Chp:5 Introduction to server side issues

Prepared by: Er. Simanta kasaju

Authentication

- Authentication: Authentication is the process of determining the identity of a user based on the user's credentials.

- The user's credentials are usually in the form of user ID and password, which is checked against any credentials' store such as database. If the credentials provided by the user are valid, then the user is considered an authenticated user.

- there are different ways in which authentication is performed as discussed below:

- 1) Anonymous Access: There is no authentication performed and the user is treated as anonymous user by IIS.

- Sometimes it is necessary or possible to access any data from remote server or database without any authenticated person.

- If the data is easily accessible without any authentication (i.e username and password) that means the user accessing those data or file has an anonymous access.

- There is no difference between a user who is "anonymously authenticated" and an unauthenticated user. There can be many situations where anonymous authentication is useful.

Cont..

- Windows Authentication: Provides information on how to use Windows authentication in conjunction with

- Microsoft Internet Information Services (IIS) authentication to secure ASP.NET applications.

- Windows Authentication uses the security features of Windows clients and servers. Unlike Basic authentication, initially, it does not prompt users for a user name and password. The current Windows user information on the client computer is supplied by the web browser through a cryptographic exchange involving hashing with the Web server.

- If the authentication exchange initially fails to identify the user, the web browser will prompt the user for a Windows user account user name and password.

- Windows authentication is generally used if the users accessing the application belong to same organization.

- This authentication method uses Windows accounts for validating users' credentials. This type of authentication is very good for intranet Web sites where we know our users.

Authorization

- After successful authentication ,deciding which resources a user can access based on their identify and checking whether the authenticated user has sufficient rights to access the requested resource is authorization.

Impersonation

It is a process In which user accesses the resources(Ex: Files,DB) by using the identify of another user.

Integrated windows Authentication

- Read yourself

Database Connectivity

- In working with databases, the following are the concepts which are common to all databases.

- To work with the data in a database, the first step is the connection. The connection to a database normally consists of the given parameters.

- **Database name or Data Source** – The first important parameter is the database name to which the connection needs to be established. Each connection can only work with one database at a time.

- **Credentials –** The next important aspect is the username and password which needs to be used to establish a connection to the database. It ensures that the username and password have the necessary privileges to connect to the database.

- **Optional parameters -** For each database type, you can specify optional parameters to provide more information on how php should handle the connection to the database. For example, one can specify a parameter for how long the connection should stay active. If no operation is performed for a specific period of time, then the parameter would determine if the connection has to be Selecting data from the database – Once the connection has been established, the next important aspect is to fetch the data from the database. php can execute 'SQL' select command against the database. The 'SQL' statement can be used to fetch data from a specific table in the database.

- **Inserting data into the database – php** can also be used to insert records into the database. Values can be specified in php for each row that needs to be inserted into the database.

- **Updating data into the database – php** can also be used to update existing records into the database. New values can be specified in php for each row that needs to be updated into the database.

- **Deleting data from a database – php** can also be used to delete records into the database. Select commands to specify which rows need to be deleted can be specified in php.

SQL

- SQL is a database computer language designed for the retrieval and management of data in a relational database.

• All task related to relational data management creating tables, querying the database for information, modifying the data in the database ,deleting them, granting access to users and so on can be done using SQL.

SQL is based on relational algebra and tuple relational calculus.

• SQL stands for Structured Query language, pronounced as "S-Q-L" or sometimes as "See-Quel"... Relational databases like MySQL Database, Oracle, MS SQL Server, Sybase, etc. use ANSI SQL.

cont..

- SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in relational database.

- **SQL (Structured Query Language)** is a special-purpose programming language designed for managing data in relational database management systems (RDBMS).

- SQL is a standardized query language for requesting information from a database.

Basic SQL Queries (CRUD)

- A query is a question or a request.
- C refers to Create, R refers to Read, U refers to Update and D refers to Delete.

CREATE

- It is used to create a new table or new database in the database.
- CREATE DATABASE database_name;
- CREATE TABLE TABLE_NAME (COLUMN_NAME DATATYPES[,....]);
- CREATE TABLE EMPLOYEE(Name VARCHAR(20), Email VARCHAR(100), DOB DATE);
- Example

CREATE DATABASE DB;//creating database

CREATE TABLE Persons (PersonID int, LastName varchar(255), FirstName varchar(255), Address varchar(255), City varchar(255));

INSERT

- The INSERT statement is a SQL query.

- It is used to insert data into the row of a table.

- INSERT INTO TABLE_NAME (col1, col2, col3,…. col N) VALUES (value1, value2, value3, …. valueN);

Or

INSERT INTO TABLE_NAME VALUES (value1, value2, value3, …. valueN);

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country) VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');

SELECT

- It is used to select the attribute based on the condition described by WHERE clause.

SELECT expressions FROM TABLES
WHERE conditions;
 Example SELECT emp_name FROM employee WHERE age > 20;

UPDATE

- • This command is used to update or modify the value of a column in the table.

