## PHP

With pure **HTML web pages**, the server simply serves **static HTML** that only display content. PHP is used to turn the web sites into interactive web application.

  



Fig 1: Web Server 🡪 Static HTML File 🡪 Client Web Browser

With **PHP** in the mix, the web server is able to **dynamically generate HTML** web pages

   





Fig 2: Web Server 🡪 PHP 🡪 MySQL 🡪 PHP 🡪 Client Web Server

**PHP**: **Personal Home Pages**, it was later changed **PHP Hypertext Processor**.

**PHP**: **Server-side programming language** – it runs on a web server.

### Simple Form(Form.html)

<!doctype html>

<**html**>

<**head**>

<**meta** charset = "UTF-8"/>

<**title**> Simple Form </**title**>

</**head**>

<**body**>

<**p**> Please fill out the form </**p**>

<**img** src = "database.png" width = "100" height = "175"

alt = "database picture"/><**br**/>

<**form** method = "post" action = "khan.photon@gmail.com">

<**label** for = "firstname"> First Name: </**label**>

<**input** type = "text" id = "filename" name = "firstname"/><**br**/>

<**label** for = "lastname"> Last Name: </**label**>

<**input** type = "text" id = "lastname" name = "lastname"/><**br**/>

<**label** for = "email"> Email: </**label**>

<**input** type = "email" id = "email" name = "email"/><**br**/>

<**label** for = "gender">Gender: </**label**>

Male <**input** type = "radio" id ="gender" name = "gender" value = "male">

Female <**input** type = "radio" id = "gender" name = "gender"

value = female"><**br**/>

<**label** for = "description"> Description: </**label**>

<**textarea** name = "description"> Max word 500 words </**textarea**><**br**/>

<**input** type = "submit" value = "Submit" name = "submit"/>

</**form**>

</**body**>

</**html**>

Code 1 (Form.html): Simple Form HTML Code

**Tags:** <**form**></**form**>, <**input></input>,** <**label></label>**

**Attributes**: action, type, id, method, name, class, value

### Breakdown(Form.html)

Basic HTML Format

<!doctype html>

<**html**>

<**head**>

<**meta** charset = "UTF-8"/>

<**title**> Simple Form </**title**>

</**head**>

<**body**>

</**body**>

</**html**>

|  |  |
| --- | --- |
| **Labels/Tags** | **Description** |
| <!doctype html> | A document type declaration |
| <**html**> | Tells the browser that this is an HTML document |
| <**head**></**head**> | Includes title for the document, scripts, styles, meta information |
| <**meta** charset="UTF-8"/> | Specifies what character set is our website written with |
| "UTF-8" | Universal Character Set + Transformation Format 8-bit |
| <**body**></**body**> | This is where all the HTML contents are written |
| <**p**></**p**> | Paragraph |
| <**img** src="database.png"/> | Include Image, Attribute src is the image src path |
| alt="database picture" | Alternatively, text will show up if no image found |
| width="100" height="175" | Fixing the width and height of the image |

<**form** method = "post" action = "khan.photon@gmail.com">

Form action attribute should be changed

<**form** method = "post" action = "report.php"></**form**>

Form tag causes the PHP script to run on the server.

|  |  |
| --- | --- |
| **Labels/Tags** | **Description** |
| <**input><**/**input>** | Input tag is the input field where user can enter data |
| type="text | Attribute type is the value received and saved from the user |
| id="filename" | Attribute id is the unique identifier; used for manipulating the element |
| name="firstname" | Attribute name is the value received and saved from the user |

<**label** for = "gender">Gender: </**label**>

Male <**input** type = "radio" id ="gender" name = "gender" value = "male">

Female <**input** type = "radio" id = "gender" name = "gender" value = "female">

Label tag allows user clicks on the text within the <**label>** element, it toggles the control.

For attribute should be equal to the id attribute of the related element to bind together

<**textarea** name = "description"> Max word 500 words </**textarea**><**br**/>

Textarea tag allows users to type multi-lines, description type situations

<**input** type = "submit" value = "Submit" name = "submit"/>

Type attribute allows user to make a submit button

Value attribute allows user to change the text in the submit button

<**br**/>

It allows the display go to next line. Single line break.

### Confirmation(Report.php)

<?php

**if**(!empty($\_POST["submit"])):

$firstname = $\_POST["firstname"];

$lastname = $\_POST["lastname"];

$email = $\_POST["email"];

$gender = $\_POST["gender"];

$description = $\_POST["description"];

**echo** "First Name: ".$firstname."<br/>";

**echo** "Last Name: ".$lastname."<br/>";

**echo** "Email: ".$email."<br/>";

**echo** "Gender: ".$gender."<br/>";

**echo** "Description: ".$description."<br/>";

**endif**;

?>

Code 2 (Report.php): Receiving the user input information from form.html

### Breakdown(Report.php)

|  |  |
| --- | --- |
| **Labels/Tags** | **Description** |
| <?php ?> | It indicates that the PHP code is written in this block |
| **If**(condition): | If the condition is true, do the following statement |
| $\_POST["submit"] | $\_POST receives the value from the name attribute |
| $firstname | $ creates the variable |
| **echo** | Outputting information beyond the confines of the <?php ?> |

Web browser know nothing about PHP and, therefore, have no ability to run PHP scripts.

Web servers with PHP support are equipped to run PHP scripts and turn them into HTML web pages that browsers can understand.

Every PHP must end with a semicolon “;”.

Name the PHP file with “.php”

### PHP Information(info.php)

<?php

phpinfo();

?>

Code 3 (Info.php): Detailed information about the PHP installed in the PC

### Variable

* PHP variable names must begin with a dollar sign, and cannot contain spaces
* The first character after the dollar be a letter or an underscore “\_”
* Characters after the first character after that can be a **letter, an underscore, or a number**
* Spaces and special characters are not allowed in any part of a variable name
* Use all lowercase for variable name
* Separate words in a multi-word variable name with underscores

### Assigning value to the variable

$description = “I am the man.”;

Pieces of text(strings) must be enclosed by quotes, either single quotes or double quotes.

