the file upload form in **box.php**. Verify that all three of these criteria have been met. If not, please correct the code.

2. Filter the Input for the File Upload Form

Objective: You will need to filter the `box_file` and `description` parameters. See the first **TODO** in **box.php**.

Take note that accessing uploaded files is different than parameters:

- For file inputs, instead of using `$_POST["input_name"]`, you will want to use `$_FILES["input_name"]`.
- `$_FILES["input_name"]` (`$_FILES["box_file"]`) is an associative array.

  See the reference documentation for the key/value pairs: http://php.net/manual/en/features.file-upload.post-method.php
- You should probably store the uploaded file associative array as a variable:
  `$upload_info = $_FILES["box_file"];`
- You should use the error codes to determine if file uploaded successfully: http://php.net/manual/en/features.file-upload.errors.php.

  Hint: `'error'` and `UPLOAD_ERR_OK`.
- If the file uploaded successfully, you need to extract the filename in order to store it in the database.

  The uploaded file name can be found in the associative array: `$upload_info["name"]`

  The `basename()` helper function can help you get the name of the file:
  http://php.net/manual/en/function.basename.php.
- You also need to extract the file extension from the basename name above. This will be used to store the file with the proper file extension in our uploads folder.

  The `pathinfo()` helper function can help you with this: http://php.net/manual/en/function.pathinfo.php.

  You should also store all file extensions as lower case for consistency. See the `strtolower()` documentation:
  http://php.net/manual/en/function.strtolower.php.

  ```
  $upload_ext = strtolower( pathinfo($basename, PATHINFO_EXTENSION) );
  ```

Tip: `var_dump()` (http://php.net/manual/en/function.var-dump.php) is always a useful function to check that you are accessing and storing your intended values.

Note: You may feel very lost and confused here. That's normal. We aren't providing much direction here because we want you to start thinking through how you would solve problems like this on your own. We also want you to develop your *reference documentation* skills. Please use this lab as an opportunity to figure out how to make this work on your own. You'll learn a lot if you do!

3. Store the Uploaded File

After you have filtered the input for the file upload, it's now time to store the related data in the database and store the uploaded file in the uploads directory.

We do not store our file uploads in the database because many databases do not handle storing large pieces of content, like file uploads, well. Instead, many web frameworks store files directly to disk. Recall that will name our uploaded file with the `id` primary key and its file extension: `uploads/TABLE_NAME/ID.FILE_EXTENSION`

Objective: Before we can place our uploaded file in the **uploads/documents** directory, we first need a primary key for this upload. That means we must first insert a record into our database. **Insert a new record into the `documents` table with the following fields: `file_name`, `file_ext`, `description`.**

Objective: If the query successfully inserted the entry into the documents table, we'll want to place the uploaded file in the **uploads/documents** folder.

- PHP places the uploaded file in a temporary location on your computer: `$_FILES["box_file"]["tmp_name"]`.
- You will want use the `move_uploaded_file( $_FILES["box_file"]["tmp_name"], $new_path )` function to move the temporary file to its permanent home: `$new_path = "uploads/documents/ID.FILE_EXTENSION"` (where ID is the primary key (`id`) of the inserted record and FILE_EXTENSION is the `file_ext` field).

Summary:

1. Insert a record into the database for the current upload.
2. Construct the new file name and path for the uploaded file to the **uploads/documents** folder.
    - You can get the ID of the last record inserted into the database using: `$db->lastInsertId("id");`
3. You need to use the `move_uploaded_file()` (http://php.net/manual/en/function.move-uploaded-file.php) helper function to place the uploaded file in the uploads/documents directory.
    - You will want to move the uploaded file, `$upload_info["tmp_name"]` to the file name and path you constructed above.

4. Test it!

Objective: Test uploading files. Check that documents table is properly updated. Check that the uploaded file is moved the **documents/uploads** folder. If you encounter any issues, fix them!

Feel free to discuss this with a peer just to make sure you both understand how this works.

Submit

Follow the instructions at the beginning of this document to submit your assignment.

1. Contributors

The following individuals made contributions to this assignment:

- Kyle Harms
- Sharon Jeong