Advanced PHP

ITCS 210 Web Programming

Class Objectives

- Students would be able to know:
  - Concept of Cookie and Session
  - Environmental Variables
  - File, File Upload

Review Basic PHP: How it works?

```
1.<html>
2.    <head><title>PHP Page</title></head>
3.    <body>
4.        14:42-10/30/12<br/>
5.        <table border="1">
6.          <tr>
7.             <td>0</td>
8.             <td>1</td>
9.             <td>2</td>
10.         </tr>
11.       </table>
12.    </body>
13.</html>
```

test.php

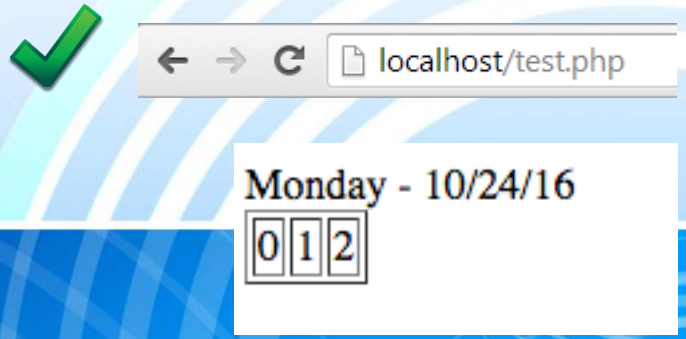
```
1.<html>
2.    <head><title>PHP Page</title></head>
3.    <body>
4.        <?php echo date("G:i-m/d/y")."<br/>"; ?>
5.        <table border="1"><tr>
6.        <?php
7.            for($i=0;$i<3;$i++)
8.                echo "<td>".$i."</td>";
9.        ?>
10.       </tr></table>
11.    </body>
12.</html>
```

Review Basic PHP: How it works?

test.php

```
1.<html>
2.    <head><title>PHP Page</title></head>
3.    <body>
4.        <?php echo date("l-m/d/y")."<br/>";?>
5.        <table border="1"><tr>
6.        <?php
7.            for($i=0;$i<3;$i++)
8.                echo "<td>".$i."</td>";
9.        ?>
10.       </tr></table>
11.    </body>
12.</html>
```

C:\root directory
e.g. wamp64/www/… or xampp/htdocs



Form Handling

Vs

Session

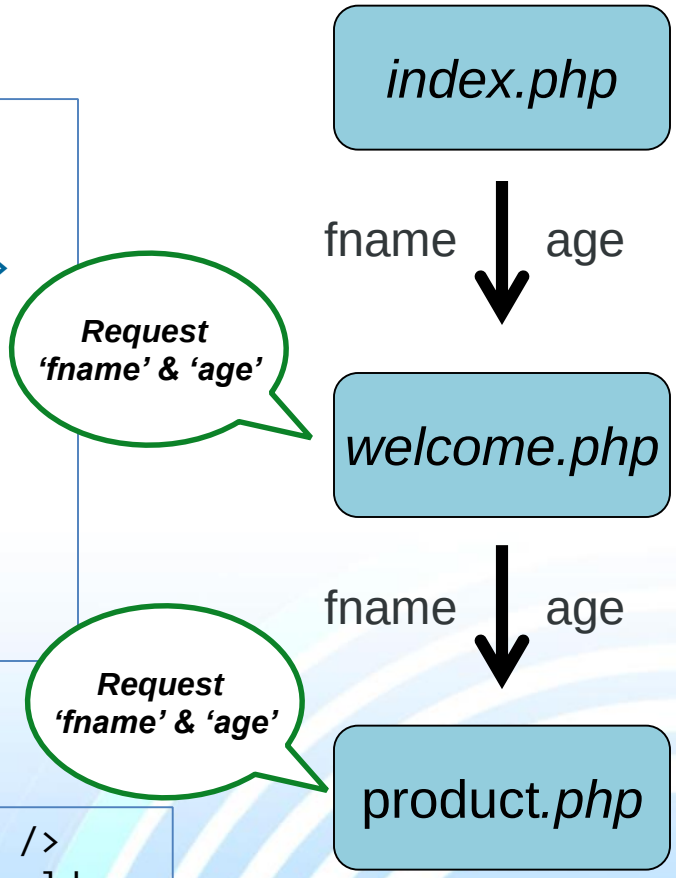
Form Handling

- Any HTML form element will **automatically** be available to PHP scripts.
- User input should be validated on the browser (client) whenever possible
  - Traditional & Customization ➔ JS
  - Modern ➔ HTML5
- Server validation is considered if you intends to insert data to the database.

Form Handling (Cont)

index.php

```
1. <html>
2. <body>
3.
4. <form action="welcome.php" method="post">
5. Name: <input type="text" name="fname" />
6. Age: <input type="text" name="age" />
7. <input type="submit" />
8. </form>
9.
10.</body>
11.</html>
```



welcome.php

```
1.Welcome <?php echo $_POST["fname"]; ?>!<br />
2.You are <?php echo $_POST["age"]; ?> years old.
```