### Special Variable

$\_POST is a special variable that is known as a superglobal. It is a collection of storage locations used to hold data from a web form. It is also an **array**. This array is filled the values the user entered into the form.

### Concatenation

**echo** "Description: ".$description."<br/>";

The period allows us to stick multiple strings of text together as one. This process is known as concatenation.

<?php

$name = "Jack Stubbort";

$age = "17";

$place = "Netherland";

$msg = $name."is ".

$age."-years-old. "

."He works in ".$place.".";

**echo** $msg;

?>

Code 4 (Concatenation.php): Long line of PHP codes spanned across multiple lines

### Escape Characters

* Escape characters in PHP starts with a backlash “\”
* Escape characters can be escaped in double-quoted only
* Single-quoted string only allow ‘\’ but not ‘\\’

<?php

$name = "Allen Smith";

$age = "21";

$occupation = "Engineer";

**echo** '$name is $age-years-old. \r\n He is an \"$occupation.\"';

**echo** "<br/><br/>";

**echo** nl2br("**$name** is **$age**-years-old. **\n** He is an **\"$occupation**.**\"\n\n**");

**echo** "**\\** **\"**";

?>

Code 5 (DoubleSingle.php): Escape Characters.

For newline break, we need to use nl2br() function to go new line.

### Mailing

#### Initial Setup

We can send mail from localhost with sendmail package, sendmail package is inbuild in XAMPP. So, if we are using XAMPP then you can easily send mail from localhost.

For example: We can configure C:\xampp\php\php.ini and c:\xampp\sendmail\sendmail.ini for gmail to send mail.

In C:\xampp\php\php.ini find extension=php\_openssl.dll and remove the semicolon from the beginning of that line to make SSL working for gmail for localhost.

In php.ini file find [mail function] and change

SMTP=smtp.gmail.com

smtp\_port=587

sendmail\_from = my-gmail-id@gmail.com

sendmail\_path = "\"C:\xampp\sendmail\sendmail.exe\" -t"

Now Open C:\xampp\sendmail\sendmail.ini. Replace all the existing code in sendmail.ini with following code

[sendmail]

smtp\_server=smtp.gmail.com

smtp\_port=587

error\_logfile=error.log

debug\_logfile=debug.log

auth\_username=my-gmail-id@gmail.com

auth\_password=my-gmail-password

force\_sender=my-gmail-id@gmail.com

Now you have done!! create php file with mail function and send mail from localhost.

PS: Don't forgot to replace my-gmail-id and my-gmail-password in above code. Also, don't forget to remove duplicate keys if you copied settings from above. For example comment following line if there is another sendmail\_path : sendmail\_path="C:\xampp\mailtodisk\mailtodisk.exe" in the php.ini file

Also remember to restart the server using the XAMMP control panel so the changes take effect.

For gmail please check [**https://support.google.com/accounts/answer/6010255**](https://support.google.com/accounts/answer/6010255%20) to allow access from less secure apps.

**Source:** [**https://stackoverflow.com/questions/15965376/how-to-configure-xampp-to-send-mail-from-localhost**](https://stackoverflow.com/questions/15965376/how-to-configure-xampp-to-send-mail-from-localhost)

### Mailing(Mail.php)

<?php

**echo** '<!doctype html>';

**echo** ' <html>';

**echo** ' <head>';

**echo** ' <meta charset = "UTF-8"/>';

**echo** ' <title> Simple Form </title>';

**echo** ' </head>';

**echo** ' <body>';

**echo** ' <img src = "images/database.png" width = "100" height = "100"';

**echo** ' alt = "database picture"/><br/>';

**echo** ' <p> Please fill out the form </p>';

**echo** ' <form method = "post" action = "';

**echo** $\_SERVER['PHP\_SELF']."**\"**>";

**echo** ' <label for = "firstname"> First Name: </label>';

**echo** ' <input type = "text" id = "filename" name = "firstname"/><br/>';

**echo** ' <label for = "lastname"> Last Name: </label>';

**echo** ' <input type = "text" id = "lastname" name = "lastname"/><br/>';

**echo** ' <label for = "email"> Email: </label>';

**echo** ' <input type = "email" id = "email" name = "email"/><br/>';

**echo** ' <label for = "gender">Gender: </label>';

**echo** ' Male <input type = "radio" id ="gender" name = "gender" value = "male">';

**echo** ' Female <input type = "radio" id = "gender" name = "gender" value = "female"><br/>';

**echo** ' <label for = "description"> Description: </label>';

**echo** ' <textarea name = "description"> Max word 500 words </textarea><br/>';

**echo** ' <input type = "submit" value = "Submit" name = "submit"/>';

**echo** ' </form>';

**echo** ' </body>';

**echo** ' </html>';

?>

Code 6 (Mail.php): First Section.

Form.html file has been converted to PHP file by adding echo.

$\_SERVER['PHP\_SELF']

The form sends the data to the same page and not directing to another PHP file.

<?php

**if**(!**empty**($\_POST["submit"])):

$firstname = $\_POST["firstname"];

$lastname = $\_POST["lastname"];

$email = $\_POST["email"];

$gender = $\_POST["gender"];

$description = $\_POST["description"];

**echo** "First Name: ".$firstname."<br/>";

**echo** "Last Name: ".$lastname."<br/>";

**echo** "Email: ".$email."<br/>";

**echo** "Gender: ".$gender."<br/>";

**echo** "Description: ".$description."<br/>";

$to = "khan.photon@gmail.com";

$subject = "Extra Description";

$message = $description;

mail($to, $subject, $message, 'From:'.$email);

*//Allow less secure apps on*

**endif**;

?>

Code 7 (Mail.php): Second Section.

