WEB DEVELOPMENT WITH PHP

COURSE #09

fasttrackit
CLICK TO RESTART

LET'S RECAP

COURSE 8 WAS ABOUT:

- MySQL - CREATE TABLE
- MySQL - INSERT, UPDATE, DELETE
- MySQL - JOIN
- MySQL - Altering tables
- MySQL - Aggregate functions
- MySQL - Advanced queries

TOPICS

COURSE #09

1. Uploading files
2. Validating files
3. Processing files
4. Storing files
5. Using files

1. UPLOADING FILES

REMEMBER HTML FORMS?

TO UPLOAD FILES WE NEED TO USE HTML FORMS

HTML forms have a special field which allows users to browse their computer and select the file to upload.

When the form is submitted, the browser sends the file together with the other form fields to the server.

HOW TO CREATE A FILE UPLOAD FORM

1. ENCTYPE

To tell the browser that this form will contain file uploads, we need to add a special attribute when defining the form, named enctype.

```
<form enctype="multipart/form-data" [..] >
</form>
```

HOW TO CREATE A FILE UPLOAD FORM

2. METHOD

Make sure you set the form method to POST, as the files can't be send via GET.

```
<form method="post" [..] >
</form>
```

HOW TO CREATE A FILE UPLOAD FORM

3. ACTION

Make sure you set the form action to go to a PHP file which knows how to handle file uploads.

```
<form action="file-upload.php" [..] >
</form>
```

HOW TO CREATE A FILE UPLOAD FORM

4. INPUT TYPE="FILE"

Finally, add a file field to the form. We can add multiple file fields to the same form and also the form can contain other types of fields. For now, we recommend creating a separate form / page for uploading files.

```
<input type="file" name="filePhoto" />
```

CODING TIME

A FILE UPLOAD FORM (upload-form.php)

```
<html>
<body>
<form method="post" enctype="multipart/form-data" action="file-upload.php" name="
fileUpload">
     Student ID: <input type="text" name="studentId" /><br />
     Photo:<input type="file" name="filePhoto" /><br />
     <input type="submit" name="btnSubmit" value="Submit" />
</form>
</body>
</html>
```

PUTTING IT TOGETHER

IMPORTANT THINGS TO REMEMBER:

- ✓ enctype="multipart/form-data"
- ✓ method="post"
- ✓ action="file-upload.php"
- ✓ input type="file"

HTML FILE UPLOAD - ACTION

The action attribute is crucial. It means: *"Where do you want the form sent?"*

Let's create a new file, called `file-upload.php`:

```
<?php
    print "<pre>";
    var_dump($_POST);
    print "<hr>";
    var_dump($_FILES);
?>
```

HTML FILE UPLOAD - ACTION

LET'S LOOK AT THE OUTPUT

- ✓ What do you get in $_POST variable?
- ✓ What do you get in $_FILES variable?
- ✓ Where is the information about the file uploaded stored?

All the information about the uploaded file is stored in the $_FILES global variable.

WHAT'S IN $_FILES

`name`	file name of the uploaded file
`type`	the mime type of the uploaded file
`tmp_name`	the temporary location of the uploaded file (we need to copy it to a permanent location)
`error`	the error code; if no errors occurred during the upload, its value will be zero
`size`	the file size of uploaded file in bytes

WHAT'S NEXT

WHAT SHOULD WE DO NEXT WITH THE FILE?

- ✓ Validation
- ✓ Processing
- ✓ Storing for later access

2. VALIDATING FILES

VALIDATING UPLOAD ERROR CODES

If an error happens during upload, then we will find an error code in the $_FILES entry.

http://www.php.net/manual/en/features.file-upload.errors.php

```
if ($_FILES['filePhoto']['error'] == UPLOAD_ERR_PARTIAL) {
    $error = 'Partial update! Full file was not uploaded';
}
```

CONNECT TO MYSQL FROM PHP

First, let's connect to MySQL:

```
$server = 'localhost';
$user = 'root';
$pass = '';
$db_name = 'school';

$db_conn = mysqli_connect($server, $user, $pass, $db_name);
```

CODING TIME

In `file-upload.php`, test for file upload success. If success print "File uploaded." else print "Upload error: [error number]".

```
if ($_FILES['filePhoto']['error'] === 0) {
    print "File uploaded.";
    exit;
} else {
    print "Upload error: " . $_FILES['filePhoto']['error'];
    exit;
}
```