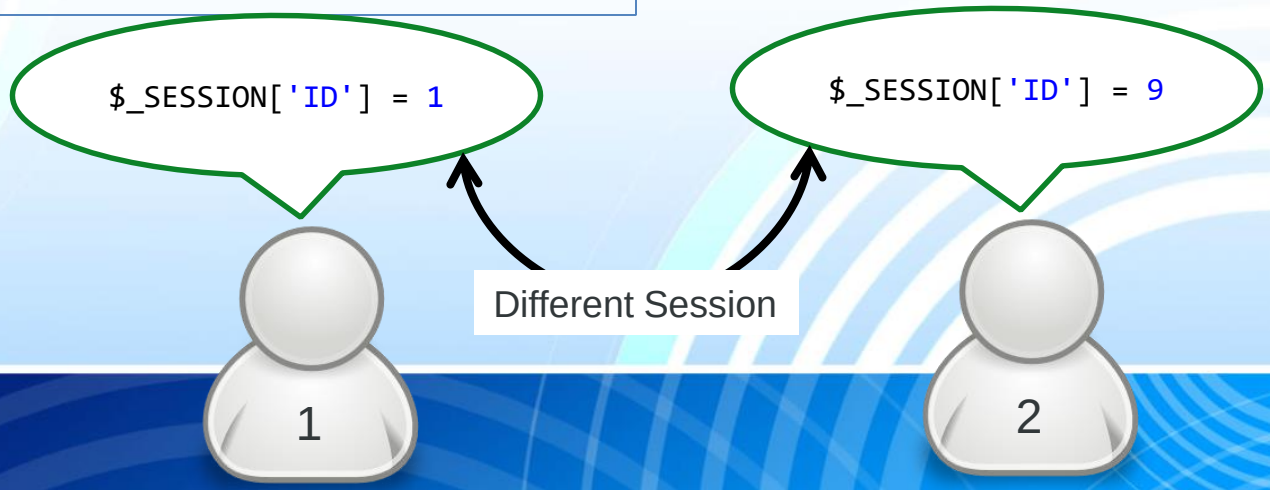
$_POST or $_GET

Session

- **<u>Temporary</u>** store user information on a server
- Session variables hold information about one **<u>single user</u>**, and are available to **<u>all pages</u>** in one application.

myPage.php

```
1.<?php
2.session_start(); // store session data
3.$_SESSION['ID'] = $userID;
4.?>
```



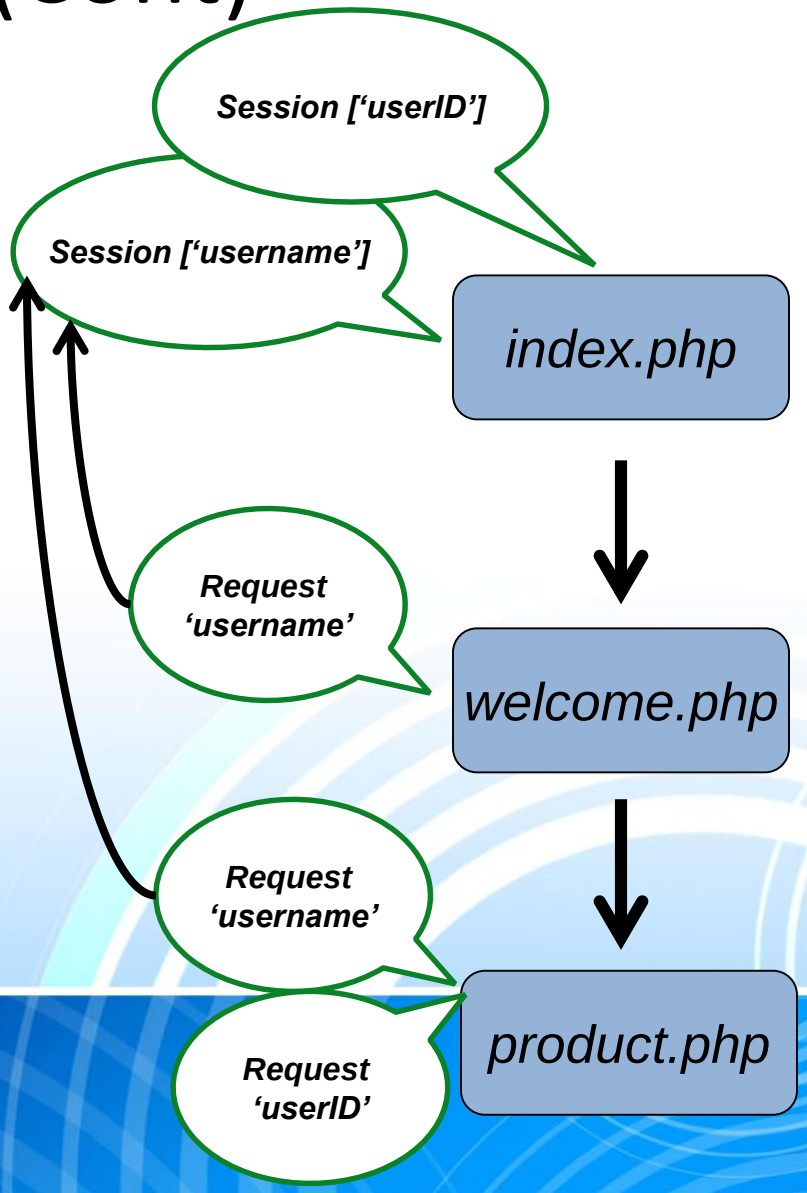
Session (Cont)

index.php

```
1.<?php
2.session_start(); // store session data
3.$_SESSION['username'] = "John";
4.?>
```

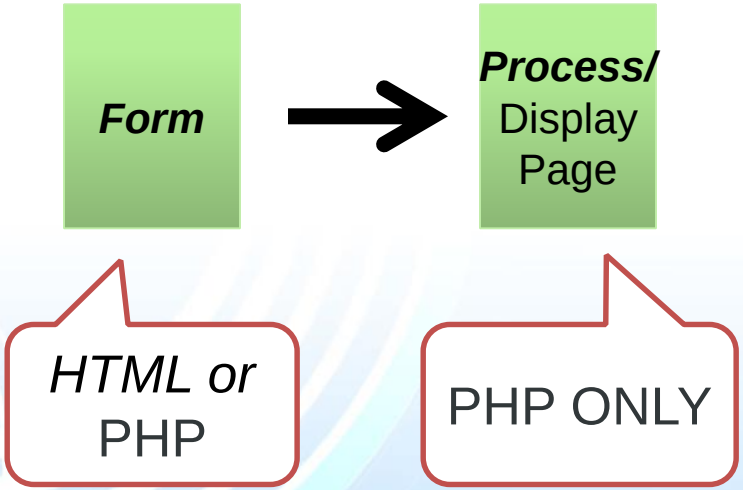
welcome.php

```
1.<?php session_start(); ?>
2.<html>
3.<body>
4.
5.<?php  //retrieve session data
6.echo “Welcome:". $_SESSION[‘username'];
7.?>
8.
9.</body>
10.</html>
```