- Note: Be careful when updating records in a table! Notice the WHERE clause in the UPDATE statement. The WHERE clause specifies which record(s) that should be updated. If you omit the WHERE clause, all records in the table will be updated!

- Syntax

- UPDATE table_name SET column1 = value1, column2 = value2, ... WHERE condition;

```
UPDATE Customers SET ContactName = 'Alfred Schmidt', City=
'Frankfurt' WHERE CustomerID = 1;
```

DELETE

- It is used to remove one or more row from a table.

Syntax

DELETE FROM table_name WHERE condition;

- Note: Be careful when deleting records in a table! Notice the WHERE clause in the DELETE statement. The WHERE clause specifies which record(s) should be deleted. If you omit the WHERE clause, all records in the table will be deleted!

Database connectivity

- There are three types of methods in PHP to connect MySQL database through backend:

- MySQL

- MySQLi

- PDO

- mysql() is now obsolete because of security issues like SQL injection etc, but the other two are being actively used.

MySQLi

- MySQLi is an API used as a connector function to link the backend of the PHP app to the MySQL database.

- It works just like the previous version, but it is safer and faster, and provides a better set of functions and extensions.

- MySQLi was introduced with PHP 5.0.0 and the drivers were installed in 5.3.0. The API was designed to support MySQL from version 4.1.13 to newer ones.

PDO

- PHP Data Objects (PDO) extension is a Database Abstraction Layer. It is like an interface for the backend to interact with the MySQL database and make changes without making any change in the PHP code. It also gives you the freedom to work with multiple databases. The major advantage of using PDO is that your code stays simple and portable.

1. Connect MySQL using Localhost Server

2. Connect MySQL using Cloudways Server

3. Connect MySQL using PDO

4. Connect MySQL using Remote MySQL

Open a Connection to MySQL
Before we can access data in the MySQL database, we need to be able to connect to the server:

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = mysqli_connect($servername, $username, $password);

// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?>
```

By default
 username ="root"
Password: " "

mysqli_connect
_error()

Close the Connection

- The connection will be closed automatically when the script ends. To close the connection before, use the following:

- `$conn->close();`

- `mysqli_close($conn);`

Create a MySQL Database

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";

// Create connection
$conn = mysqli_connect($servername, $username, $password);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}
// Create database
$sql = "CREATE DATABASE myDB";
$result= mysqli_query($conn, $sql))
if ($result){
  echo "Database created successfully";
} else {
  echo "Error creating database: " . mysqli_error($conn);
}
mysqli_close($conn);
?>
```

Create a MySQL Table

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}
// sql to create table
$sql = "CREATE TABLE Teacher (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
gender VARCHAR(50),
salary int
)";
```

Insert values

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}
$sql = "INSERT INTO teacher (firstname, lastname, email,gender,salary)
VALUES ('simanta', 'Kasaju', 'sim.kasaju@gmail.com','male',50000)";
if (mysqli_query($conn, $sql)) {
  echo "New record created successfully";
} else {
  echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}
mysqli_close($conn);
?>
```

Insert Multiple values

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}
$sql = "INSERT INTO teacher (firstname, lastname, email,gender,salary)
VALUES ('prashant', 'Kasaju', 'pra.kasaju@gmail.com','male',60000),
 ('salina', 'shrestha', 'sali.kasaju@gmail.com','Female',30000),
('shashi', 'shrestha', 'shashi.kasaju@gmail.com','Female',20000)";

if (mysqli_query($conn, $sql)) {
  echo "New record created successfully";
} else {
  echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}
mysqli_close($conn);
?>
```

Select values from database

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

$sql = "SELECT id, firstname, lastname FROM teacher";
$result = mysqli_query($conn, $sql);
$num= mysqli_num_rows($result) ;
if ($num> 0) {
  // output data of each row
  while($row = mysqli_fetch_assoc($result)) {
    echo "id: " . $row["id"]. " -
 Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
  }
} else {
  echo "0 results";
}
mysqli_close($conn);
?>
```

First, we set up an SQL query that selects the id, firstname and lastname columns from the teacher table. The next line of code runs the query and puts the resulting data into a variable called $result.

Then, the function num_rows() checks if there are more than zero rows returned.

If there are more than zero rows returned, the function fetch_assoc() puts all the results into an associative array that we can loop through. The while() loop loops through the result set and outputs the data from the id, firstname and lastname columns.

Select values from database and show them in tabular form

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}
?>
<!doctype html>
<head></head>
<body>
    <table border="2">
        <th>ID</th>
        <th>Firstname</th>
        <th>Lastname</th>
        <th>Email</th>
        <th>Gender</th>
        <th>Salary</th>
</thead>
    <tbody>
```

```php
<?php
$sql = "SELECT * FROM teacher";
$result = mysqli_query($conn, $sql);
$num= mysqli_num_rows($result) ;
if ($num> 0) {
    // output data of each row
    while($row = mysqli_fetch_assoc($result)) {

        echo "<tr>
            <td>". $row['id'] ."</td>
            <td>".$row['firstname']."</td>
            <td>".$row['lastname']."</td>
            <td>".$row['email']."</td>
            <td>".$row['gender']."</td>
            <td>".$row['salary']."</td>
            </tr>";
} }
mysqli_close($conn);
?>
</tbody>
</table>
</body>
</html>
```

O/P

