

# SWE 363: Web Engineering & Development

Module 7-3

**Working with  
Database and Files**



# Objectives

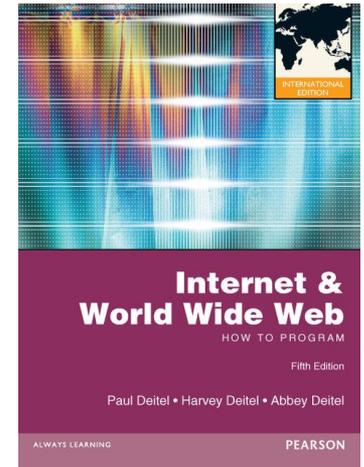
---

- ❑ Learn the role that databases play in web development
- ❑ Learn how to connect your web pages to database
- ❑ Learn how to manage files

- ❑ Connection to Database
- ❑ Connecting to MySQL through PHP
- ❑ Open a Connection to MySQL using [MySQLi](#)
- ❑ Create a MySQL Database Using [MySQLi](#)
- ❑ Create Tables Using [MySQLi](#)
- ❑ Inserting Data into a MySQL Database Table: [MySQLi](#)
- ❑ Selecting Data From Database Tables: [MySQLi](#)
- ❑ Working with Files in PHP

# References

- ❑ “Internet & World Wide Web: How to Program 5th editions”  
© Pearson Education
- ❑ “Fundamentals of Web Development” Book by Randy Connolly and Ricardo Hoar, 2015
- ❑ Lots of resources are available at <http://www.php.net>
  - Documentation: manual
    - <http://www.php.net/manual/en/>
    - <http://us2.php.net/manual/en/index.php>
  - Tutorials
    - <http://php.net/manual/en/tutorial.php>
  - Documented PHP functions
    - <http://us2.php.net/quickref.php>
- ❑ W3schools tutorial <http://www.w3schools.com/php/default.asp>
- ❑ Tutorial Public <https://www.tutorialrepublic.com/>



w3schools.com



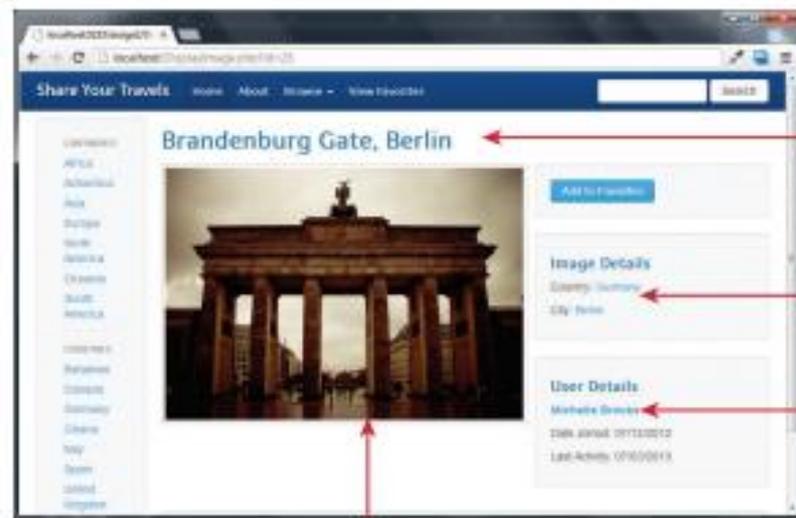
# Connecting to Database

# Databases and Web Development

- ❑ Almost every **dynamic website** makes use of server-based data source.
  - >> most common data source for these sites is a **database**.
- ❑ Many real-world sites make use of a **database server**, which is a computer that is devoted to **running a relational DBMS**.
  
- ❑ There are many relational DBMS
  - MySQL, PostgreSQL, Oracle Database, IBM DB2, and Microsoft SQL Server.
- ❑ All of these DBMS are capable of
  - managing large amounts of data,
  - maintaining data integrity,
  - responding to many queries,
  - creating indexes and triggers, and more.
  
- ❑ We will be using the DBMS **MySQL** - While the MySQL source code is **openly available**, it is now owned by Oracle Corporation.