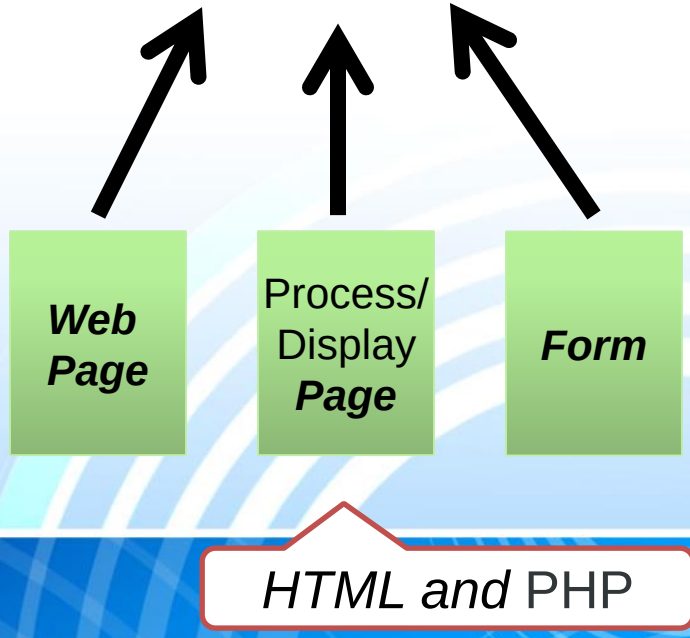


Review Basic PHP (Cont)

PHP Form Handling

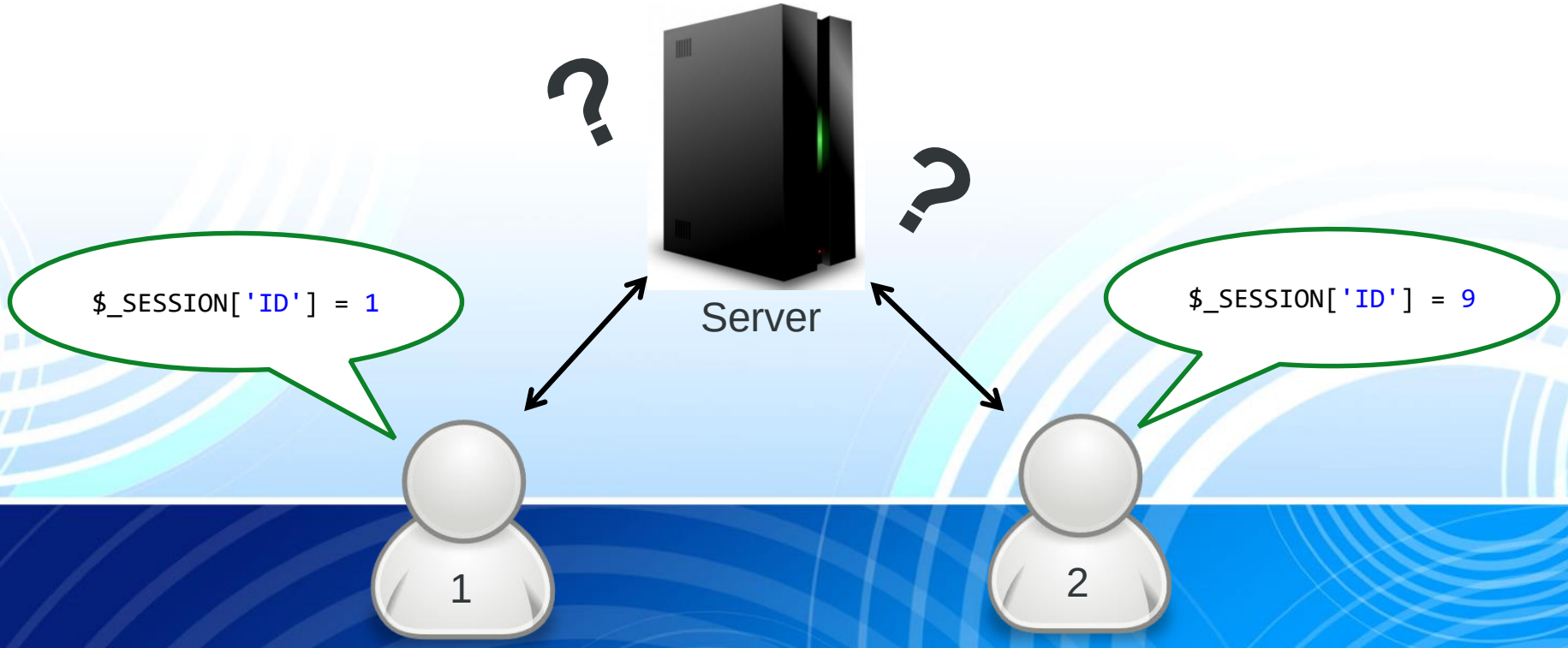


PHP Session



Concept of Cookie and Session

- HTTP (HyperText Transfer Protocol) lacks of association between any two HTTP requests. Because the protocol does not provide any method that the client can use to identify itself, the server cannot distinguish between clients.