```
$sql = "SELECT * FROM teacher";
```

ID	Firstname	Lastname	Email	Gender	Salary
1	simanta	Kasaju	sim.kasajju@gmail.com	male	50000
2	puran	Kasaju	puran.kasaju@gmail.com	male	40000
3	shashi	shrestha	shashi.kasaju@gmail.com	Female	20000
5	prashant	Kasaju	pra.kasaju@gmail.com	male	60000
6	salina	shrestha	sali.kasaju@gmail.com	Female	30000

Using Where clause

```
$sql = "SELECT * FROM teacher WHERE firstname='simanta'";
```

ID	Firstname	Lastname	Email	Gender	Salary
1	simanta	Kasaju	sim.kasajju@gmail.com	male	50000

```
$sql = "SELECT * FROM teacher WHERE salary>50000";
```

ID	Firstname	Lastname	Email	Gender	Salary
5	prashant	Kasaju	pra.kasaju@gmail.com	male	60000

```
$sql = "SELECT * FROM teacher WHERE salary>60000 and
gender='male' ";
```

ID	Firstname	Lastname	Email	Gender	Salary

Update Data

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDB";
// Create connection
$conn = mysqli_connect($servername,
$username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli
_connect_error());
}
$sql
```

```php
= "update Teacher set
lastname='shretha' where id=1";
$result = mysqli_query($conn, $sql);

if($result){
    echo "updated succesfully";
}
else{
echo "error occurs" .mysqli_error($conn);}
mysqli_close($conn);
?>
```

ID	Firstname	Lastname	Email	Gender	Salary
1	simanta	shretha	sim.kasajju@gmail.com	male	50000
2	puran	Kasaju	puran.kasaju@gmail.com	male	40000
3	shashi	shrestha	shashi.kasaju@gmail.com	Female	20000
5	prashant	Kasaju	pra.kasaju@gmail.com	male	60000
6	salina	shrestha	sali.kasaju@gmail.com	Female	30000

DELETE DATA

```
$sql = "delete from Teacher
 where gender='female' ;
```

ID	Firstname	Lastname	Email	Gender	Salary
1	simanta	shretha	sim.kasajju@gmail.com	male	50000
2	puran	Kasaju	puran.kasaju@gmail.com	male	40000
3	prashant	Kasaju	pra.kasaju@gmail.com	male	60000

Order By

- SELECT column_name(s) FROM table_name ORDER BY column_name(s) ASC|DESC

$sql = "SELECT id, firstname, lastname FROM Teacher ORDER BY Lastname";

PHP File Handling

- File handling is needed for any application. For some task to be done file needs to be processed. File handling in php is similar as file handling is done by using any programming language like c.Php has many function to work with normal files. Those functions are

readfile() Function

- The readfile() function reads a file and writes it to the output buffer.

- ```php
<?php
echo readfile("webdictionary.txt");
?>
```

# PHP File Open/Read/Close

- PHP Open File - fopen()

- A better method to open files is with the fopen() function. This function gives you more options than the readfile() function

```php
<?php
$myfile =
fopen("webdictionary.txt", "r") or die("Unable
to open file!");
echo fread($myfile,filesize("webdictionary.txt"
));
fclose($myfile);
?>
```

The file may be opened in one of the following modes:

Modes	Description
r	**Open a file for read only**. File pointer starts at the beginning of the file
w	**Open a file for write only**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a	**Open a file for write only**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x	**Creates a new file for write only**. Returns FALSE and an error if file already exists
r+	**Open a file for read/write**. File pointer starts at the beginning of the file
w+	**Open a file for read/write**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a+	**Open a file for read/write**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x+	**Creates a new file for read/write**. Returns FALSE and an error if file already exists

PHP Read File - fread()

- The fread() function reads from an open file.

- The first parameter of fread() contains the name of the file to read from and the second parameter specifies the maximum number of bytes to read.

- The following PHP code reads the "webdictionary.txt" file to the end:

- fread($myfile,filesize("webdictionary.txt"));

PHP Close File - fclose()

- The fclose() function is used to close an open file.

- ```php
<?php
$myfile = fopen("webdictionary.txt", "r");
// some code to be executed....
fclose($myfile);
?>
```

PHP Read Single Line - fgets()

- The fgets() function is used to read a single line from a file.
- The example below outputs the first line of the "webdictionary.txt" file:

```php
<?php
$myfile =
fopen("webdictionary.txt", "r") or die("Unable
to open file!");
echo fgets($myfile);
fclose($myfile);
?>
```

PHP Check End-Of-File - feof()

- The feof() function checks if the "end-of-file" (EOF) has been reached.
- The feof() function is useful for looping through data of unknown length.

```php
<?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);
?>
```

PHP Read Single Character - fgetc()

- The fgetc() function is used to read a single character from a file

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

PHP Write to File - fwrite()

- The fwrite() function is used to write to a file.