# The role of DB in Web Development

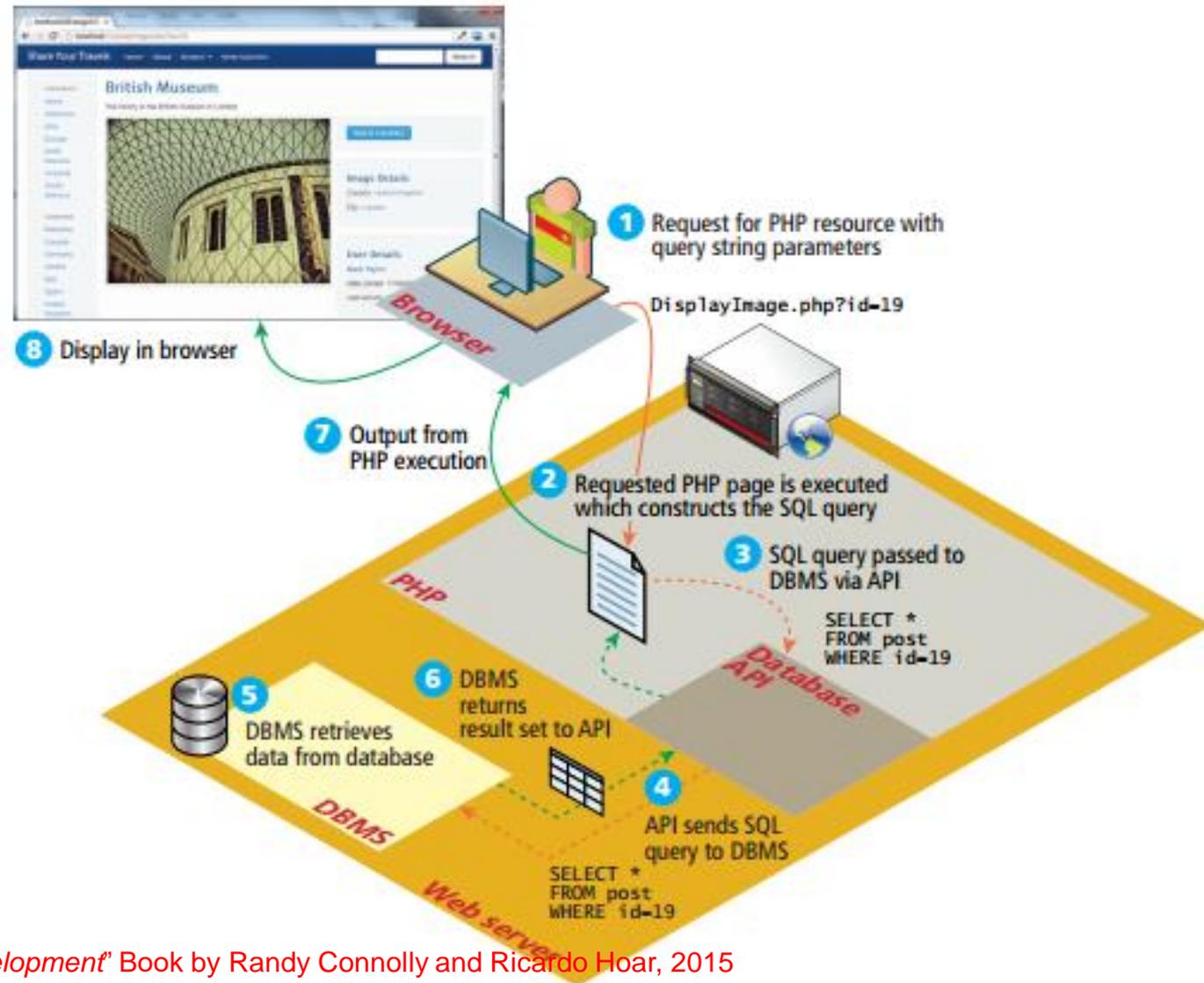
- ❑ Sites typically display **different content** on different pages.
- ❑ Pages **share similar user interface** elements
- ❑ **Separation** can be made by placing the content into a database, then “insert” the content into the markup
- ❑ When comparing with files, BD offer optimized systems



Content (data) varies but the markup (design) stays the same.

# How websites use databases

- ❑ The program (PHP) determines which data to display, often from information in the GET or POST query string, and then uses a database API to interact with the database



# Database APIs

- ❑ A server-side web technology such as PHP **interacts with the DBMS via a database API**, which refers to a programming interface to the features of the database system



- ❑ API refers to the classes, methods, functions, and variables that your application uses to perform some task.
- ❑ **Note:** Some database APIs work only with a specific type of database; others are cross-platform and can work with multiple databases.
- ❑ There are **two basic styles of database APIs** available in PHP:
  - **A procedural API** - uses function calls to work with the database
  - **An object-oriented API**- requires instantiating objects and invoking methods and properties

- ❑ With PHP, you **can connect to and manipulate databases**.
- ❑ **MySQL is the most popular database system used with PHP**.
- ❑ Although MySQL is free, it is used for large and busy websites. Some largest sites (like Facebook, Twitter, and Wikipedia) make use of some forms of MySQL
  
- ❑ **Why MySQL?**
  - MySQL is a database system that MySQL is easy to use, extremely powerful, fast, secure, and scalable.
  - MySQL **runs on a wide range of operating systems**, including UNIX or Linux, Microsoft Windows, Apple Mac OS X, and others.
  - MySQL **supports standard SQL** (Structured Query Language).
  - MySQL is **ideal database solution** for both small and large applications.
  - MySQL is developed, and distributed by Oracle Corporation.
  - MySQL **includes data security layers** that protect sensitive data
  - MySQL is free to download and use