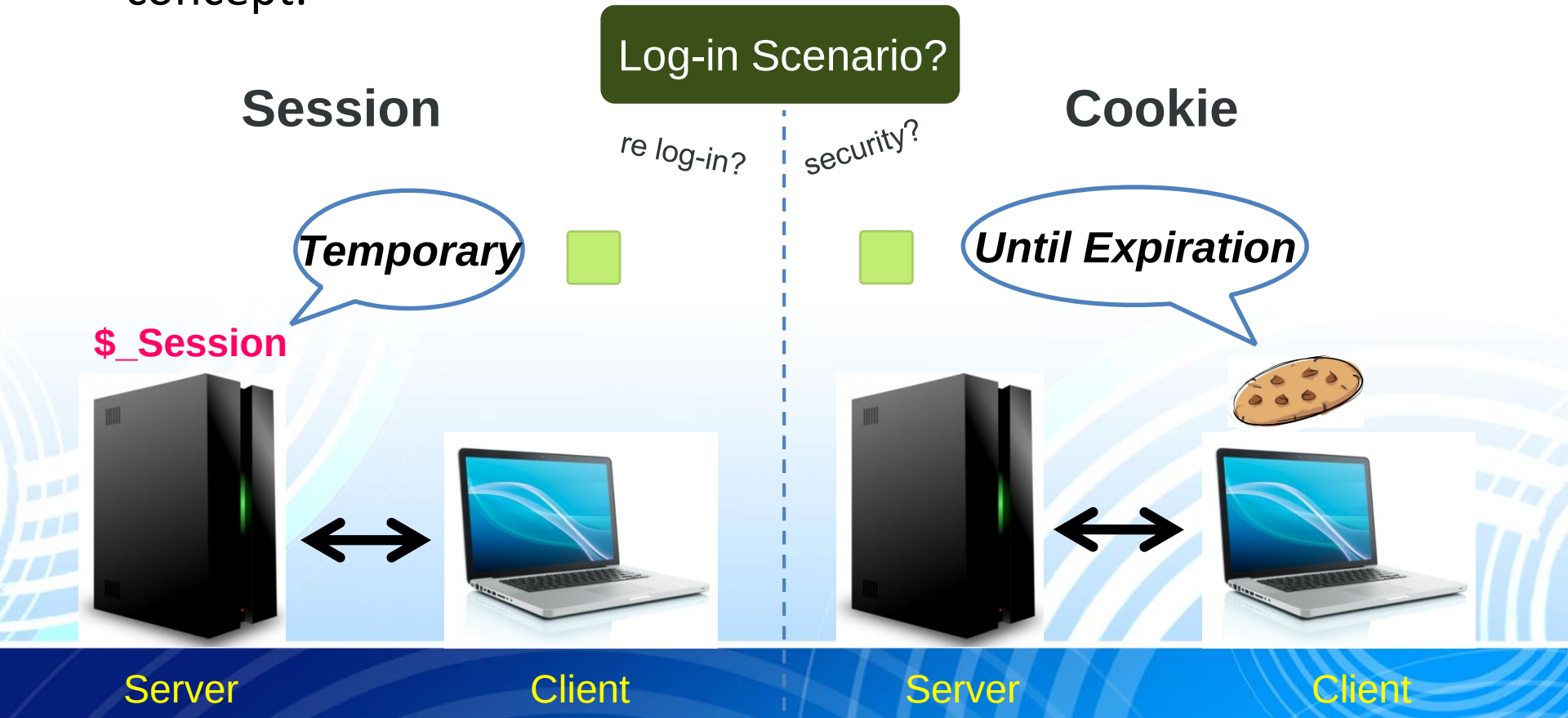


Concept of Cookie and Session (Cont)

- One solution is using **Cookie,** which is invented to keep some data used to identify clients and keep some information specific to each client.

Concept of Cookie and Session (Cont)

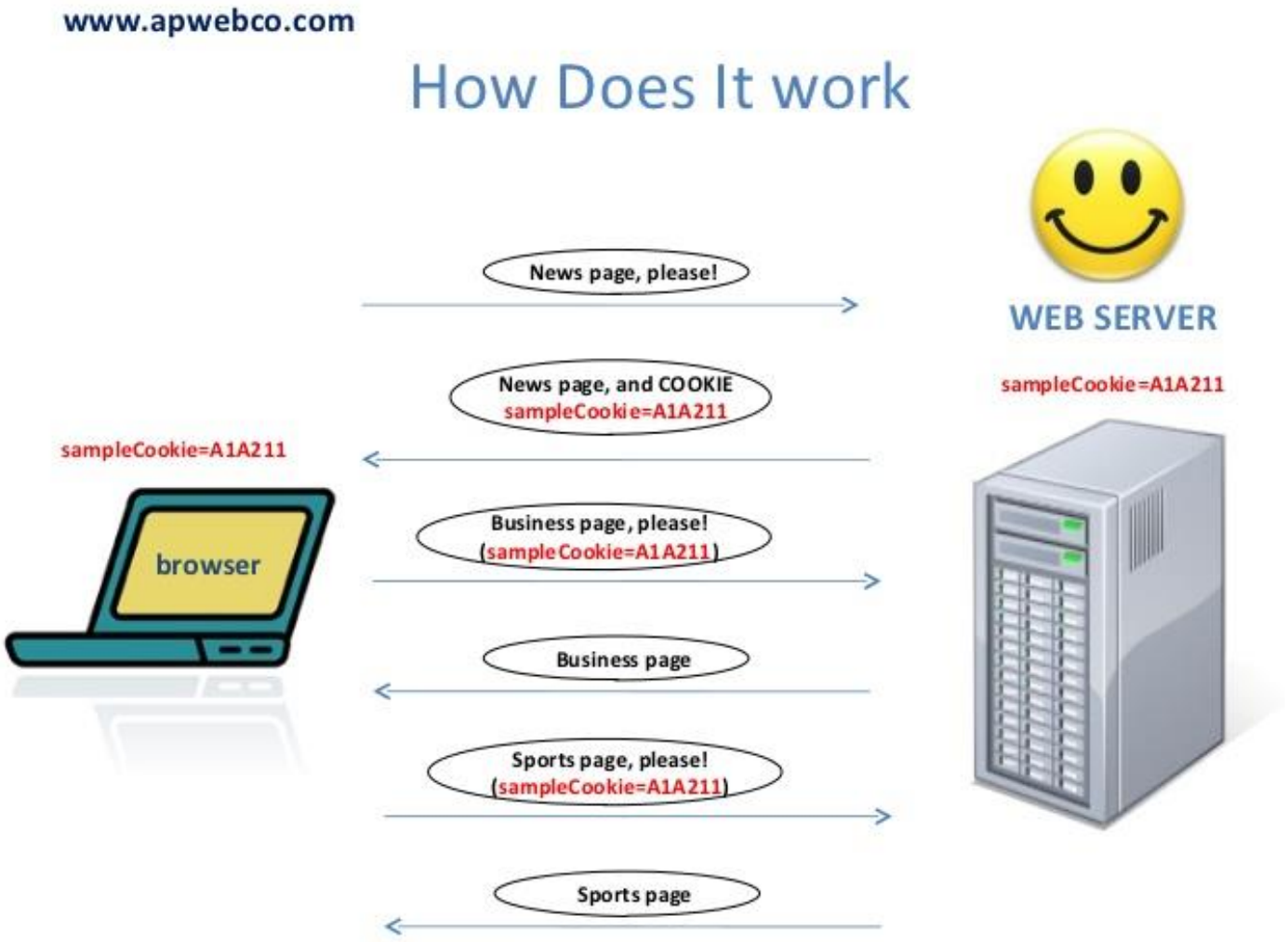
- Both concept can identify clients, however there are different in concept.