It is exactly like the Report.php file only the mail($to, $subject, $message, 'From:'.$email) function has been added.

|  |  |
| --- | --- |
| **Variable** | **Description** |
| $to | Whom you are sending email to |
| $subject | Subject part of the email |
| $message | The actual message of the email |
| 'From:'.$email | Email Address of the sender |
| "From:".$email."\r\nCc:".$cc | Separate From and CC using (\r\n), Use double quotation |

### Connecting to the Database

SQL: Structured Query Language, it is the language used to communicate.

MySQL: Stores data inside of database tables. MySQL databases are organized into tables, which store information as rows and columns of related data.



  

 

Fig 3: Client Web Server 🡪 Request 🡪 Web server Response

### Communicating with a Databaser Server

PhpMyAdmin: It is a graphical tool that allows us to create databases and tables through a web interface



Fig 4: PhpMyAdmin

First of all, install XAMPP for Windows or install LAMPP for Linux.

#### Installing XAMPP for Windows

Go to this link, [**https://www.apachefriends.org/download.html**](https://www.apachefriends.org/download.html), Download the latest version of XAMPP through the Mozilla browser.

After installation, open the XAMPP software and turn on



Fig 5: XAMPP Control Panel

Create a folder in **C:\xampp\htdocs\,** you can name folder anything, for the time being let us just create a folder named “practice”. Now we will have **C:\xampp\htdocs\practice,** we can create PHP or HTML files inside it and then deploy it on the browser. Let us create a file, **form.php**. Therefore, we will end up having **C:\xampp\htdocs\practice\form.php.**

Now go to your favorite browser, type [**http://localhost/practice/form.php**](http://localhost/practice/form.php)**,** code in PHP would display here. Therefore **C:\xampp\htdocs\** folder location gets replaced by [**http://localhost/**](http://localhost/) in your browser.

If we type, [**http://localhost/phpmyadmin/**](http://localhost/phpmyadmin/)**,** this will show our MySQL database.

#### Light Weight Editor (Sublime Text Editor)

From this link download, [**https://www.sublimetext.com/3**](https://www.sublimetext.com/3)**,** this is probably one of the best coding editor.

If you want to install packages in sublime text first you need to install **package control.** Press the **Ctrl+`** in sublime editor and copy paste the code given in [**https://packagecontrol.io/installation**](https://packagecontrol.io/installation)**.**



Fig 6: Sublime Text Editor (Package Control Install)

Once installed, press **Ctrl+Shift+P,** type in **Package Control: Install Package**

#### Installing XAMPP for Linux

Go to this link, <https://www.apachefriends.org/download.html>, Download the latest version of XAMPP.

Go to **/Home/Your-Username/Downloads.** Ex: **/Home/Photon/Downloads**

Open the command terminal there and type:

**sudo chmod 777 name-of-the-downloaded-file.**

Ex: **sudo chmod 777 ./xampp-linux-x64-7.2.2-0-installer.run**

This allows the administration permission to install the file

**sudo ./name-of-the-file**

Ex: **./xampp-linux-x64-7.2.2-0-installer.run**

This is how the installation of the file works

**sudo /opt/lamp/lampp start**

This starts the LAMPP on

#### Installing Sublime Editor for Linux

sudo apt-get-repository ppa:webupd8team/sublime-text-3

sudo apt-get update

sudo apt-get install sublime-text-installer

This is how the installation of the file works

### Creating a Database

Creating a database (form\_table). Tables are stored in database



Fig 7: Creating a Database (form\_table)

Click on New 🡪 Type in database name (form\_table) 🡪 click on create

#### Creating a Table

Tables serves as a way to divide up the data in a database into related groups



Fig 8: Creating a Table (Inside the form\_table Database)

#### Breakdown (SQL: form)

CREATE TABLE form (

id INT AUTO\_INCREMENT,

first\_name VARCHAR(30),

last\_name VARCHAR(30),

email VARCHAR(50),

gender VARCHAR(10),

description VARCHAR(100),

PRIMARY KEY(id)

)

Code 8: Create a form table

CREATE TABLE form

Name of the table is form

id INT AUTO\_INCREMENT

Number the rows using id, (integer and it will increment by itself)

first\_name VARCHAR(30)

First name will be stored here, its variable character length is 30

email VARCHAR(50)

Variable character length is 50

PRIMARY KEY(id)

This allows id to uniquely identify each record in the table

#### Using INSERT command

INSERT INTO form(first\_name, last\_name, email, gender, description)

VALUES('Photon', 'Khan', 'khan.photon@gmail.com', 'male', 'I am too fat')

Code 9: Insert data into the form table

INSERT INTO table\_name (column\_name, …. …. ….. ….)

VALUES (‘value 1’, ‘value 2’)

See the values are in single quotation. Values in the set of parentheses have to be in the same order as the database column names

#### Using SELECT command

SELECT columns FROM table\_name

Ex: SELECT first\_name, last\_name FROM form

SELECT \* FROM table\_name

Ex: SELECT \* FROM form

\*: Fetch all the data from all the columns in the table

Code 10: Select data from the table

### Automating SQL using PHP

Using PHP all the data entered through the form will be saved into the database.