# Connecting to MySQL through PHP

- ❑ PHP offers **two different ways to connect to MySQL server**:
  - **MySQLi** (Improved MySQL) and
  - **PDO** (PHP Data Objects) extensions
  
- ❑ MySQLi vs. PDO

MySQLi	PDO
<ul style="list-style-type: none"><li>• Provides an <b>easier way to connect</b> to, and execute queries on, a MySQL database server</li><li>• Offer an <b>object-oriented and procedural API</b></li><li>• Provides both <b>speed and feature benefits</b></li><li>• MySQLi extension supports <b>MySQL database only</b>.</li></ul>	<ul style="list-style-type: none"><li>• Offer an <b>object-oriented API</b></li><li>• PDO extension is <b>more portable</b> and supports more than twelve different databases</li></ul>

localhost/dashboard/

Apache Friends

Applications

FAQs

HOW-TO Guides

PHPInfo

phpMyAdmin

## XAMPP Apache + MariaDB + PHP + Perl

### Welcome to XAMPP for Windows 5.6.32

You have successfully installed XAMPP on this system! Now you can start using Apache, MariaDB, PHP and other components. You can find more info in the [FAQs](#) section or check the [HOW-TO Guides](#) for getting started with PHP applications.

XAMPP is meant only for development purposes. It has certain configuration settings that make it easy to develop locally but that are insecure if you want to have your installation accessible to others. If you want have your XAMPP accessible from the internet, make sure you understand the implications and you checked the [FAQs](#) to learn how to protect your site. Alternatively you can use [WAMP](#), [MAMP](#) or [LAMP](#) which are similar packages which are more suitable for production.

Start the XAMPP Control Panel to check the server status.

- ❑ A popular web-based front-end (written in PHP) called **phpMyAdmin** allows developers to access management tools through a web portal

# phpMyAdmin

The screenshot displays the phpMyAdmin web interface. The browser's address bar shows the URL `localhost/phpmyadmin/`. The top navigation bar features the 'Apache Friends' logo and several links: 'Applications', 'FAQs', 'HOW-TO Guides', 'PHPInfo', and 'phpMyAdmin' (which is circled in red). The main content area is titled 'Server: 127.0.0.1' and includes a navigation menu with options like 'Databases', 'SQL', 'Status', 'User accounts', 'Export', 'Import', 'Settings', 'Replication', 'Variables', 'Charsets', and 'Engines'. The 'General settings' section shows 'Server connection collation' set to 'utf8mb4\_unicode\_ci'. The 'Appearance settings' section shows 'Language' set to 'English' and 'Theme' set to 'pmahomme'. The right sidebar contains information about the 'Database server' and 'Web server'. The 'Database server' section lists details such as 'Server: 127.0.0.1 via TCP/IP', 'Server type: MariaDB', 'Server version: 10.1.28-MariaDB - mariadb.org binary distribution', 'Protocol version: 10', 'User: root@localhost', and 'Server charset: UTF-8 Unicode (utf8)'. The 'Web server' section lists details such as 'Apache/2.4.29 (Win32) OpenSSL/1.0.2l PHP/5.6.32', 'Database client version: libmysql - mysqlnd 5.0.11-dev - 20120503 - \$Id: 76b08b24596e12d4553bd41fc93cccd5bac2fe7a \$', 'PHP extension: mysqli curl mbstring', and 'PHP version: 5.6.32'. The 'phpMyAdmin' section lists details such as 'Version information: 4.7.4, latest stable version: 4.7.6', 'Documentation', 'Official Homepage', 'Contribute', and 'Get support'.

- ❑ A phpMyAdmin provides a clickable interface that lets you navigate your databases more intuitively.

# Connecting to MySQL using PHP

- ❑ Before starting the work with the DB either by creating tables, inserting data, or running queries; **you need to set up a connection to the relevant database**
  
- ❑ With MySQL databases, we have to supply the following information when making a database connection:
  1. the host or URL of the **database server**,
  2. the **database name**, and
  3. the database **user name** and **password**.