- The first parameter of fwrite() contains the name of the file to write to and the second parameter is the string to be written.

- The example below writes a couple of names into a new file called "newfile.txt":

```php
<?php
$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");
$txt = "John Doe\n";
fwrite($myfile, $txt);
$txt = "Jane Doe\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

PHP Form Handling
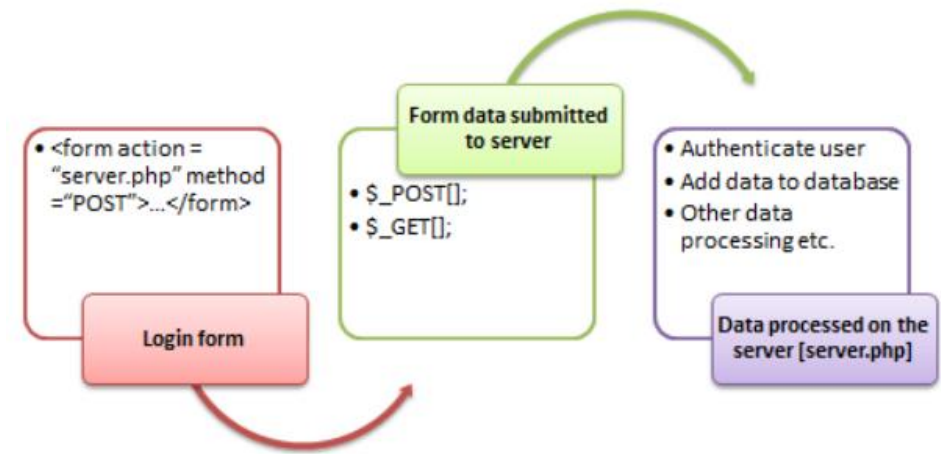
- **What is Form?**

- When you login into a website or into your mail box, you are interacting with a form.

- Forms are used to get input from the user and submit it to the web server for processing.

- The diagram below illustrates the form handling process.

- A form is an HTML tag that contains graphical user interface items such as input box, check boxes radio buttons etc.

- The form is defined using the <form>…</form> tags and GUI items are defined using form elements such as input.

Submitting the form data to the server

- The action attribute of the form specifies the submission URL that processes the data. The method attribute specifies the submission type.

- **PHP POST method**

- This is the built in PHP super global array variable that is used to get values submitted via HTTP POST method.

- The array variable can be accessed from any script in the program; it has a global scope.

- This method is ideal when you do not want to display the form post values in the URL.

- A good example of using post method is when submitting login details to the server.

```
<?php
$_POST['variable_name'];
?>
```

- "$_POST[…]" is the PHP array
- "'variable_name'" is the URL variable name.

PHP GET method

- This is the built in PHP super global array variable that is used to get values submitted via HTTP GET method.

- The array variable can be accessed from any script in the program; it has a global scope.

- This method displays the form values in the URL.

- It's ideal for search engine forms as it allows the users to book mark the results.

```
<?php
$_GET['variable_name'];
?>
```

- "$_GET[…]" is the PHP array
- "'variable_name'" is the URL variable name.

GET vs POST Methods

POST	GET
Values not visible in the URL	Values visible in the URL
Has not limitation of the length of the values since they are submitted via the body of HTTP	Has limitation on the length of the values usually 255 characters. This is because the values are displayed in the URL. Note the upper limit of the characters is dependent on the browser.
Has lower performance compared to Php_GET method due to time spent encapsulation the Php_POST values in the HTTP body	Has high performance compared to POST method dues to the simple nature of appending the values in the URL.
Supports many different data types such as string, numeric, binary etc.	Supports only string data types because the values are displayed in the URL
Results cannot be book marked	Results can be book marked due to the visibility of the values in the URL

FORM SUBMISSION POST METHOD

```html
<form action="registration_form.php" method="POST">
    First name: <input type="text" name="firstname"><br>
    Last name: <input type="text" name="lastname">
    <br>
    <input type="hidden" name="form_submitted" value="1"/>
    <input type="submit" value="Submit">
</form>
```

Submission URL does not show form values

localhost/tuttis/registration_form.php ☆

FORM SUBMISSION GET METHOD

```html
<form action="registration_form.php" method="GET">
    First name: <input type="text" name="firstname"><br>
    Last name: <input type="text" name="lastname">
    <br>
    <input type="hidden" name="form_submitted" value="1"/>
    <input type="submit" value="Submit">
</form>
```

SUBMISSION URL SHOWS FORM VALUES

localhost/tuttis/registration_form.php?firstname=Smith&lastname=Jones&form_submitted=1 ☆

Example

```
Example.html
<html>
<body>
<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
</body>
</html>
```

```
Welcome.php
<html>
<body>
Welcome <?php echo $_POST["name"];
?><br>
Your email address is: <?php echo
$_POST["email"];?>
</body>
</html>
```

Name: [   ]

E-majl: [   ]

[Submit]

Welcome Simanta
Your email address is: sim.kasaju@gmail.com