Cookie

- Cookies are an extension of the HTTP protocol. More precisely, they consist of two HTTP headers: **the Set-Cookie response** header and **the Cookie request** header.
- In other words, to **create** and **retrieve** the cookies.
- If you use this concept to allow a unique identifier to be included in each request (in a **Cookie header**), you can begin to uniquely identify clients and associate their requests together.

Cookie (Cont)



setcookie

- To identify clients with cookie, we must create it first.
- **setcookie()** defines a cookie to be sent along with the rest of the HTTP headers.
- Cookies must be sent ***before*** any output from your script (this is a protocol restriction). This requires that you place calls to this function prior to any output, including *<html>* and *<head>* tags as well as any whitespace.
- Once the cookies have been set, they can be accessed on the next page load with the **$_COOKIE** arrays.

setcookie

- Syntax:

`setcookie(name, value, expire, path, domain);`

–Name - Name of the cookie.

–Value - Value to be stored by the cookie in the client.

–Expire - the time a cookie will be deleted with the format time**() + N seconds of duration**

–Path - Subdirectory where the cookie has a value.

–Domain - Domain where the cookie has a value.

- **Example**

60 seconds x 60 minutes x 24 hours x 60 days

```
1.<?php
2.   setcookie("user", "Alex Porter", time()+3600);
3.   $inTwoMonths = 60 * 60 * 24 * 60 + time();
4.   setcookie('lastVisit',date("G:i-m/d/y"),$inTwoMonths);
5.?>
6.<html>
.....
```

G: 24-hour format (0-23)
i: Minutes (00-59)
m: Month d: Day y: Year

How to access cookie

- The PHP global $_COOKIE array is used to retrieve a cookie value.
- We refer to each value using cookie's name as an index.
- Example

```
1. <html>
2. <body>
3. <?php
4.   if (isset($_COOKIE["user"]))
5.     echo "Welcome ".$_COOKIE["user"]."!<br />";
6.   else
7.     echo "Welcome guest!<br />";
8. ?>
9. </body>
10.</html>
```

Welcome Alex Porter!

Welcome guest!

How to delete cookie

- Cookie will be deleted if and only if its expire time is reach.
- If you need to delete it suddenly, you must change its expire time.
- Example

```
1.<?php
2.  // set the expiration date to one hour ago
3.  setcookie("user", "", time()-3600);
4.?>
```

Environmental Variables

- Beside cookies, there are environmental variables which allow us to refer to data related to network and system.
- Example:

```
1.<?php
2.  echo "You are using $_SERVER[HTTP_USER_AGENT]<br />";
3.  echo "Your Internet address is $_SERVER[REMOTE_ADDR]<br />";
4.?>
```

- Output:

```
You are using Mozilla/5.0 (Windows; U; Windows NT 6.1; th; rv:1.9.2.6) Gecko/20100625 Firefox/3.6.6
Your Internet address is 202.28.180.202
```

Note: ***Specific environment variables, $_SERVER[HTTP_USER_AGENT], and $_SERVER[REMOTE_ADDR] contain the name of the browser being used and the IP address of the user.***

Superglobal Variables

Superglobal variables are predefined arrays and are accessible from anywhere on the page.

Variable	Description
$_SERVER	Contains information about the server and the HTTP connection. Analogous to the old $HTTP_SERVER_VARS array (which is still available, but deprecated).
$_COOKIE	Contains any cookie data sent back to the server from the client. Indexed by cookie name. Analogous to the old $HTTP_COOKIE_VARS array (which is still available, but deprecated).
$_GET	Contains any information sent to the server as a search string as part of the URL. Analogous to the old $HTTP_GET_VARS array (which is still available, but deprecated).
$_POST	Contains any information sent to the server as a POST style posting from a client form. Analogous to the old $HTTP_POST_VARS array (which is still available, but deprecated).
$_FILE	Contains information about any uploaded files. Analogous to the old $HTTP_POST_FILES array (which is still available, but deprecated).