# Connecting to MySQL using PHP

## ❑ `mysqli_connect` connects to the MySQL database

```
$connection= mysqli_connect("hostname", "username", "password", "database");
```

- It takes three parameters:
  - The **hostname** parameter (e.g. localhost), or IP address of the MySQL server,
  - The **username** and **password** parameters >> the credentials to access MySQL server,
  - The **database** parameter, if provided will specify the default MySQL database to be used when performing queries.
- It returns a database handle (a representation of PHP's connection to the database), or false if the connection fails

**Note:** the default username for MySQL database server is **root** and there is no password

## ❑ Connect to the MySQL database using **PDO**

```
$pdo = new PDO("mysql:host=hostname;dbname=database", "username", "psw");
```

PDO Tutorial for MySQL Developers:

<http://www.webcarpenter.com/blog/52-PDO-Tutorial-for-MySQL-Developers>

# Connecting to MySQL using PHP

- ❑ Best Practice

```
// modify these variables for your installation  
$host = "localhost";  
$database = "bookcrm";  
$user = "testuser";  
$pass = "mypassword";  
  
$connection = mysqli_connect($host, $user, $pass, $database);
```

- ❑ Define connection details via constants in a separate file (e.g. config.php)

```
<?php  
define('DBHOST', 'localhost');  
define('DBNAME', 'bookcrm');  
define('DBUSER', 'testuser');  
define('DBPASS', 'mypassword');  
?>
```

- ❑ Use the require\_once() function

```
require_once('protected/config.php');  
$connection = mysqli_connect(DBHOST, DBUSER, DBPASS, DBNAME);
```

# Handling connection errors

- ❑ Not every database connection always works.

```
$connection = mysqli_connect(DBHOST, DBUSER, DBPASS, DBNAME);  
  
// mysqli_connect_error returns string description of the last  
// connect error  
$error = mysqli_connect_error();  
if ($error != null) {  
    $output = "<p>Unable to connect to database<p>" . $error;  
    // Outputs a message and terminates the current script  
    exit($output);  
}
```

```
$connection = mysqli_connect(DBHOST, DBUSER, DBPASS, DBNAME);  
  
// mysqli_connect_errno returns the last error code  
if ( mysqli_connect_errno() ) {  
    die( mysqli_connect_error() ); // die() is equivalent to exit()  
}
```

# Close a Connection to MySQL using MySQLi

## To Close a Connection

- The connection to the MySQL database server **will be closed automatically** as soon as the execution of the script ends.
- However, if you want to close it earlier you can do this by simply calling the PHP `mysqli_close()` function.

```
mysqli_close($connection);
```

# Create a MySQL Database Using MySQLi

```
<?php
```

```
/* Attempt MySQL server connection. Assuming you are running MySQL  
server with default setting (user 'root' with no password) */
```

```
$link = mysqli_connect("localhost", "root", "");
```

```
// Check connection
```

```
if($link === false){
```

```
die("ERROR: Could not connect. " . mysqli_connect_error());}
```

```
// Attempt create database query execution
```

```
$sql = "CREATE DATABASE demo";
```

```
if(mysqli_query($link, $sql)){ // performs a query against the database
```

```
echo "Database created successfully";
```

```
} else{
```

```
echo "ERROR: Could not able to execute $sql. " . mysqli_error($link);
```

```
}
```

```
// Close connection
```

```
mysqli_close($link);
```

```
?>
```

# Create a Table Using MySQLi

```
<?php
    /* default setting (user 'root' with no password) */
    $link = mysqli_connect("localhost", "root", "", "demo");
    // Check connection

    // Attempt create table query execution
    $sql = "CREATE TABLE persons (
        id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
        first_name VARCHAR(30) NOT NULL,
        last_name VARCHAR(30) NOT NULL,
        email VARCHAR(70) NOT NULL UNIQUE
    )";

    if(mysqli_query($link, $sql)){
        echo "Table created successfully.";
    } else{
        echo "ERROR: Could not able to execute $sql. " . mysqli_error($link);
    }
    // Close connection
    mysqli_close($link);
?>
```

# Inserting Data into a MySQL Database Table: MySQLi

```
<?php
/* default setting (user 'root' with no password) */
$link = mysqli_connect("localhost", "root", "", "demo");
// Check connection

// Attempt insert query execution
$sql = "INSERT INTO persons
      (first_name, last_name, email) VALUES
      ('John', 'Rambo', 'johnrambo@mail.com'),
      ('Peter', 'Parker', 'peterparker@mail.com')";

if(mysqli_query($link, $sql)){
echo "Records inserted successfully.";
} else{
echo "ERROR: Could not able to execute $sql. " . mysqli_error($link);
}

// Close connection
mysqli_close($link);

?>
```

# Insert Data into a Database from an HTML Form: MySQLi

```
<form action="insert.php" method="post">
  <p> First Name: <input type="text" name="first_name" > </p>
  <p> Last Name: <input type="text" name="last_name" > </p>
  <p> Email Address:<input type="text" name="email" > </p>
  <input type="submit" value="Submit">
</form>
```

## Person information

First Name:

Last Name:

Email Address:

# Insert Data into a Database from an HTML Form: MySQLi

```
<?php
$link = mysqli_connect("localhost", "root", "", "demo");
// Check connection

// Escape user inputs for security
$first_name = mysqli_real_escape_string($link, $_POST['first_name']);
$last_name = mysqli_real_escape_string($link, $_POST['last_name']);
$email = mysqli_real_escape_string($link, $_POST['email']);

// attempt insert query execution
$sql = "INSERT INTO persons (first_name, last_name, email) VALUES
('$first_name', '$last_name', '$email')";

if(mysqli_query($link, $sql)){
echo "Records added successfully.";
} else{
echo "ERROR: Could not able to execute $sql. " . mysqli_error($link);
}
// close connection
mysqli_close($link);
?>
```