```html
<html>
<head>
    <title>Registration Form</title>
</head>
<body>
    <?php if (isset($_POST['form_submitted'])): ?>
        <h2>Thank You <?php echo $_POST['firstname']; ?> </h2>
        <p>You have been registered as
        <?php echo $_POST['firstname'] . ' ' . $_POST['lastname']; ?>
        </p>
        <p>Go <a href="/registration_form.php">back</a> to the form</p>
        <?php else: ?>
            <h2>Registration Form</h2>
            <form action="registration_form.php" method="POST">
                 First name:
                <input type="text" name="firstname">
                <br> Last name:
                <input type="text" name="lastname">
            <input type="hidden" name="form_submitted" value="1" />
                <input type="submit" value="Submit">
            </form>
        <?php endif;?>
</body>
</html>
```

Registration Form

First name: [   ]

Last name: [   ]

[Submit]

Thank You simanta

You have been registered as simanta Kasaju

Go back to the form

$_SERVER["PHP_SELF"]

- The $_SERVER["PHP_SELF"] is a super global variable that returns the filename of the currently executing script.

- So, the $_SERVER["PHP_SELF"] sends the submitted form data to the page itself, instead of jumping to a different page. This way, the user will get error messages on the same page as the form.

- **What is the htmlspecialchars() function?**

  The htmlspecialchars() function converts special characters to HTML entities. This means that it will replace HTML characters like < and > with < and >. This prevents attackers from exploiting the code by injecting HTML or Javascript code (Cross-site Scripting attacks) in forms.

Validate Form Data With PHP

- The first thing we will do is to pass all variables through PHP's htmlspecialchars() function.
- When we use the htmlspecialchars() function; then if a user tries to submit the following in a text field:
- <script>location.href('http://www.hacked.com')</script>
- - this would not be executed, because it would be saved as HTML escaped code, like this:
- <script>location.href('http://www.hacked.com')</script>
- The code is now safe to be displayed on a page or inside an e-mail.
- We will also do two more things when the user submits the form:
1. Strip unnecessary characters (extra space, tab, newline) from the user input data (with the PHP trim() function)
2. Remove backslashes (\) from the user input data (with the PHP stripslashes() function)
- The next step is to create a function that will do all the checking for us (which is much more convenient than writing the same code over and over again).
- We will name the function test_input().

```php
<?php
// define variables and set to empty values
$name = $email = $gender = $comment = $website
= "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = test_input($_POST["name"]);
  $email = test_input($_POST["email"]);
  $website = test_input($_POST["website"]);
  $comment = test_input($_POST["comment"]);
  $gender = test_input($_POST["gender"]);
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>
```

```php
<?php
// define variables and set to empty
values
$nameErr = $emailErr = $genderErr =
$websiteErr = "";
$name = $email = $gender = $comment =
$website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST")
{
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
  }

  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
  }

  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);
  }

  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }
}
?>
```

Form Validation in PHP

- An HTML form contains various input fields such as text box, checkbox, radio buttons, submit button, and checklist, etc. These input fields need to be validated, which ensures that the user has entered information in all the required fields and also validates that the information provided by the user is valid and correct.

- There is no guarantee that the information provided by the user is always correct. PHP validates the data at the server-side, which is submitted by HTML form. You need to validate a few things:

- Empty String

- Validate String

- Validate Numbers

- Validate Email

- Validate URL

- Input length

Empty String

The code below checks that the field is not empty. If the user leaves the required field empty, it will show an error message. Put these lines of code to validate the required field.

```php
if (empty ($_POST["name"])) {
$errMsg = "Error! You didn't enter the Name.";
echo $errMsg;  }
else {
$name = $_POST["name"]; }
```

Validate Number

The below code validates that the field will only contain a numeric value. **For example** - Mobile no. If the *Mobile no* field does not receive numeric data from the user, the code will display an error message:

Validate String

The code below checks that the field will contain only alphabets and whitespace, for example - name. If the name field does not receive valid input from the user, then it will show an error message:

```php
$name = $_POST ["Name"];
if (!preg_match ("/^[a-zA-z]*$/", $name) ) {
    $ErrMsg = "Only alphabets and whitespace are allowed.";
        echo $ErrMsg;
} else {
    echo $name;
}
```

```php
$mobileno = $_POST ["Mobile_no"];
if (!preg_match ("/^[0-9]*$/", $mobileno) ){
$ErrMsg = "Only numeric value is allowed.";
echo $ErrMsg;
} else {
echo $mobileno;  }
```

Preg_match: Perform a regular expression match

Validate Email

A valid email must contain @ and . symbols. PHP provides various methods to validate the email address. Here, we will use regular expressions to validate the email address.

The below code validates the email address provided by the user through HTML form. If the field does not contain a valid email address, then the code will display an error message:

```php
$email = $_POST ["Email"];
$pattern = "^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*(\.[a-z]{2,3})$^";
if (!preg_match ($pattern, $email) ){
$ErrMsg = "Email is not valid.";
echo $ErrMsg;  } else {
echo "Your valid email address is: " .$email;  }
```

Input Length Validation

The input length validation restricts the user to provide the value between the specified range, for Example - Mobile Number. A valid mobile number must have 10 digits.

The given code will help you to apply the length validation on user input:

```php
$mobileno = strlen ($_POST ["Mobile"]);
$length = strlen ($mobileno);
if ( $length < 10 && $length > 10) {
$ErrMsg = "Mobile must have 10 digits.";
echo $ErrMsg;  } else {
echo "Your Mobile number is: " .$mobileno;  }
```

Validate URL

The below code validates the URL of website provided by the user via HTML form. If the field does not contain a valid URL, the code will display an error message, i.e., "URL is not valid".