Some Apache Environmental Variables

Variable	Description	Example of result
QUERY_STRING	The information (if any) following the "?" in the URL for this request, for example, myform.html?a=b&c=d would provide	QUERY_STRING=a=b&c=d
REMOTE_ADDR	The IP address of the host making this request	REMOTE_ADDR=207.35.76.27
REMOTE_PORT	The port number used by the remote host when making this request	REMOTE_PORT=4325
REQUEST_METHOD	The method used for this request for HTTP "GET", HEAD" or "POST"	REQUEST_METHOD=GET
REQUEST_URI	The URI for this request (relative to DOCUMENT_ROOT)	REQUEST_URI=/tech/web/ssi.htm
SERVER_ADDR	The IP address of the server for this URL	SERVER_ADDR=207.35.76.24
SERVER_ADMIN	The administrators e-mail address for this SERVER_NAME	SERVER_ADMIN=webmaster@zytrax.com

Some Apache Environmental Variables

Variable	Description	Example of result
SERVER_NAME	The servers host name, DNS alias or IP address.	SERVER_NAME=www.zytrax.com
SERVER_PORT	The port number on this server to which this request was directed	SERVER_PORT=80
SERVER_SOFTWARE	The name and version of the information server answering the query	SERVER_SOFTWARE=Apache/1.3.14 (Unix) (Red-Hat/Linux) PHP/4.0.3pl1
HTTP_ACCEPT_LANGUAGE	The LANGUAGE types the server is requested to accept as defined in the HTTP header and typically used for content negotiation	HTTP_ACCEPT_LANGUAGE=en-us
HTTP_CONNECTION	The type of connection as defined in the HTTP header	HTTP_CONNECTION=Keep-Alive
HTTP_HOST	The base URL of the host	HTTP_HOST=www.zytrax.com
HTTP_USER_AGENT	The browser id or user-agent string identifying the browser	HTTP_USER_AGENT=Mozilla/4.0 (compatible; MSIE 5.0; Windows NT; DigExt)

File

- fopen() function is used to open files
- exit() function generates a message if the fopen() function is unable to open the specified file
- fclose() function is used to close files
- Example

```
1.<html>
2.<body>
3.
4.<?php
5.  $file =  fopen("welcome.txt","r")   or exit("Unable to open file!");
6.
7.  //some code to be executed
8.
9.  fclose($file)
10.?>
11.
12.</body>
13.</html>
```

File: Modes

Modes	Description
r	Read only. Starts at the beginning of the file
r+	Read/Write. Starts at the beginning of the file
w	Write only. Opens and clears the contents of file; or creates a new file if it doesn't exist
w+	Read/Write. Opens and clears the contents of file; or creates a new file if it doesn't exist
a	Append. Opens and writes to the end of the file or creates a new file if it doesn't exist
a+	Read/Append. Preserves file content by writing to the end of the file
x	Write only. Creates a new file. Returns FALSE and an error if file already exists
x+	Read/Write. Creates a new file. Returns FALSE and an error if file already exists

File: Reading a file

- Line by line

```
1.<?php
2.  $file = fopen("welcome.txt", "r") or exit("Unable to open file!");
3.  //Output a line of the file until the end is reached
4.  while(!feof($file))
5.    {
6.      echo fgets($file). "<br />";
7.    }
8.  fclose($file);
9.?>
```

End-of-file

- Character by character

```
1.<?php
2.  $file = fopen("welcome.txt", "r") or exit("Unable to open file!");
3.  //Output a line of the file until the end is reached
4.  while(!feof($file))
5.    {
6.      echo fgetc($file). "<br />";
7.    }
8.  fclose($file);
9.?>
```

File: Reading a file (Cont)

```
AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language
```

Webdictionary.txt

```
<!DOCTYPE html>
<html>
<body>
<?php
$myfile =
fopen("webdictionary.txt", "r") or die
("Unable to open file!");
// Output one line until end-of-file
while(!feof($myfile)) {
  echo fgets($myfile) . "<br>";
}
fclose($myfile);
?>
</body>
</html>
```

Output

AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language

```
<!DOCTYPE html>
<html>
<body>
<?php
$myfile =
fopen("webdictionary.txt", "r") ordie("Unable
to open file!");
// Output one character until end-of-file
while(!feof($myfile)) {
  echo fgetc($myfile);
}
fclose($myfile);
?>
</body>
</html>
```

Output

AJAX = Asynchronous JavaScript and XML CSS = Cascading Style Sheets HTML = Hyper Text Markup Language PHP = PHP Hypertext Preprocessor SQL = Structured Query Language SVG = Scalable Vector Graphics XML = EXtensible Markup Language

File Upload

- A very useful aspect of PHP is its ability to manage file uploads to your server. Allowing users to upload a file to your server including malicious files such as worms. So we must consider carefully when enabling file uploads.
- We upload through HTML form in a sender page. Then we use PHP code to manage at a receiver page.

File Upload: Sender page

sender.html

```
1.<form action = "receiver.php" method="post " enctype="multipart/form-data">
2.    <input type="hidden" name="MAX_FILE_SIZE" value="100000" />
3.    <input type="file" name="file" id="file" /><br />
4.    <input type="submit" name="submit" value="Submit" />
5.</form>
```

- **enctype="multipart/form-data"** - Necessary for a to-be-created PHP file to function properly.
- **input type="hidden" name="MA...** - Sets the maximum allowable file size, in bytes, that can be uploaded. This safety mechanism is easily bypassed and we will show a solid backup solution in PHP. We have set the max file size to 100KB in this example.
- **input name="file"** – We will refer to **file** later in PHP script.

File Upload: Receiver page

- After submitting, files will, by default, be stored in the server's default temporary directory. If the file is not moved to a different location it will be destroyed.
- **Steps should be followed in order to completely save the file**

1. **Checking step**: We use a PHP code to refer to that file with the global **$_FILES** array where PHP stores all the information about files.

At the receiver page, the PHP file should make a key decision to keep the file or throw it away. You may need to thrown it away in some cases such as too large file or unacceptable file type.

```
$_FILES[ referringName ][ fileInfoNeeded ]
```

2. **Saving step**: Move the file to a different location with the code

```
move_uploaded_file($_FILES[ referringName ]["tmp_name"],destination)
```

***Note referringName is a value in name attribute in the sender page

File Upload: global $_FILES array

There are information about the file which can be used in Checking step

Note: "**`file`**" in the code below is a value in name attribute in the sender page, which can be replaced by any name.