'insert.php'

The `mysqli_real_escape_string()` function escapes special characters in a string for use in an SQL statement. Especial characters, such as `\x00`, `\n`, `\r`, `\`, `'`, `"` and `\x1a`

# Selecting Data From Database Tables: MySQLi

```
<?php
$link = mysqli_connect("localhost", "root", "", "demo");
// Attempt select query execution
$sql = "SELECT * FROM persons";
if($result = mysqli_query($link, $sql)){
if(mysqli_num_rows($result) > 0){
    echo "<table>"; echo "<tr>";
    echo "<th>id</th>"; echo "<th>first_name</th>";
    echo "<th>last_name</th>"; echo "<th>email</th>"; echo "</tr>";

    while($row = mysqli_fetch_array($result)){
    echo "<tr>";
    echo "<td>" . $row['id'] . "</td>";
    echo "<td>" . $row['first_name'] . "</td>";
    echo "<td>" . $row['last_name'] . "</td>";
    echo "<td>" . $row['email'] . "</td>";
    echo "</tr>";
    }
    echo "</table>";
mysqli_free_result($result); // Free result set

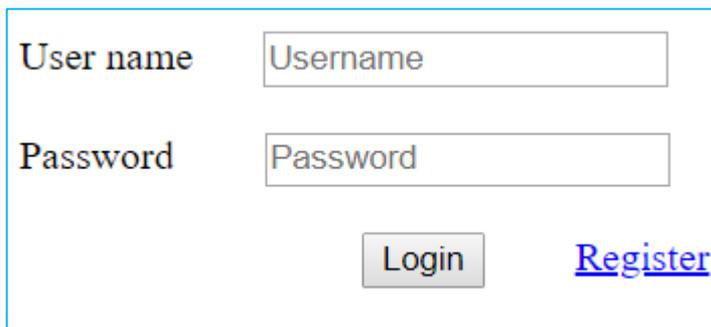
} else{ echo "No records matching your query were found."; }
} else{
echo "ERROR: Could not able to execute $sql. " . mysqli_error($link);
}
// Close connection
mysqli_close($link);
```

id	first_name	last_name	email
3	Clark	Kent	clarkkent@mail.com
5	Harry	Potter	harrypotter@mail.com
2	John	Rambo	johnrambo@mail.com
4	John	Carter	johncarter@mail.com
1	Peter	Parker	peterparker@mail.com

# Simple Login Script in PHP and MySQL

## 1. Creating HTML User Login Form

```
<form action= "home.php" method="POST">
  <h2> Please Login</h2>
  <div>
    <span id="basic-addon1">@</span>
    <input type="text" name="username" placeholder="Username" required>
  </div>
  <label for="inputPassword"> Password</label>
  <input type="password" name="password" placeholder="Password" required>
  <button type="submit">Login</button>
  <a href="register.php">Register</a>
</form>
```



User name

Password

[Register](#)

# Simple Login Script in PHP and MySQL

## 2. Connect to Database

```
<?php
$connection = mysqli_connect('localhost', 'root', '', '');
if (!$connection){
    die("Database Connection Failed" . mysqli_error($connection));
}
$select_db = mysqli_select_db($connection, 'DBname');
if (!$select_db){
    die("Database Selection Failed" . mysqli_error($connection));
}
```

connect.php

### 3. PHP Logic for User Login

home.php

```
<?php
session_start(); //Start the Session
require('connect.php');
if (isset($_POST['username']) and isset($_POST['password']))
{
    $username = $_POST['username'];
    $password = $_POST['password'];
    // Checking the values are existing in the database or not
    $query = "SELECT * FROM user WHERE username='$username' and
password='$password'";
    $result = mysqli_query($connection, $query) or
        die(mysqli_error($connection));
    $count = mysqli_num_rows($result);
    if ($count == 1){
        $_SESSION['username'] = $username;
    }else{// the login credentials doesn't match
        $fmsg = "Invalid Login Credentials.";
    }
}
// if the user is logged in Greets the user with message
if (isset($_SESSION['username'])) {
    $username = $_SESSION['username'];
    echo "Welcome " . $username . " ";
    echo "This is the Members Area";
    echo "<a href='logout.php'>Logout</a>";
}

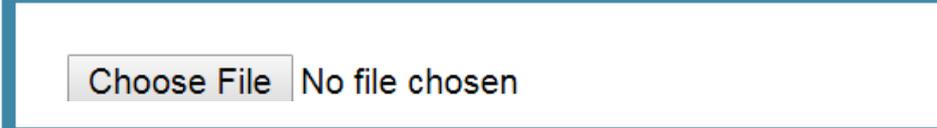
```

# Working with Files in PHP

# \$\_FILES array

- ❑ The **\$\_FILES** associative array contains items that have been uploaded to the current script.

```
1 | <input name="myFile" type="file">
```



Choose File No file chosen

- ❑ In **HTML**, the **<input type="file">** element is used to create the user interface for uploading a file from the client to the server.
  - However, the user interface is only one part of the uploading process.
  - A server script must process the upload file(s) in some way; >> the **\$\_FILES** array helps in this process.