```php
$websiteURL = $_POST["website"];
if(!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
 $websiteErr = "URL is not valid";
echo $websiteErr;  } else {
echo "Website URL is: " .$websiteURL;  }
```

```php
<!DOCTYPE html>
<html>
<head>
<style>
.error {color: #FF0001;}
</style>
</head>
<body>
<?php
// define variables to empty values
$nameErr = $emailErr = $mobilenoErr =
$genderErr = $websiteErr = $agreeErr = "";
$name = $email = $mobileno = $gender = $website
= $agree = "";

//Input fields validation
if ($_SERVER["REQUEST_METHOD"] == "POST") {

//String Validation
    if (empty($_POST["name"])) {
        $nameErr = "Name is required";
    } else {
        $name = input_data($_POST["name"]);
            // check if name only contains
letters and whitespace
            if (!preg_match("/^[a-zA-Z
]*$/",$name)) {
                $nameErr = "Only alphabets and
white space are allowed";
            }

    }
```

```php
//Email Validation
    if (empty($_POST["email"])) {
        $emailErr = "Email is required";
    } else {
        $email = input_data($_POST["email"]);
        // check that the e-mail address is well-formed
        if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
            $emailErr = "Invalid email format";
        }}

//Number Validation
if (empty($_POST["mobileno"])) {
        $mobilenoErr = "Mobile no is required";
    } else {
        $mobileno = input_data($_POST["mobileno"]);
        // check if mobile no is well-formed
        if (!preg_match ("/^[0-9]*$/", $mobileno) ) {
        $mobilenoErr = "Only numeric value is allowed.";
        }
    //check mobile no length should not be less and greator than 10
    if (strlen ($mobileno) != 10) {
        $mobilenoErr = "Mobile no must contain 10 digits.";
        }}
//URL Validation
if (empty($_POST["website"])) {
    $website = "";
    } else {
        $website = input_data($_POST["website"]);
        // check if URL address syntax is valid
        if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
            $websiteErr = "Invalid URL";
        }}
```

```php
//Empty Field Validation
    if (empty ($_POST["gender"])) {
            $genderErr = "Gender is required";
    } else {
            $gender = input_data($_POST["gender"]);
    }

    //Checkbox Validation
    if (!isset($_POST['agree'])){
            $agreeErr = "Accept terms of services before submit.";
    } else {
            $agree = input_data($_POST["agree"]);
    }
}
function input_data($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>
```

```html
<h2>Registration Form</h2>
<span class = "error">* required field </span>
<br><br>
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]); ?>" >
    Name:
    <input type="text" name="name">
    <span class="error">* <?php echo $nameErr; ?> </span>
    <br><br>
    E-mail:
    <input type="text" name="email">
    <span class="error">* <?php echo $emailErr; ?> </span>
    <br><br>
    Mobile No:
    <input type="text" name="mobileno">
    <span class="error">* <?php echo $mobilenoErr; ?> </span>
    <br><br>
    Website:
    <input type="text" name="website">
    <span class="error"><?php echo $websiteErr; ?> </span>
    <br><br>
    Gender:
    <input type="radio" name="gender" value="male"> Male
    <input type="radio" name="gender" value="female"> Female
    <input type="radio" name="gender" value="other"> Other
    <span class="error">* <?php echo $genderErr; ?> </span>
    <br><br>
    Agree to Terms of Service:
    <input type="checkbox" name="agree">
    <span class="error">* <?php echo $agreeErr; ?> </span>
    <br><br>
    <input type="submit" name="submit" value="Submit">
    <br><br>
</form>
```

```php
<?php
    if(isset($_POST['submit'])) {
    if($nameErr == "" && $emailErr == "" && $mobilenoErr == "" &&
$genderErr == "" && $websiteErr == "" && $agreeErr == "") {
        echo "<h3 color = #FF0001> <b>You have sucessfully
registered.</b> </h3>";
        echo "<h2>Your Input:</h2>";
        echo "Name: " .$name;
        echo "<br>";
        echo "Email: " .$email;
        echo "<br>";
        echo "Mobile No: " .$mobileno;
        echo "<br>";
        echo "Website: " .$website;
        echo "<br>";
        echo "Gender: " .$gender;
    } else {
        echo "<h3> <b>You didn't filled up the form correctly.</b>
</h3>";
    }
    }
?>

</body>
</html>
```

Registration Form

* required field

Name: [   ] *

E-mail: [   ] *

Mobile No: [   ] *

Website: [   ]

Gender: ○ Male ○ Female ○ Other *

Agree to Terms of Service: ☐ *

Submit

Registration Form

Name: [   ] * Name is required

E-mail: [   ] * Email is required

Mobile No: [   ] * Mobile no is required

Website: [   ]

Gender: ○ Male ○ Female ○ Other * Gender is required

Agree to Terms of Service: ☐ * Accept terms of services before submit.

[Submit]

You didn't filled up the form correctly.