$_FILES array	Description
$_FILES['**file**']['name']	The original name of the file on the client machine.
$_FILES['**file**']['size']	The size, in bytes, of the uploaded file.
$_FILES['**file**']['tmp_name']	The temporary filename of the file in which the uploaded file was stored on the server.
$_FILES['**file**']['error']	The error code associated with this file upload.
$_FILES['**file**']['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.

File Upload: MIME Type

- Multipurpose Internet Mail Extensions
  - Attachment file type for email clients → Image file, spreadsheet
- Now, it is called **“Internet media type”**
- It’s an identifier for <u>file formats</u> on the Internet
- Example
  - application/zip
  - audio/mp3, audio/ogg
  - image/gif, image/jpeg
  - text/html, text/css




File Upload: Receiver page example

```
<?php
    if ((($_FILES["file"]["type"] == "image/gif")
      || ($_FILES["file"]["type"] == "image/jpeg")
      || ($_FILES["file"]["type"] == "image/pjpeg"))
      && ($_FILES["file"]["size"] < 20000)) {
        if($_FILES["file"]["error"] > 0){
            echo "Return Code: ".$_FILES["file"]["error"]."<br />";
        }
        else{
            echo "Upload: ".$_FILES["file"]["name"]."<br />";
            echo "Type: ".$_FILES["file"]["type"]."<br />";
            echo "Size: ".($_FILES["file"]["size"]/1024)." Kb<br />";
            echo "Temp file: ".$_FILES["file"]["tmp_name"]."<br />";
            if (file_exists("upload/".$_FILES["file"]["name"])){
                echo $_FILES["file"]["name"]." already exists. ";
            }
            else{
                move_uploaded_file($_FILES["file"]["tmp_name"],"upload/"
                . $_FILES["file"]["name"]);
                echo "Stored in: "."upload/".$_FILES["file"]["name"];
            }
        }
    }
    else{ echo "Invalid file";}
?>
```

Checking step

Display to user

Saving step

Some useful PHP Filesystem Functions

Function	Description
basename(path,suffix)	Returns the filename component of a path
file_exists(filePath)	Checks whether or not a file or directory exists
fileatime(filename)	Returns the last access time of a file
filectime(filename)	Returns the last change time of a file
filemtime(filename)	Returns the last modification time of a file
filesize(filename)	Returns the file size
filetype(filename)	Returns the file type

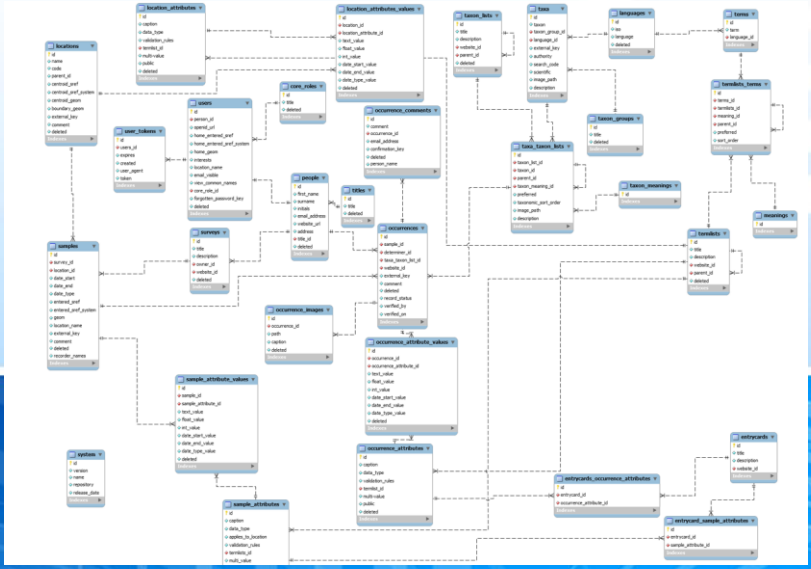
Example

```
<?php
        if(file_exists("test.txt"))
                echo filesize("test.txt");
?>
```

Database

A **database** is an organized collection of data, which has many types including Relational database, Big data, Graph database, XML database, etc.

A relational database is a collection of schemas, tables, queries, reports, views, and other elements.



SQL

- SQL is a standard language for storing, manipulating and retrieving data in databases.

- SQL is used to manage database in database management system (DBMS) such as MySQL, SQL Server, MS Access, Oracle, Sybase, Informix, Postgres, and other database systems.

Basic SQL Command

Commands	Description
INSERT	Insert row(s) into a table
UPDATE	Modify an attribute's value in one or more table's rows
DELETE	Delete one or more rows from a table
SELECT	Select attributes from rows in one or more tables
WHERE	Restrict the selection of rows based on condition expression
GROUP BY	Group the selected rows based on one or more attributes
HAVING	Restrict the selection of group rows based on a condition
ORDER BY	Order the selected rows based on one or more attributes

DB Connection with PHP

- Database(DB) is used to store data at server.
- PHP(server-side script) can be used to connect to database in order to manage data using SQL, e.g., insert new data, update data, delete data.