VALIDATING FILE TYPE

Ok, so the file is uploaded correctly. However, the user can upload a wrong file. For example, we asked for his profile photo and he uploaded a PDF document. We can find the file type in the $_FILES entry.

http://www.freeformatter.com/mime-types-list.html

```
if ($_FILES['filePhoto']['type'] != 'image/gif') {
    print “You are only allowed to upload gif images”;
}
```

CODING TIME

Update `file-upload.php` to only allow `gif`, png and `jpeg` file types.

```
if ($_FILES['filePhoto']['error'] === 0) {
     if ($_FILES['filePhoto']['type'] == 'image/gif'
       || $_FILES['filePhoto']['type'] == 'image/png'
       || $_FILES['filePhoto']['type'] == 'image/jpeg') {
          print "File uploaded.";
          die;
     }
     else {
          print "Only gif, png or jpeg files allowed";
          die;
     }
}
else {
    print "Upload error: " . $_FILES['filePhoto']['error'];
     exit;
}
```

VALIDATING EXTENSION

We can also check if the file has the correct extension.

Check the following output:

```
$path_parts = pathinfo($_FILES['filePhoto']['tmp_name']);
var_dump($path_parts);
```

How can we check if the file has .jpg extension for example ?

Documentation: http://php.net/manual/en/function.pathinfo.php

VALIDATING EXTENSION

How to check if an uploaded file has the .jpg extension.

```
$path_parts = pathinfo($_FILES['filePhoto']['tmp_name']);
if ($path_parts['extension'] != 'jpg') {
    print “You are only allowed to upload files with .jpg extension”;
}
```

OTHER VALIDATIONS

FILE SIZE

We might want to make sure files are smaller than a certain size.

```
$_FILES['uploadedFile']['size']
```

IMAGE DIMENSIONS

With images we might be interested in certain image dimensions

```
getimagesize()
```

3. PROCESSING FILES

PROCESSING FILES

PROCESSING THE FILE = MOVE FROM TEMP DIRECTORY

By default all files submitted through a form are uploaded in a temporary location, where they are stored pending processing.

WHY?

Because usually you want to do some validation on the file before actually storing it in a permanent location.

PROCESSING FILES

After we've validated the file, moving a file from temp to permanent location can be done with move_uploaded_file function.

```
move_uploaded_file($_FILES['filePhoto']['tmp_name'],
    "uploads/" . $_FILES['filePhoto']['name']);
```

PARAMETERS FOR move_uploaded_file()

- ✓ $filename - Current temporary path
- ✓ $destination - Destination path (*must exist and be writable*)

CODING TIME

Let's move the uploaded file from the temporary location to a permanent location, in a folder named "uploads".

WHAT ARE THE PARAMETERS?

```
$filename      $_FILES['filePhoto']['tmp_name']

$destination   'uploads/' . $_FILES['name']
```

CODING TIME

If we have no upload error and the file type is correct, let's move the file to its permanent location.

```
if ($_FILES['filePhoto']['type'] == 'image/gif'
  || $_FILES['filePhoto']['type'] == 'image/png'
  || $_FILES['filePhoto']['type'] == 'image/jpeg') {
    $filename = $_FILES['filePhoto']['tmp_name'];
    $destination = 'uploads/' . $_FILES['name'];
    if (move_uploaded_file($filename, $destination)) {
        print 'File moved successfully';
    }
    else {
        print 'Destination not writable, cannot move file.';
    }
[..]
```

4. STORING FILES

STORING FILES

We have so far uploaded, validated and saved a file to disk.

All we need is to remember where he have stored it in order to use it further on.