```html
Login.html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Document</title>
    <link rel="stylesheet" type="text/css" href=Login.css>
</head>
<body>

    <div id="A">
    <h3 id="B">LOGIN</h3>
    <form action="login.php" method="post">

    <label>Username:</label>
    <input type="text" name="username">
   <br><br>
<label>Password:</label>
    <input type="password" name="password">
    <br><br>
<input type="submit" id="btn" value="login">
    </form>
</div>
</body>
</html>
```

```css
Login.css
body{
    background: rgb(4, 250, 197);
}
#B
{
    text-align: center;
}
#A{
    border: solid rgb(0, 0, 0)
5px;
    width: 20%;
    height: 20%;
    border-radius: 5px;
    margin: 100px auto;
    background: white;
    padding: 50px;
}
#btn{
    color:#fff;
    background: #337ab7;
    padding: 15px;
    margin-left: 45%;
}
```

```php
login.php
<?php
$server="localhost";
$user="root";
$password="";
$db="login";
$conn= mysqli_connect($server,$user,$password,$db);
if(isset($_POST['username'])){

    $uname=$_POST['username'];
    $password=$_POST['password'];

    $sql="select * from loggin where
Username='".$uname."'AND Password='".$password."' limit 1";
    $result=mysqli_query($conn,$sql);
    if(mysqli_num_rows($result)==1){
        echo "<h1> You Have Successfully Logged in</h1> ";
        exit();
    }
    else{
        echo "<h1>You Have Entered Incorrect Password</h1>
";
        exit();
    }}
?>
```

When providing wrong password

You Have Entered Incorrect Password

When providing right password

You Have Successfully Logged in

example.HTML

```html
<!DOCTYPE html>
<html>
<head>
    <title>FORM</title>
</head>
<body>

<h2 align="center" style="margin-top: 100px;">FORM</h2>

<form action="insert.php" method="post">
    <table align="center">
        <tr>
            <td>ID :</td>
            <td><input type="text" name="ID" required="required"></td>
        </tr>
        <tr>
            <td>Name :</td>
            <td><input type="text" name="name" required="required"></td>
        </tr>
        <tr>
            <td>Email :</td>
            <td><input type="text" name="email" required="required"></td>
        </tr>
        <tr>
            <td>Phone :</td>
            <td><input type="number" name="phone" required="required"></td>
        </tr>
        <tr>
            <td colspan="2" align="center"><input type="submit"
name="submit"></td>
        </tr>
    </table>
</form>


</body>
</html>
```

Insert.php

```php
<?php
        $ID = $_POST ['ID'];
        $name = $_POST ['name'];      //from index.php(html
form) name
        $email = $_POST ['email'];  //from index.php(html
form) name
        $phone = $_POST ['phone'];//from index.php(html form)
name

        $conn = mysqli_connect('localhost', 'root', '',
'school');

        $query = "INSERT INTO `class` (`ID`, `Name`, `Email`,
`Phone`) VALUES ('$ID','$name','$email','$phone');";

        $run = mysqli_query($conn, $query);

        if ($run == TRUE){
            echo "This is insert into Database table";
        }
        else
            echo "error!";
?>
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<form action="retrive.php" method="post">

<input type="submit" name="viewrecords" value="view records">
    </form>
```

```php
<?php

    $var = mysqli_connect('localhost', 'root', '', 'school');
    function display()
    {
        global $var;
        $query ="SELECT * FROM `class`";
        $run = mysqli_query($var, $query);

        if ($run == TRUE) {
            ?>
            <table border="5" width="50%">
                <?php

            while ( $data = mysqli_fetch_assoc($run)) {
                ?>
                <tr>
                    <td> <?php echo $data['ID'];
                            echo "<br>"; ?>

                    </td>
                    <td> <?php echo $data['Name'];
                            echo "<br>"; ?>

                    </td>
                    <td> <?php echo $data['Email'];
                            echo "<br>"; ?>

                    </td>
                    <td> <?php echo $data['Phone'];
                            echo "<br>"; ?>
                    </td>
                </tr>
                <?php
            }
            ?>
            </table>
            <?php