Implement the connection

- We will assume that we have this table in database:

Table Structure

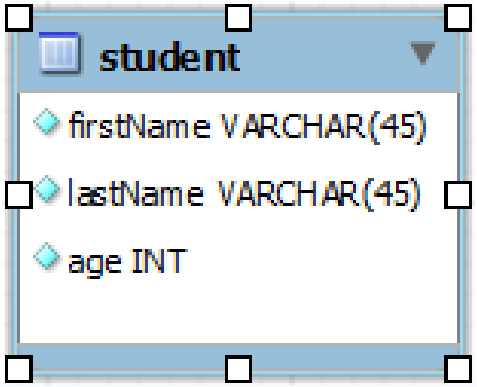


Table data

Firstname	Lastname	Age

Implement SQL: Insert Example

```
1.<?php
2.    $conn = new mysqli("localhost"," Chris ","abc007","student");
3.    // $conn = new mysqli($servername, $username, $password, $dbname);
4.         if (!$conn){
5.       Connect to database   die("Could not connect: ".$conn->connect_error);
6.         }                    ชื่อ Table                Attributes
7.    $sql = "INSERT INTO Student (FirstName, LastName, Age)
                 VALUES ('Peter', 'Parker', '23')"; ใส่ value อะไรเข้าไป
8.    if ($conn->query($sql) === TRUE) {
9.              echo "New record created successfully";}
10.      else {
11.             echo "Error: " . $sql . "<br>" . $conn->error; }
12.     $conn->close();
13.?>
```

Can you guess what will happen in the database?

Student Table

Firstname	Lastname	Age
Peter	Parker	23

Implement SQL: Update Example

```
1.<?php
2.    $conn = new mysqli("localhost"," Chris ","abc007","student");
3.    // $conn = new mysqli($servername, $username, $password, $dbname);
4.         if (!$conn){
5.                   die("Could not connect: ".$conn->connect_error);
6.         }
7.    $sql = "UPDATE Student SET Age = '36'
                 WHERE firstName = 'Peter'
                 AND lastName = 'Parker'";
8.    if ($conn->query($sql) === TRUE) {
9.              echo "New record created successfully";}
10.      else {
11.             echo "Error: " . $sql . "<br>" . $conn->error; }
12.    $conn->close();
13.?>
```

ต้องบอกว่าจะ update คนไหน
ไม่งั้นมันจะ update ทุก row

Can you guess what will happen in the database?

Student Table

Firstname	Lastname	Age
Peter	Parker	23 ➔ 36

Implement SQL: Delete Example

```
1.<?php   Start connection
2.    $conn = new mysqli("localhost"," Chris ","abc007","student");
3.    // $conn = new mysqli($servername, $username, $password, $dbname);
4.         if (!$conn){
5.                   die("Could not connect: ".$conn->connect_error);
6.         }
7.    $sql = "DELETE FROM Student
                  WHERE lastName = 'Parker'";
8.    if ($conn->query($sql) === TRUE) {
9.              echo "New record created successfully";}
10.       else {
11.             echo "Error: " . $sql . "<br>" . $conn->error; }
12.     $conn->close();
13.?>
```

Could you guess what will happen in the database?

Student Table

Firstname	Lastname	Age

Implement SQL: Select Example

```
1.<?php
2.    $conn = new mysqli("localhost"," Chris ","abc007","student");
3.    // $conn = new mysqli($servername, $username, $password, $dbname);
4.       if (!$conn){
5.                 die("Could not connect: ".$conn->connect_error);
6.       }
7.      $sql = "SELECT * FROM Students";
8.      $result = $conn->query($sql)
9.
10.    while($row = $result->fetch_assoc()){
11.      echo $row['FirstName'] . " " . $row['LastName'];
12.      echo "<br />";
13.    }
14.    $conn->close();
15.?>
```

Every columns + Every rows

Fetch information from $result row by row

Could you guess what will display on screen?

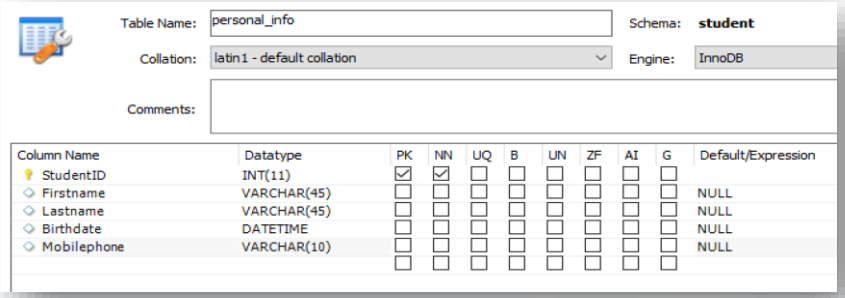
Result

Peter Parker

Activity 1

DB connection with PHP

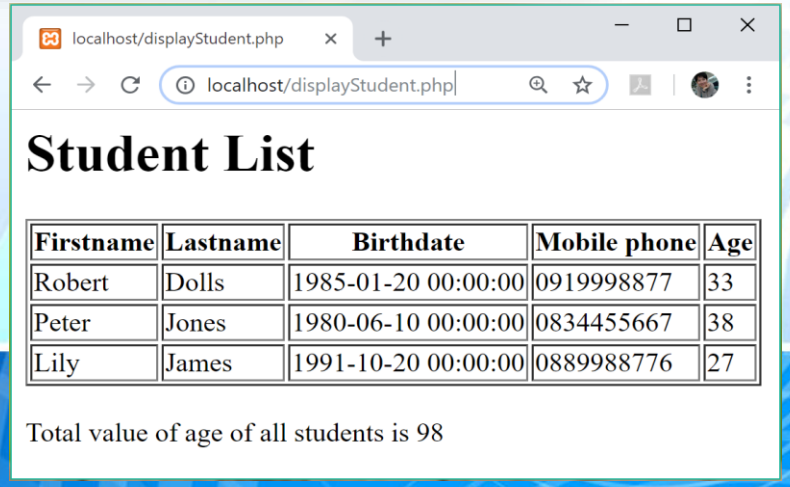
- Install MySQL
- Create schema and table



- Download and run display.php
- Modify display.php

StudentID	Firstname	Lastname	Birthdate	Mobilephone
1	Robert	Dolls	20 Jan 1985	0919998877
2	Peter	Jones	10 June 1980	0834455667
3	Lily	James	20 Oct 1991	0889988776

display.php



References

- W3Schools website: http://www.w3schools.com
- PHP Manual: http://www.php.net/manual/en/
- PHP Tutorial: http://www.tizag.com/phpT/index.php