STORING FILES

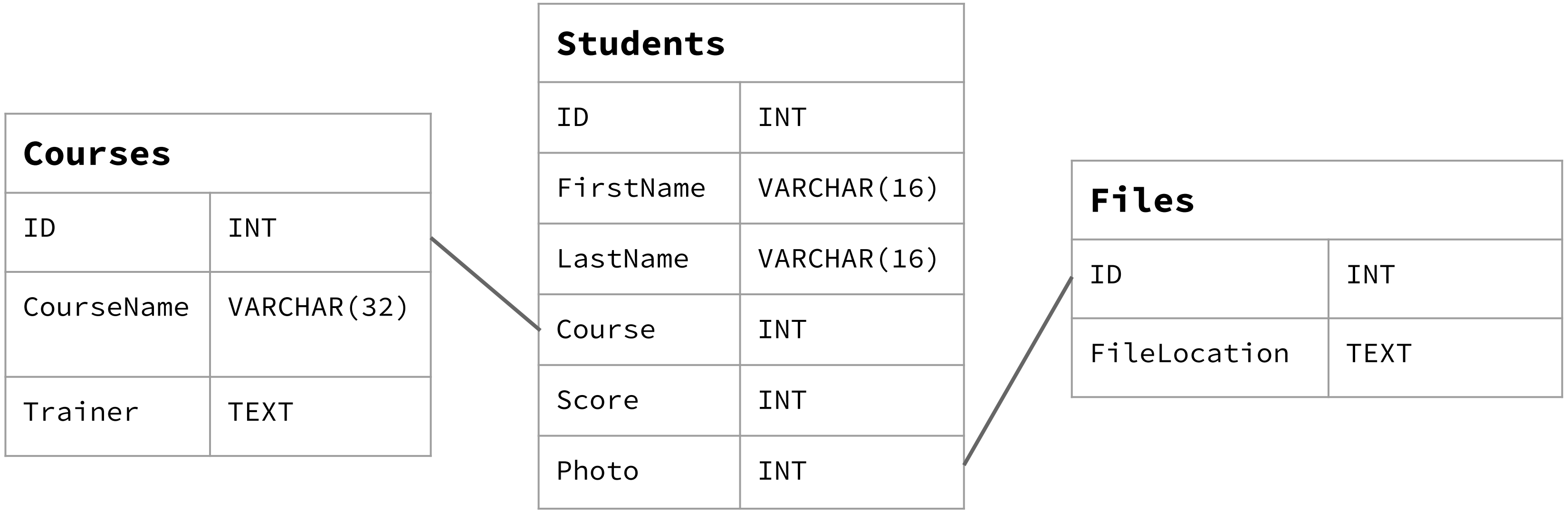
STEP 1: CREATE A TABLE FOR FILES

We'll add a photo to the students from last course. Let's create a table (named `Files`) in the last course's database `school` where to store the uploaded photos information, with the following fields:

`ID`	Auto-incremented ID
`FileLocation`	File path on disk

STORING FILES

LET'S DRAW IT



Courses	
ID	INT
CourseName	VARCHAR(32)
Trainer	TEXT

Students	
ID	INT
FirstName	VARCHAR(16)
LastName	VARCHAR(16)
Course	INT
Score	INT
Photo	INT

Files	
ID	INT
FileLocation	TEXT

CODING TIME

LET'S CREATE THE TABLES

```
CREATE TABLE Files (
    ID INT NOT NULL AUTO_INCREMENT,
    FileLocation TEXT NOT NULL,
    PRIMARY KEY (ID)
);
```

```
ALTER TABLE Students ADD Photo INT;
```

STORING FILES

STEP 2: INSERT FILE INFO INTO DATABASE

After uploading, validating and moving the files to a permanent location, we need to do three more things:

- insert the file information in the Files table
- retrieve the ID of the new entry
- store that ID in the Students table, for the correct student

CODING TIME

BUT BEFORE, LET'S MAKE SURE WE HAVE STUDENT ID

Check student ID before moving the file to its permanent location.

```
$student_id = $_POST['studentId'];
if (!student_id) {
    print 'Please provide a student ID.';
    exit;
}
if (move_uploaded_file($filename, $destination)) {
[..]
```

STORING FILES

INSERT THE FILE INFORMATION IN THE DATABASE

```
if (move_uploaded_file($filename, $destination)) {
  $sql = 'INSERT INTO Files (FileLocation) VALUES ("' . $destination . '")';
  mysqli_query($conn, $sql);
  if ($file_id = mysqli_insert_id($conn)) // Insert was successful
    $sql = 'UPDATE Students SET Photo = ' . $file_id . ' WHERE ID = ' . $student_id;
    mysqli_query($conn, $sql);
  }
}
```

5. USING FILES

USING FILES

LET'S SHOW THE STUDENT PHOTO

We'll create a new page, which displays a student's photo.

Let's name the page `student_photo.php`

When we open the page with a student ID in the URL, we want to display that student's photo.