        } else {
            echo "Error!";
        }
    }
?>
<!DOCTYPE html>
<html>
<head>
    <title>School</title>
</head>
<body>
        <?php
            display();
        ?>
</body>
</html>
```

Retrive.php

FORM

ID :

Name :

Email :

Phone :

Submit
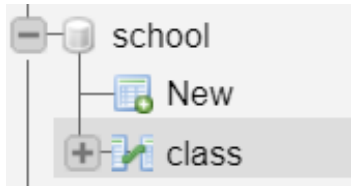
FORM

ID : 1

Name : simanta

Email : sim.kasaju@gmail.com

Phone : 12345678

Submit

Records has been inserted into Database table

view records

- school
  - New
  - class

←T→	ID	Name	Email	Phone

←T→				▼	ID	Name	Email	Phone
☐	🖊 Edit	Copy	⊖ Delete		1	simanta	sim.kasaju@gmail.com	12345678

FORM

ID : 2

Name : prashant

Email : prashant.kasaju@gmail.com

Phone : 01234567

Submit

Records has been inserted into Database table

view records

←T→				▼	ID	Name	Email	Phone
☐	🖊 Edit	Copy	⊖ Delete		1	simanta	sim.kasaju@gmail.com	12345678
☐	🖊 Edit	Copy	⊖ Delete		2	prashant	prashant.kasaju@gmai	1234567

Similarly multiple data can be inserted

				ID	Name	Email	Phone
☐	🖊 Edit	⊞ Copy	⊖ Delete	1	simanta	sim.kasaju@gmail.com	12345678
☐	🖊 Edit	⊞ Copy	⊖ Delete	2	prashant	prashant.kasaju@gmai	1234567
☐	🖊 Edit	⊞ Copy	⊖ Delete	3	Salina	Sali.kasaju@gmail.co	11223344
☐	🖊 Edit	⊞ Copy	⊖ Delete	4	Shashi	Shashi.kasaju@gmail.	223344
☐	🖊 Edit	⊞ Copy	⊖ Delete	5	Puran	Puran.kasaju@gmail.c	98451

When click view records

1	simanta	sim.kasaju@gmail.com	12345678
2	prashant	prashant.kasaju@gmai	1234567
3	Salina	Sali.kasaju@gmail.co	11223344
4	Shashi	Shashi.kasaju@gmail.	223344
5	Puran	Puran.kasaju@gmail.c	98451

How a domain Name is translated to an IP address(read yourself)

- DNS (Domain Name System) is what translate domain name (for eg. www.google.com) into an IP address that our browser can use (for eg. 173.194.35.148).

- Before the page is loaded, the DNS must be resolved so the browser can establish a TCP connection to make the HTTP request.

- The DNS Resolution process starts when the user types a URL address on the browser and hits Enter.

- On the Internet, many communications programs deal only with IP addresses, yet allow their users to specify machines in terms of their host names (or alias host names).

- Or a program which already knows the IP address must determine the domain name for the network to which the machine is connected. Such programs must somehow convert the host names into IP addresses (or vice versa) behind the scenes.

- How do they achieve this translation between IP addresses and host names?

- The mapping of host names to IP addresses is handled through a service called Domain Name Service (DNS).

- Rather than require individual machines, applications, or users to keep up with the constant changes in host names and IP addresses, a series of special DNS servers across the world (known as "name servers") keep track of the name/address information for all the computers on the Internet.

- Applications that need to determine an IP address from a host name (or vice versa) contact the local "name server" to supply this information.

- For instance, if you use a web browser to check out the site "web.mit.edu", the program actually first contacts your local DNS machine to obtain the IP-address that matches the host name you provided; then the program uses that IP address to complete your request. DNS is used much more frequently than is usually supposed: virtually every activity that moves information across the network (getting web documents, transferring files, sending or receiving electronic mail) relies on DNS.

Authentication by IP address

In some rare instances, we may wish to limit access to a certain page or pages to certain IP addresses. It may be because we have an internal network that has a publicly viewable website. We may wish to have certain pages be viewable only by certain machines on internal network.

Or it may be that we have our own remote website, and we wish to restrict access to a certain page or pages so that only we, from a static IP address, can access those pages.

These methods work by determining the IP addresses of the user trying to view the pages and checking it against a set value. If the IP address matches, then the page can be viewed.

This method could be fooled by IP spoofing or other hacks, and should not be used to protect sensitive information, such as credit card numbers, proprietary information, or the like/ it is simple method that can protect pages from the majority of users surfing the net.

When using this script, we should be aware that our computer may have more than one IP address and that the one your browser is reporting to the script may not be the same one we are attempting to verify

Er. Simanta Kasaju

1

against. This is especially true when the browser and web server resides on the same machine, as we will probably encounter when testing the script.

For example, computers have a default local IP 127.0.0.1, otherwise known as "localhost". When we start up the web server and type in "http://localhost", our browser is telling the script that we are at IP address of 127.0.0.1. If we type in the IP address as 192.168.0.1, our browser reports that we are at 192.168.0.1, but the script only accepts the IP of 127.0.0.1. If we run into problems, try echoing the value of $REMOTE_ADDR to screen to see which IP.

Example: If authorization PHP

```php
<?php
$accept = array("127", "0", "0", "1");
$remote = explode("." $REMOTE_ADDR);
$match = 1;
for($i=0; $i<sizeof($accept); $i++){
if($remote[$i] != $accept[$i]){
$match = 0;
}}
if($match){
echo"<h2>Access Granted</h2>";
}
else{
echo"<h2>Access Forbidden</h2>";
}
?>
```

End of chapter 5