<?php

**echo** '<!doctype html>';

**echo** ' <html>';

**echo** ' <head>';

**echo** ' <meta charset = "UTF-8"/>';

**echo** ' <title> Simple Form </title>';

**echo** ' </head>';

**echo** ' <body>';

**echo** ' <p> Please fill out the form </p>';

**echo** ' <form method = "post" action = "';

**echo** $\_SERVER['PHP\_SELF']."**\"**>";

**echo** ' <label for = "firstname"> First Name: </label>';

**echo** ' <input type = "text" id = "filename" name = "firstname"/><br/>';

**echo** ' <label for = "lastname"> Last Name: </label>';

**echo** ' <input type = "text" id = "lastname" name = "lastname"/><br/>';

**echo** ' <label for = "email"> Email: </label>';

**echo** ' <input type = "email" id = "email" name = "email"/><br/>';

**echo** ' <label for = "gender">Gender: </label>';

**echo** ' Male <input type = "radio" id ="gender" name = "gender" value = "male">';

**echo** ' Female <input type = "radio" id = "gender" name = "gender" value = "female"><br/>';

**echo** ' <label for = "description"> Description: </label>';

**echo** ' <textarea name = "description"> Max word 500 words </textarea><br/>';

**echo** ' <input type = "submit" value = "Submit" name = "submit"/>';

**echo** ' </form>';

**echo** ' </body>';

**echo** ' </html>';

?>

Code 11: Form.php (First Part)

<?php

**if**(!**empty**($\_POST["submit"])):

$firstname = $\_POST["firstname"];

$lastname = $\_POST["lastname"];

$email = $\_POST["email"];

$gender = $\_POST["gender"];

$description = $\_POST["description"];

**echo** "First Name: ".$firstname."<br/>";

**echo** "Last Name: ".$lastname."<br/>";

**echo** "Email: ".$email."<br/>";

**echo** "Gender: ".$gender."<br/>";

**echo** "Description: ".$description."<br/>";

$server\_name = "localhost";

$user\_name = "root";

$password = "";

$database\_name = "form\_table";

$connection = mysqli\_connect($server\_name, $user\_name, $password, $database\_name)

**or** **die**("Could not connect to the server");

$query = "INSERT INTO form(first\_name, last\_name, email, gender, description)

VALUES('**$firstname**','**$lastname**', '**$email**', '**$gender**', '**$description**')";

$result = mysqli\_query($connection, $query) **or** **die**("Error Querying database");

mysqli\_close($connection);

**endif**;

?>

Code 12: Form.php (Second Part)

#### Breakdown (Form.php (Second Part))

$connection = mysqli\_connect($server\_name, $user\_name, $password, $db\_name)

**or** **die**("Could not connect to the server");

$query = "INSERT INTO form(first\_name, last\_name, email, gender, description)

VALUES('**$firstname**','**$lastname**', '**$email**', '**$gender**', '**$description**')";

$result = mysqli\_query($connection, $query) **or** **die**("Error Querying database");

mysqli\_close($connection);

**endif**;

|  |  |
| --- | --- |
| **Terms** | **Explanation** |
| Mysqli | “i” over here stands for improved |
| mysqli\_connect | This allows us to connect to the database |
| $server\_name | It is “localhost” by default if we are running website “locally” |
| $user\_name | “root”, by default phpMyAdmin gives username as root |
| $password | “”, by default, there is no password |
| $db\_name | Database name |
| mysqli\_query() | It allows us to send the SQL command using PHP |
| mysqli\_close() | Connection should be closed or else the DB might get corrupted |
| **die**() | This function terminates a PHP script and gives the feedback of the error |

To be sure check your phpMyAdmin home page



Fig 9: Checking the server name, user name and the password

#### Database check in actual website

First go to **your-name-of-the-website/cpanel**

Ex: [**www.sphotonkhan/cpanel**](http://www.sphotonkhan/cpanel)

Before going to phpMyAdmin, go to MySQL® Database Wizard



Fig 10: MySQL Database Wizard (1st Step)



Fig 11: MySQL Database Wizard (2st Step)

|  |  |
| --- | --- |
| $server\_name | bh-40 |
| $user\_name | sphotonkhan |
| $password | photon |
| $db\_name | sphotonk\_test |

To get the server information go to Server Information

Fig 12: General Information 🡪 Server Information



Fig 13: Server Information

#### Selecting all the data from the table

<?php

$server\_name = "localhost";

$user\_name = "root";

$password = "";

$db\_name = "form\_table";

$connection = mysqli\_connect($server\_name, $user\_name, $password, $db\_name)

**or** **die**("Could not connect to the database");

$query = "SELECT \* FROM form";

$result = mysqli\_query($connection, $query)

**or** **die**("Query not working");

**while**($row = mysqli\_fetch\_array($result)){

$id = $row['id'];

$first\_name = $row['first\_name'];

$last\_name = $row['last\_name'];

$email = $row['email'];

$gender = $row['gender'];

$description = $row['description'];

**echo** nl2br("Name: **$first\_name** **$last\_name\n**");

**echo** nl2br("Email: **$email\n**");

**echo** nl2br("Gender: **$gender\n**");

**echo** nl2br("Description: **$gender\n\n**");

}

mysqli\_close($connection);

?>

Code 13: Select.php

### Web Application

A web application is a dynamic web site that designed to fulfill a particular purpose for it users.

A database is a container storing data in a very structured way. Table stores data in a grid-like pattern of columns and rows. Columns consists of specific type of data. Rows are collection of columns where a single row consists one of each column. Table rows are horizontal, and table columns are vertical.

**Application Purpose**: Customer wants to add into the newsletter program for a particular shop, i.e. get emails about the discounts and so on. Customer also have the authority to get removed from the newsletter program.

    

Add Email PHP Script PHP Engine Server Database

    

Send Email PHP Script PHP Engine Server Client



Database

Fig 14: Web Application

CREATE DATABASE is the SQL command used to create a new database.

CREATE DATABASE database\_name

Ex: CREATE DATABASE store

Text data takes more room to store than integer data. To create a table, we need to know what type of data is stored in each table column.

**Column Types**

CHAR: Character, Highly efficient for the texts with fixed length

INT: Integer, Whole Number, Negative numbers can also be stored

TINYINT: Small Integer

BLOB: Large gobs of binary data

DEC: Decimal

DATETIME: She keeps track of the date

TIMESTAMP: She keeps track of the date

DATE: Keep track of dates

VARCHAR: Variable Character

TEXT: Store huge amount of text

CREATE TABLE SQL is used to create a new table in a database.