# HTML required for File Uploads

To allow users to upload files, there are some specific things you must do:

- ❑ **First**, you must ensure that **the HTML form uses the HTTP POST method**,
  - transmitting a file through the URL is not possible.
- ❑ **Second**, you **must add the `enctype="multipart/form-data"` attribute** to the HTML form that is performing the upload
  - so that the HTTP request can submit multiple pieces of data (namely, the HTTP post body, and the HTTP file attachment itself).
- ❑ **Third**, you **must include an input type of file in your form**.
  - This will show up with a browse button beside it so the user can select a file from their computer to be uploaded.

```
<form enctype='multipart/form-data' method='post'>
  <input type='file' name='file1' id='file1' />
  <input type='submit' />
</form>
```

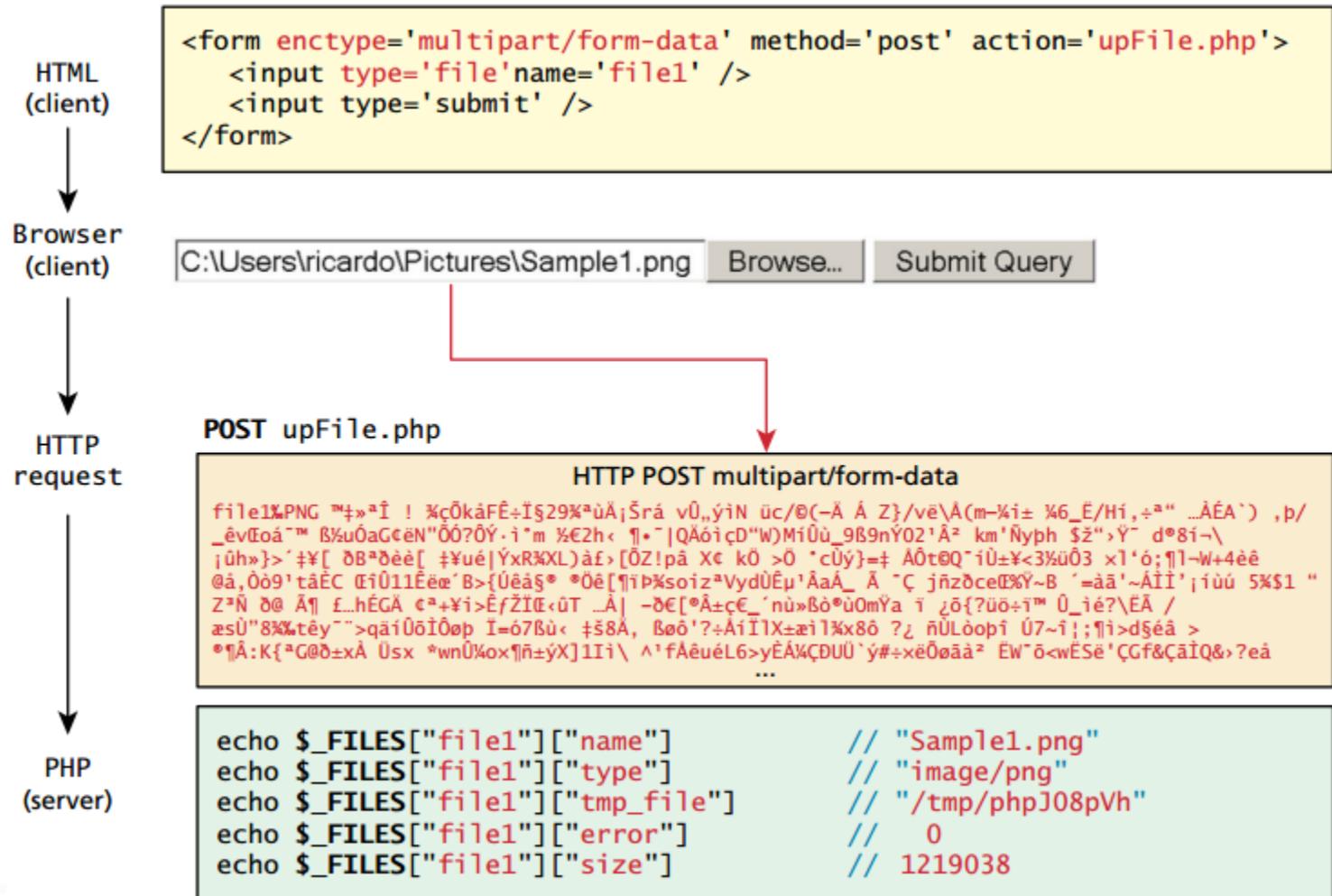
Choose File No file chosen

Submit

# Handling the File Upload in PHP

- ❑ The superglobal `$_FILES` array contain a **key=value pair** for each file uploaded in the post.
  - The key for each element will be the name attribute from the HTML form,
  - The value will be an array containing information about the file as well as the file itself.
- ❑ The keys in that array are the:
  - **name** is a string containing **the full file name + file extension** used on the client side (no path)
  - **type** defines the MIME type of the file.
  - **tmp\_name** is the full path to the location on your server where the file is being temporarily stored. The file will cease to exist upon termination of the script, so it should be copied to another location if storage is required.
  - **error** is an integer that encodes many possible errors and is set to UPLOAD\_ERR\_OK (integer value 0) if the file was uploaded successfully.
  - **size** is an integer representing the size in bytes of the uploaded file.

# Handling the File Upload in PHP



# Checking for errors

- ❑ For every uploaded file, there is an error value associated with it in the `$_FILES` array. The error values are specified **using constant values**, which resolve to integers.

Error Code	Integer	Meaning
<code>UPLOAD_ERR_OK</code>	0	Upload was successful.
<code>UPLOAD_ERR_INI_SIZE</code>	1	The uploaded file exceeds the <code>upload_max_filesize</code> directive in <code>php.ini</code> .
<code>UPLOAD_ERR_FORM_SIZE</code>	2	The uploaded file exceeds the <code>max_file_size</code> directive that was specified in the HTML form.
<code>UPLOAD_ERR_PARTIAL</code>	3	The file was only partially uploaded.
<code>UPLOAD_ERR_NO_FILE</code>	4	No file was uploaded. Not always an error, since the user may have simply not chosen a file for this field.
<code>UPLOAD_ERR_NO_TMP_DIR</code>	6	Missing the temporary folder.
<code>UPLOAD_ERR_CANT_WRITE</code>	7	Failed to write to disk.