Example: `../student_photo.php?ID=1`

USING FILES

HERE ARE THE STEPS

- ✓ Connect to DB
- ✓ Read the student ID from URL parameters
- ✓ Create a query to read the photo ID from the Students table
- ✓ Create a query to read the photo's FileLocation from Files table
- ✓ Display the photo as a HTML IMG tag

USING FILES

CONNECT TO DB

```
$server = 'localhost';
$user = 'root';
$pass = '';
$db_name = 'school';

$db_conn = mysqli_connect($server, $user, $pass, $db_name);
```

CONNECT TO MYSQL FROM PHP

READ THE STUDENT ID FROM URL

```php
if (isset($_GET['ID'])) {
    $student_id = $_GET['ID'];
}
else {
    print 'No student ID provided.';
    exit;
}
```

CONNECT TO MYSQL FROM PHP

READ THE PHOTO ID FROM STUDENTS TABLE

```
$sql = 'SELECT Photo FROM Students WHERE ID = ' . $student_id;
$result = mysqli_query($conn, $sql);
if ($result) {
    $row = mysqli_fetch_assoc($result);
    $photo_id = $row['Photo'];
}
else {
    print 'Error reading student info';
    exit;
}
```

CONNECT TO MYSQL FROM PHP

READ THE PHOTO LOCATION

```
$sql = 'SELECT FileLocation FROM Photos WHERE ID = ' . $photo_id;
$result = mysqli_query($conn, $sql);
if ($result) {
    $row = mysqli_fetch_assoc($result);
    $file_location = $row['FileLocation'];
}
else {
    print 'Error reading file info';
    exit;
}
```

CONNECT TO MYSQL FROM PHP

DISPLAY THE PHOTO

```
// close the PHP tag first
?>
<html>
<body>
<img src="<?php print $file_location; ?>">
</body>
</html>
```

6. HOMEWORK

HOMEWORK

STUDENTS MANAGEMENT PAGE

Similar with the homework from last week, you'll have to create some pages for managing the students list.

The site should:

- list all the students in the database
- allow you to add / edit / delete students
- allow you to upload a photo for the student
- allow you to upload a CV for the student

HOMEWORK

1. LIST STUDENTS

- Create a page called students.php
- Read all the students from the database using a SQL query
- Loop through all the students and put them in an array
- Loop through the array and create a HTML table
- Below the table add a link to a page called add_student.php

(see next page for a wireframe)

HOMEWORK

1. LIST STUDENTS

Student name	**Course**	**Operations**			
John Doe	Introduction in QA	Edit	Delete	Upload photo	Upload CV
Jane Doe	Web development with PHP	Edit	Delete	Upload photo	Upload CV

Add a new student

HOMEWORK

2. ADD STUDENT

- Create a page called add_student.php
- The page should contain a form with the fields
  - First name, Last name - Text fields
  - Course - Dropdown with list of courses from Courses table
  - Score - Text field
- On submit, verify that all fields are correctly filled
- Create a SQL query to insert the new student in the database
- Redirect to students.php

(see next page for a wireframe)

HOMEWORK

2. ADD STUDENT

Add student

First name	
Last name	
Course	▽
Score	

Add student

HOMEWORK

3. DELETE STUDENT

- Link the delete links to delete_student.php?ID=[student ID]
- Create a page called delete_students.php
- The page should accept a $_GET parameter called ID
- If the parameter is present, create a SQL query to delete the student having this parameter
- Redirect to students.php

HOMEWORK

4. EDIT STUDENT

- Link the edit links to edit_student.php?ID=[student ID]
- Create a page called edit_student.php
- The page should accept a $_GET parameter called ID
- If the parameter is present, create a SQL query to read the student details from database

HOMEWORK

4. EDIT STUDENT - CONTINUED

- Create a form identical with the one from Add student
- The fields should be pre-filled with the correct values for current student
- Also - a hidden field pre-filled with the student ID
- On submit, verify if the text fields are correctly filled
- Then create a SQL query which updates the record which has the ID from the hidden field with the values received by form
- Redirect to students.php

HOMEWORK

5. UPLOAD PHOTO

- Create a page called student_photo.php
- The page should accept a $_GET parameter called ID
- The page should contain a form with a file field
- Also - a hidden field pre-filled with the student ID
- On submit, verify that a file was correctly uploaded and that the file it's an image (jpg or png)
- Store the photo in the Students table (see example in current course)

HOMEWORK

6. UPLOAD CV

Similar with the upload photo page, create a page for uploading a CV for the students. The CV can be a DOC or PDF file.

7. VIEW STUDENT

Create a page to display ALL the information for a student, including the photo (as an IMG) and a link to download the CV.