CREATE TABLE table\_name (column name column type, …)

It is not possible to recreate the same table using CREATE TABLE

USE command selects a database as the default database for subsequent SQL statements

Ex: USE store

DESCRIBE reveals the structure of tables

Ex: DESCRIBE store

DROP deletes a table and all its data from the database

Ex: DROP TABLE store

### Add Email Script

First Create Table

CREATE TABLE email\_list(

id INT AUTO\_INCREMENT,

first\_name VARCHAR(250),

last\_name VARCHaR(250),

email VARCHAR(60),

PRIMARY KEY(id)

)

<!doctype html>

<**html**>

<**head**>

<**title**> Add Email </**title**>

<**meta** charset = "UTF-8"/>

</**head**>

<**body**>

<**p**> Enter Your Details </**p**>

<**form** method = "post" action = "addemail.php">

<**label** for = "first\_name"> First Name: </**label**>

<**input** type = "text" name = "first\_name" id = "first\_name"><**br**/>

<**label** for = "last\_name"> Last Name: </**label**>

<**input** type = "text" name = "last\_name" id = "last\_name"><**br**/>

<**label** for = "email"> Email: </**label**>

<**input** type = "email" name = "email" id = "email"><**br**/>

<**input** type = "submit" name = "submit" value = "Submit"/>

</**form**>

</**body**>

</**html**>

Code 14: addemail.html

<?php

$server\_name = "localhost";

$user\_name = "root";

$password = "";

$db\_name = "store";

**if**(!**empty**($\_POST['submit'])):

$connection = mysqli\_connect($server\_name,

$user\_name,

$password, $db\_name)

**or** **die**("Could not connect to the server");

$first\_name = $\_POST['first\_name'];

$last\_name = $\_POST['last\_name'];

$email = $\_POST['email'];

$sql = "INSERT INTO email\_list(first\_name,

last\_name,

email)

VALUES('**$first\_name**','**$last\_name**','**$email**')";

$result = mysqli\_query($connection, $sql)

**or** **die**("Query Denied");

**echo** nl2br("<h1>Confirmation Message</h1> **\n**");

**echo** nl2br("<p>Name: **$first\_name** **$last\_name**</p>");

**echo** nl2br("<p>Email: **$email**</p> **\n**");

mysqli\_close($connection);

**endif**;

?>

Code 15: addemail.php

<!doctype html>

<**htmL**>

<**head**>

<**title**> Send Email </**title**>

<**meta** charset = "UTF-8"/>

</**head**>

<**body**>

<**form** method = "post" action = "sendemail.php">

<**label** for = "subject">Subject</**label**><**br**/>

<**input** type = "text" name = "subject" id = "subject"/><**br**/>

<**label** for = "body\_of\_email"> Body of Email </**label**><**br**/>

<**textarea** name = "body\_of\_email" id = "body\_of\_email"

rows = "8" cols = "60"></**textarea**><**br**/>

<**input** type = "submit" id = "submit" name = "submit"

value = "Submit"/>

</**body**>

</**htmL**>

Code 16: sendemail.html

<?php

**if**(!empty($\_POST['submit'])):

$from = "khan.photon@gmail.com";

$subject = $\_POST['subject'];

$body\_of\_email = $\_POST['body\_of\_email'];

$server\_name = "localhost";

$user\_name = "root";

$password = "";

$db\_name = "store";

$connection = mysqli\_connect($server\_name,

$user\_name,

$password,

$db\_name)

**or** **die**("Could connect to the server");

$query = "SELECT \* FROM email\_list";

$result = mysqli\_query($connection, $query)

**or** **die**("Query Denied");

**while**($row = mysqli\_fetch\_array($result)){

$to = $row['email'];

mail($to, $subject, $body\_of\_email, "From: **$from**");

}

mysqli\_close($connection);

**endif**;

?>

Code 17: sendemail.php

mysqli\_fetch\_array($result)

This function stores a row of data in an array.

A while loop repeats code while a condition is met. $row is interpreted as true since it isn’t set to 0 or false. When there is no more data available, mysqli\_fetch\_array($result) return false. Anything other than 0 or False is always interpreted as true.

### Removing data using DELETE and WHERE

A where clause narrows down a query to focus on specific rows of data. A where clause in a DELETE statement let us pinpoint the row we want to remove.

DELETE FROM email\_list

WHERE email = “khan.photon@gmail.com”

email = Name of the table column

[khan.photon@gmail.com](mailto:khan.photon@gmail.com) = Check whether this exists in the database.

<!doctype html>

<**html**>

<**head**>

<**title**> Remove Email </**title**>

<**meta** charset = "UTF-8"/>

</**head**>

<**body**>

<**p**> Enter Email Address to Remove </**p**><**br**/>

<**form** method = "post" action = "removeemail.php">

<**label** for = "email">Email</**label**><**br**/>

<**input** type = "email" name = "email" id = "email"/><**br**/>

<**input** type = "submit" value = "Submit" name = "submit"/>

</**form**>

</**body**>

<**html**>

Code 18: removeemail.html

<?php

$server\_name = "localhost";

$user\_name = "root";

$password = "";

$db\_name = "store";

**if**(!**empty**($\_POST)):

$email = $\_POST['email'];

$connection = mysqli\_connect($server\_name,

$user\_name,

$password,

$db\_name)

**or** **die**("Could not connect to the server");

$query = "DELETE FROM email\_list where email = '**$email**'";

mysqli\_query($connection, $query) **or**

**die**("Query Denied");

**echo** nl2br("**$email** has been successfully removed**\n**");

mysqli\_close($connection);

**endif**;

?>

Code 19: removeemail.php

### Server-Side Validation

It assures the data we get is the data we expect.

isset(): It tests to see if a variable exists, which means that it’s been assigned a value.

empty(): Determines where the variable contains empty value

PHP defines as 0, an empty string (‘’ or “”), or the values false or NULL

PHP logic operators make it possible to structure more elegant if statements.