<code>UPLOAD_ERR_EXTENSION</code>	8	A PHP extension stopped the upload.

# Checking for errors

- ❑ Checking each file uploaded for errors

```
foreach ($_FILES as $fileKey => $fileArray) {  
    if ($fileArray["error"] != UPLOAD_ERR_OK) { // error  
        echo "Error: " . $fileKey . " has error" . $fileArray["error"]  
            . "<br>";  
    }  
    else { // no error  
        echo $fileKey . "Uploaded successfully ";  
    }  
}
```

# File size restrictions

- ❑ Some scripts limit the file size of each upload.
- ❑ Three main mechanisms for maintaining uploaded file size restrictions: via **HTML** in the input form, via **JavaScript** in the input form, and via **PHP** coding.

```
<form enctype='multipart/form-data' method='post'>
  <input type="hidden" name="MAX_FILE_SIZE" value="1000000" />
  <input type='file' name='file1' />
  <input type='submit' />
</form>
```

HTML

Note that:

- Add a hidden input field **before** any other input fields in your HTML form with a name of MAX\_FILE\_SIZE.
- It should be noted that though this mechanism is set up in the HTML form, it is **only available to use when your server-side environment is using PHP.**

# File size restrictions..

- ❑ The more complete **client-side mechanism** is to pre-validate the form using JavaScript

```
<script>
var file = document.getElementById('file1');
var max_size = document.getElementById("max_file_size").value;
if (file.files && file.files.length ==1){
    if (file.files[0].size > max_size) {
        alert("The file must be less than " + (max_size/1024) + "KB");
        e.preventDefault();
    }
}
</script>
```

# File size restrictions..

- ❑ The third (and essential) mechanism for limiting the uploaded file size is to add a **simple check on the server side**
  - just in case JavaScript was turned off or
  - the user modified the MAX\_FILE\_SIZE hidden field
- ❑ This technique **checks the file size on the server** by simply **checking the size field in the \$\_FILES array**.

```
$max_file_size = 10000000;  
foreach($_FILES as $fileKey => $fileArray) {  
    if ($fileArray["size"] > $max_file_size) {  
        echo "Error: " . $fileKey . " is too big";  
    }  
    printf("%s is %.2f KB", $fileKey, $fileArray["size"]/1024);  
}
```

# Limiting the type of File Upload

- ❑ To accomplish this type of checking you typically examine the file extension and the type field.

```
$validExt = array("jpg", "png");
$validMime = array("image/jpeg","image/png");
foreach($_FILES as $fileKey => $fileArray ){
    $extension = end(explode(".", $fileArray["name"]));
    if (in_array($fileArray["type"],$validMime) &&
        in_array($extension, $validExt)) {
        echo "all is well. Extension and mime types valid";
    }
    else {
        echo $fileKey." Has an invalid mime type or extension";
    }
}
```