Logical AND is coded as && while logical OR is coded as ||

<!doctype html>

<**htmL**>

<**head**>

<**title**> Send Email </**title**>

<**meta** charset = "UTF-8"/>

</**head**>

<**body**>

<**form** method = "post" action = "sendvalidatedemail.php">

<**label** for = "subject">Subject</**label**><**br**/>

<**input** type = "text" name = "subject" id = "subject"/><**br**/>

<**label** for = "body\_of\_email"> Body of Email </**label**><**br**/>

<**textarea** name = "body\_of\_email" id = "body\_of\_email"

rows = "8" cols = "60"></**textarea**><**br**/>

<**input** type = "submit" id = "submit" name = "submit"

value = "Submit"/>

</**body**>

</**htmL**>

Code 20: sendvalidatedemail.html

<?php

**if**(**empty**($\_POST['subject']) && !**empty**($\_POST['body\_of\_email'])):

**echo** "Your subject is empty!";

**elseif**(**empty**($\_POST['body\_of\_email']) && !**empty**($\_POST['subject'])):

**echo** "Your body of email is empty!";

**elseif**(**empty**($\_POST['subject']) && **empty**($\_POST['body\_of\_email'])):

**echo** "Your subject and body of email are empty!";

**elseif**(!empty($\_POST['submit']) && !**empty**($\_POST['subject']) && !**empty**($\_POST['body\_of\_email'])):

$from = "khan.photon@gmail.com";

$subject = $\_POST['subject'];

$body\_of\_email = $\_POST['body\_of\_email'];

$server\_name = "localhost";

$user\_name = "root";

$password = "";

$db\_name = "store";

$connection = mysqli\_connect($server\_name,

$user\_name,

$password,

$db\_name)

**or** **die**("Could connect to the server");

$query = "SELECT \* FROM email\_list";

$result = mysqli\_query($connection, $query)

**or** **die**("Query Denied");

**while**($row = mysqli\_fetch\_array($result)){

$to = $row['email'];

mail($to, $subject, $body\_of\_email, "From: **$from**");

}

mysqli\_close($connection);

**else**:

**endif**;

?>

Code 21: sendvalidatedemail.php

#### Self-Referencing and Sticky Forms

An HTML form that is part of the PHP script that processes it is known as **self-referencing.**

**Sticky forms** remember the data the user has already correctly entered.

$\_SERVER[‘PHP\_SELF’] stores away the name of the current script

<?php

$error = [];

$error['sub'] = "";

$error['body'] = "";

**if**(**empty**($\_POST['subject']) && !**empty**($\_POST['body\_of\_email'])):

$error['sub'] = "Your subject is empty!";

**endif**;

**if**(**empty**($\_POST['body\_of\_email']) && !**empty**($\_POST['subject'])):

$error['body'] = "Your body of email is empty!";

**endif**;

**if**(**empty**($\_POST['subject']) && **empty**($\_POST['body\_of\_email'])):

$error['sub'] = "Your subject is empty!";

$error['body'] = "Your body of email is empty!";

**endif**;

**if**(!**empty**($\_POST['submit']) && !**empty**($\_POST['subject']) && !**empty**($\_POST['body\_of\_email'])):

$from = "khan.photon@gmail.com";

$subject = $\_POST['subject'];

$body\_of\_email = $\_POST['body\_of\_email'];

$server\_name = "localhost";

$user\_name = "root";

$password = "";

$db\_name = "store";

$connection = mysqli\_connect($server\_name,

$user\_name,

$password,

$db\_name)

**or** **die**("Could connect to the server");

$query = "SELECT \* FROM email\_list";

$result = mysqli\_query($connection, $query)

**or** **die**("Query Denied");

**while**($row = mysqli\_fetch\_array($result)){

$to = $row['email'];

mail($to, $subject, $body\_of\_email, "From: **$from**");

**echo** "Your email has been successfully sent";

}

mysqli\_close($connection);

**endif**;

?>

<!doctype html>

<htmL>

<head>

<title> Send Email </title>

<meta charset = "UTF-8"/>

</head>

<body>

<form method = "post" action = "<?php **echo** $\_SERVER['PHP\_SELF'];?>">

<label for = "subject">Subject</label><br/>

<input type = "text" name = "subject" id = "subject"

value = "<?php **if**(!**empty**($\_POST['subject'])) **echo** $\_POST['subject']; ?>"/>

<?php **if**($error['sub'] && !**empty**($\_POST['submit'])) **echo** $error['sub'];?><br/>

<label for = "body\_of\_email"> Body of Email </label><br/>

<textarea name = "body\_of\_email" id = "body\_of\_email"

rows = "8" cols = "60"><?php **if**(!**empty**($\_POST['body\_of\_email'])) **echo** $\_POST['body\_of\_email'];?></textarea>

<?php **if**($error['body'] && !**empty**($\_POST['submit'])) **echo** $error['body'];?><br/>

<input type = "submit" id = "submit" name = "submit"

value = "Submit"/>

</body>

</htmL>

Code 22: sendemailupdated.php

### Avoid duplicate entry in Database

Table rows should be **uniquely identifiable.** Therefore, we will create a unique integer column, also called a **primary key**

To alter the table that already exists

ALTER TABLE table\_name ADD column\_name, column\_type

Location: first in the table

ALTER TABLE email\_list ADD id INT NOT NULL AUTO\_INCREMENT FIRST, ADD PRIMARY KEY(id)

Column that we want to add Increase the value of id by 1 automatically id is the primary key

A primary key is a column in our table that makes each row unique

<!doctype html>

<html>

<head>

<title> Remove Email </title>

<meta charset = "UTF-8"/>

</head>

<body>

<p> Delete Email Addresses </p>

<?php

$showform = **TRUE**;

$server\_name = "localhost";

$user\_name = "root";

$password = "";

$db\_name = "store";

$connection = mysqli\_connect($server\_name,

$user\_name,

$password,

$db\_name)

**or** **die**("Could not connect to the server");

**if**(isset($\_POST['submit'])):

$email = $\_POST['email'];

$showform = **FALSE**;

$query = "SELECT \* FROM email\_list WHERE email = '**$email**'";

$result = mysqli\_query($connection, $query) **or**

**die**("Select Query Denied");

**echo** "<form method = 'post' action = '".$\_SERVER['PHP\_SELF']."'>";

**while**($row = mysqli\_fetch\_array($result)){

$id = $row['id'];

$name = $row['first\_name']." ".$row['last\_name'];

$email = $row['email'];

**echo** "<input type = 'checkbox' value = **\"$id\**" name = \"todelete[] **\"**/>";

**echo** "**$name** **$email**<br/>";

}

**echo** "<br/><input type = 'submit' value = 'Remove' name = 'remove'/>";

**echo** "</form>";

**endif**;

**if**(isset($\_POST['remove'])):

$showform = **FALSE**;

**foreach**($\_POST['todelete'] **as** $delete\_id){

$query = "DELETE FROM email\_list where id = '**$delete\_id**'";

mysqli\_query($connection, $query) **or** **die** ("Remove Query Denied");

}

**echo** "Customers Removed Successfully";

**endif**;

?>

<?php

**if**($showform):

?>

<form method = "post" action = "<?php **echo** $\_SERVER['PHP\_SELF'];?>">

<label for = "email">Email</label><br/>

<input type = "email" name = "email" id = "email"/><br/>

<input type = "submit" value = "Submit" name = "submit"/>

</form>

</body>

<html>

<?php

**endif**;

?>

Code 23: removeemailupdated.php

### Inventory (ALTER TABLE)

Making a web application related to a shop

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Date Added** | **Product** | **Price** | **Image** |
| 1 | 2007-04-22 14:37:34 | Lip stick | 25.00 | --- |
| 2 | 2009-01-12 21:27:54 | Chocolate | 5.23 | --- |
| 3 | 2011-12-23 09:06:35 | Shampoo | 40.15 | --- |
| 4 | 2013-08-04 09:12:53 | Shoe | 150.14 | --- |
| 5 | 2012-02-21 14:09:50 | Soap | 9.19 | --- |

The Alter statement is used to change the structure of a database

ALTER TABLE inventory ADD COLUMN age TINYINT

Add a column to a table

ALTER TABLE inventory CHANGE COLUMN image screenshot VARCHAR(255)

Change the name and data type of the column

ALTER TABLE inventory DROP COLUMN age

Drops the table

ALTER TABLE inventory MODIFY COLUMN date DATETIME AFTER age

Only First & AFTER can be used to alter the position

Changes the data type or position of a column within a table

### Uploading a File

<form enctype = "multipart/form=data" method = "post"

action = "<?php echo **$\_SERVER['PHP\_SELF']**;?>">

enctype attribute tells the form to use a special type of encoding required for the upload – it affects how the POST data is bundled and sent when the form is submitted

<input type = "hidden" name = "MAX\_FILE\_SIZE" value = "32768"/>

Establishes the maximum file size for file uploads in this case, 32 KB (32,768)

INSERT INTO inventory VALUES(NOW(), ‘$name’, ‘$price’, ‘$image’)

NOW() function is used to insert the current date/time.

Image filenames are stored in the database as part of an INSERT statement

$\_FILES

It is a built-in super global variable provides access to information about uploaded files.

$\_FILES[‘image’][‘name’]

The name of the uploaded file

$\_FILES[‘image’][‘type’]

MIME type of the uploaded file

$\_FILES[‘image’][‘tmp\_name’]

Temporary storage location of the file on the server

$\_FILES[‘error’][‘error’]

The error code for the file upload, 0 indicates a success, other values indicate failure

move\_uploaded\_file($\_FILES[‘image’][‘tmp\_name’], $target)

Placing an image on a web page only requires reference to the image file.

The file that is being uploaded actually stores in a temporary folder on the server. Temporary folder after a certain time deletes the file. Therefore. to save the file, we need to use move\_uploaded\_file() function to move the file from temporary to the current directory.

UPLOAD\_PATH = “images/”

It is a constant, it values stays the same

Define a constant using define(‘UPLOAD\_PATH’, ‘/images’)

$target = UPLOAD\_PATH.$FILES[‘image’][‘name]

Therefore, the target location is “images/name-of-the-file.png”

<!doctype html>

<html>

<head>

<title> Inventory Index </title>

<meta charset = "UTF-8"/>

</head>

<body>

<h1> Inventory **List** </h1>

<p><a href = "addinventory.php">Add Item</a></p>

<p> **List** of items are given below</p>

<?php

$server\_name = "localhost";

$user\_name = "root";

$password = "";

$db\_name = "store";

define("UPLOAD\_PATH","images/");

$connection = mysqli\_connect($server\_name, $user\_name,

$password, $db\_name)

**or** **die**("Could not conenct to the database");

$query = "SELECT \* FROM inventory";

$result = mysqli\_query($connection, $query);

**echo** "<table>";

**echo** "<tr>";

**echo** "<th>Date</th>";

**echo** "<th>Name</th>";

**echo** "<th>Price</th>";

**echo** "<th>Image</th>";

**echo** "</tr>";

**while**($row = mysqli\_fetch\_array($result)){

**echo** "<tr>";

**echo** "<td>".$row['date']."</td>";

**echo** "<td>".$row['name']."</td>";

**echo** "<td>".$row['price']."</td>";

**echo** "<td><img src ='".UPLOAD\_PATH.$row['image']."' alt = 'Inventory Image'

height = '20px' width = '20px'/></td>";