# Working with Files in PHP

- ❑ PHP has several functions for **creating**, **reading**, **uploading**, and **editing** files.
- ❑ PHP **fopen()** and **fclose()** Functions

```
<?php
    $file = "data.txt";
    // Check the existence of file
    if(file_exists($file)){
        // Open the file for reading
        $handle = fopen($file, "r") or die("ERROR: Cannot open the
        file.");
        /* Some code to be executed */
        // Closing the file handle
        fclose($handle);
    } else{
        echo "ERROR: File does not exist.";
    }
?>
```

# Working with Files in PHP

- ❑ The easiest way to read the entire contents of a file in PHP is with the `readfile()` function.

```
<?php
    $file = "data.txt";
    // Check the existence of file
    if(file_exists($file)){
        // Reads and outputs the entire file
        readfile($file) or die("ERROR: Cannot open the file.");
    } else{
        echo "ERROR: File does not exist.";
    }
?>
```

# Working with Files in PHP

## ❑ Reading from Files with PHP `fread()` Function

- It can be used to read a specified number of characters from a file.

```
fread(file_handle, length in bytes)
```

*Two parameter — A file handle and the number of bytes to read.*

- It can be used in conjugation with the `filesize()` function to read the entire file at once. The `filesize()` function returns the size of the file in bytes.

```
$content = fread(file_handle, filesize(file));
```

```
<?php
$file = "data.txt";
if(file_exists($file)){
    $handle = fopen($file, "r") or die("ERROR: Cannot open the
    file.");
    $content = fread($handle, "20");
    fclose($handle);
    echo $content;
} ?>
```

# Working with Files in PHP

- ❑ Another way to read the whole contents of a file **without needing to open it** is with the `file_get_contents()` function.
  - Reads the entire file into a string variable
- ❑ This function accepts the **name** and **path** to a file, and reads the entire file into a **string variable**.

```
<?php
    $file = "data.txt";
    if(file_exists($file)){
        // Reading the entire file into a string
        $content = file_get_contents($file) or die("ERROR: Cannot
        open the file.");
        echo $content;
    } else{
        echo "ERROR: File does not exist.";
    }
?>
```

# Working with Files in PHP

- ❑ Writing the Files Using PHP `fwrite()` Function

```
fwrite(file handle, string)
```

- ❑ An alternative way is using the `file_put_contents()` function.

```
file_put_contents($file, $data)
```

- ❑ If the file specified in the `file_put_contents()` function already exists, PHP will overwrite it by default. So, to append the new data you can use:

```
file_put_contents($file, $data, FILE_APPEND)
```

- ❑ You can rename a file or directory using the PHP's `rename()` function

```
$file = "file.txt";  
if(file_exists($file)){  
    // Attempt to rename the file  
    if(rename($file, "newfile.txt")){  
        echo "File renamed successfully."  
    }  
    ...  
}
```

# Working with Files in PHP

- ❑ You can delete files or directories using the PHP's `unlink()` function

```
$file = "note.txt";  
if(file_exists($file)){  
    // Attempt to delete the file  
    if(unlink($file)){  
        echo "File removed successfully."  
    } ...
```

# Working with Files in PHP

## ❑ PHP Filesystem Functions

Function	Description
<code>fgetc()</code>	Reads a single character at a time.
<code>fgets()</code>	Reads a single line at a time.
<code>fgetcsv()</code>	Reads a line of comma-separated values.
<code>filetype()</code>	Returns the type of the file.
<code>feof()</code>	Checks whether the end of the file has been reached.
<code>is_file()</code>	Checks whether the file is a regular file.
<code>is_dir()</code>	Checks whether the file is a directory.
<code>is_executable()</code>	Checks whether the file is executable.
<code>realpath()</code>	Returns canonicalized absolute pathname.
<code>rmdir()</code>	Removes an empty directory.

# Working with Directories in PHP

## ❑ Creating a New Directory

- You can create a new and empty directory by calling the PHP `mkdir()` function

```
$dir = "testdir";  
if(!file_exists($dir)){  
    // Attempt to create directory  
    if(mkdir($dir)){  
        echo "Directory created successfully."  
    }  
    .....
```

## ❑ Copying Files from One Location to Another

- You can copy a file from one location to another by calling PHP `copy()` function

```
$file = "example.txt"; // Source file path  
$newfile = "backup/example.txt"; // Destination file path  
if(file_exists($file)){ // Check the existence of file  
    if(copy($file, $newfile)){ // Attempt to copy file  
        echo "File copied successfully."  
    }  
    .....
```

## ❑ rmdir — Removes directory

## ❑ scandir – List files and directories inside the specified path