**echo** "</tr>";

}

**echo** "</table>";

?>

</body>

</html>

Code 24: inventoryindex.php

<?php

$show\_form = **TRUE**;

define("UPLOAD\_PATH","images/");

$server\_name = "localhost";

$user\_name = "root";

$password = "";

$db\_name = "store";

$connection = mysqli\_connect($server\_name, $user\_name, $password, $db\_name)

**or** **die**("Server Connection Denied");

**if**(isset($\_POST['submit'])):

$name = $\_POST['item\_name'];

$price = $\_POST['price'];

$image\_name = $\_FILES['image']['name'];

**if**(!**empty**($name) && !**empty**($price) && !**empty**($image\_name)):

$show\_form = **FALSE**;

$target = UPLOAD\_PATH.$image\_name;

*// $date = date("Y-m-d h:i:s");*

$query = "INSERT INTO inventory VALUES(0, NOW(), '**$name**', '**$price**', '**$image\_name**')";

mysqli\_query($connection, $query) **or** **die**("Query Denied");

move\_uploaded\_file($\_FILES['image']['tmp\_name'], $target);

**echo** "Inventory Added Successfully";

**endif**;

**endif**;

?>

<?php

**if**($show\_form):

?>

<!doctype html>

<html>

<head>

<title> Add Inventory </title>

<meta charset = "UTF-8"/>

</head>

<body>

<p> Add Inventory </p>

<form enctype = "multipart/form-data" method = "post"

action = "<?php **echo** $\_SERVER['PHP\_SELF'];?>">

<!-- <input type = "hidden" name = "MAX\_FILE\_SIZE" value = "32768"/> -->

<label for = "item\_name"> Item Name: </label><br/>

<input type = "text" name = "item\_name" id = "item\_name"/><br/>

<label for = "price"> Price: </label><br/>

<input type = "number" name = "price" id = "price" step = "0.01"/><br/>

<input type = "file" name = "image" id = "image"/><br/><br/>

<input type = "submit" value = "Add Item" name = "submit"/>

</body>

</html>

<?php

**endif**;

?>

Code 25: addinventory.php

#### Check File or not

<?php

$file\_path = "images/chips.png";

**if**(is\_file($file\_path) && filesize($file\_path)):

**echo** "It is a file";

**endif**;

?>

Code 26: checkfile.php

Databases are great for storing text data, but it’s normally better for them to reference binary data in external files.

### Sharing Data

Shared script data needs to be accessible throughout an application without code duplication. Include files allows us to share code across multiple scripts

require\_once, the name “include file” comes from a PHP statement called include that is very similar to require\_once. The difference is that require\_once results in an error if the included file cannot be found. “include” won’t reveal an error if an include file is missing. There are in total four of them: include, include\_once, require, require\_once.

Therefore, **require** and **require\_once,** will shot **fatal error** if the file is missing

Therefore, **include** and **include\_once**, will show warning and continue load the rest of the page.

Therefore, difference between **include** and **include\_once**, if the code from a file has been already included then it will not be included again if we use **include\_once()**. Means **include\_once()** include the file only once at a time. Same for **require** and **require\_once.**

require\_once statement inserts shared script code into other scripts.

Let’s reorganize the previous files

### Homework

1. Combine HTML(form.html) and PHP file(report.php) into one PHP file (form.php)
2. Edit (addemail.php, addemail.html) into one PHP file(addemailupdated.php) and this time only unique email addresses would be added to the database.
3. Edit (addinventory.php) to see whether the file uploaded is a file or not.

## GitHub

GitHub is a code hosting platform for version control and collaboration. It lets us work together on projects from anywhere. Create a GitHub account first.

### STEP 1: Create a Repository

It is usually used to organize a single project. Repositories can contain folders and files, images, videos, spreadsheet, and data-sets anything that our project needs

* In the upper right corner, next to your avatar or identicon, click + and then select New repository.
* Name your repository hello-world.
* Write a short description.
* Select Initialize this repository with a README.



Fig 1. GitHub, Creating New Repository

### STEP 2: Branching

It is the way to work on different versions of a repository at one time. By default, our repository has one branch named master which is considered to be the definitive branch. We use branches to experiment and make edits before committing them to master. When we create a branch off the master branch, you’re making a copy, or snapshot, of master as it was at that point in time. If someone else made changes to the master branch while we were working on your branch, we could pull in those updates.



Fig 2. GitHub, Branching

#### To create a new branch

Go to your new repository hello-world.

Click the drop down at the top of the file list that says branch: master.

Type a branch name, readme-edits, into the new branch text box.

Select the blue Create branch box or hit “Enter” on your keyboard.

Now we have two branches, master and readme-edits. They look exactly the same. Next, we’ll add our changes to the new branch.

### STEP 3: Commit Changes

We are on the code view for our readme-edits branch, which is a copy of master.

On GitHub, **saved changes** are called **commits**. Each commit has an associated commit message, which is a description explaining why a particular change was made. Commit messages capture the history of our changes, so other contributors can understand what we have done and why.

### STEP 4: Open a Pull Request

When we open a pull request, we are proposing our changes and requesting that someone review and pull in your contribution and merge them into their branch. Pull requests show diffs, or differences, of the content from both branches. The changes, additions, and subtractions are shown in green and red.

As soon as we make a commit, we can open a pull request and start a discussion, even before the code is finished. By using GitHub’s **@mention** system in your pull request message, you can ask for feedback from specific people or teams. We can even open pull requests in your own repository and merge them ourselves.



Fig 3. GitHub, Open a Pull Request

### STEP 5: Merge our Pull Request

In this final step, it’s time to bring our changes together – merging our readme-edits branch into the master branch.

Click the green Merge pull request button to merge the changes into master.

Click Confirm merge.

Go ahead and delete the branch, since its changes have been incorporated, with the Delete branch button in the purple box.



Fig 4. GitHub, Merge the Pull Request