INT316 Modern Web Applications

Unit IV

Contents

- Introduction to PHP
- Expressions and Control flow
- PHP functions and objects
- Arrays
- Form handling
- Introduction to MySQL
- **Accessing MySQL using PHP**
- Cookies, Sessions and Authentication

Querying a MySQL Database with PHP

- The Process
  - The process of using MySQL with PHP is as follows:
    1. Connect to MySQL and select the database to use.
    2. Prepare a query string.
    3. Perform the query.
    4. Retrieve the results and output them to a web page.
    5. Repeat steps 2 to 4 until all desired data has been retrieved.
    6. Disconnect from MySQL.

- Creating a Login File
  - Most websites developed with PHP contain multiple program files that will require access to MySQL and will thus need the login and password details.
  - Therefore, it's sensible to create a single file to store these and then include that file wherever it's needed
  - *login.php*.

```
<?php // login.php
        $hn = 'localhost';
        $db = 'publications';
        $un = 'username';
        $pw = 'password';
?>
```

- Connect to database
  - Create $conn object by calling a new instance of the mysqli method, passing all the values retrieved from the *login.php* file.
  - Error checking is achieved by referencing the $conn->connect_error property.

```
<?php
    require_once 'login.php';
    $conn = new mysqli($hn, $un, $pw, $db);
    if ($conn->connect_error) die("Fatal Error");
?>
```

- Building and executing a query
  - Sending a query to MySQL from PHP is as simple as including the relevant SQL in the query method of a connection object.
  - query() method in $conn object is used to execute the query

```
<?php
        $query = "SELECT * FROM classics";
        $result = $conn->query($query);
        if (!$result) die("Fatal Error");
?>
```

- Here the variable $query is assigned a string containing the query
- $query is then passed to the query method of the $conn object query method
- query() method of $conn object returns a result which is placed in the object $result.
- If $result is FALSE, there was a problem and the error property of the connection object will contain the details, so the die function is called to display that error.

- Fetching a result
  - The result object, $result, contains the result of the query.
  - To extract data from $result object, fetch_assoc method is used

```
$rows = $result->num_rows;
for ($j = 0 ; $j < $rows ; ++$j)  {
        $result->data_seek($j);
        echo 'Author: ' .htmlspecialchars($result->fetch_assoc()['author']) .'<br>';
        $result->data_seek($j);
        echo 'Title: ' .htmlspecialchars($result->fetch_assoc()['title']) .'<br>';
        $result->data_seek($j);
        echo 'Category: '.htmlspecialchars($result->fetch_assoc()['category']).'<br>';
        $result->data_seek($j);
        echo 'Year: ' .htmlspecialchars($result->fetch_assoc()['year']) .'<br>';
        $result->data_seek($j);
        echo 'ISBN: ' .htmlspecialchars($result->fetch_assoc()['isbn']) .'<br><br>';
}
```

Another method to fetch data from $result:

```
if ($result->num_rows > 0) {
  // output data of each row
  while($row = $result->fetch_assoc()) {
    echo "id: " . $row[“author"]. " - Name: " . $row[“title"]. " " . $row[“year"]. "<br>";
  }
} else {
  echo "0 results";
}
```

Another method to fetch data from $result:

MYSQLI_NUM - Numeric array
MYSQLI_ASSOC - Associative array
MYSQLI_BOTH - Associative and numeric array.

```
$rows = $result->num_rows;
for ($j = 0 ; $j < $rows ; ++$j)
{
        $row = $result->fetch_array(MYSQLI_ASSOC);
        echo 'Author: ' . htmlspecialchars($row['author']) . '<br>';
        echo 'Title: ' . htmlspecialchars($row['title']) . '<br>';
        echo 'Category: ' . htmlspecialchars($row['category']) . '<br>';
        echo 'Year: ' . htmlspecialchars($row['year']) . '<br>';
        echo 'ISBN: ' . htmlspecialchars($row['isbn']) . '<br><br>';
}
```

- Closing a connection
  - Closing of connection object and result object is important

```
$result->close();
$conn->close();
```

Some Important Steps

- To check the method type of form submission

  $_SERVER["REQUEST_METHOD"] == "POST“

isset() function

- The isset() function is an inbuilt function in PHP which checks whether a variable is set and is not NULL.
- This function also checks if a declared variable, array or array key has null value, if it does, isset() returns false, it returns true in all other possible cases.

Example usage:

isset($_POST[“form_field_name”])

isset($_POST[“cgpa”])

```
<?php
 // PHP program to illustrate isset() function
$num = '0';

if( isset( $num ) ) {
    print_r(" $num is set <br>");
}

// Declare an empty array
$array = array();

// Use isset function
echo isset($array['geeks']) ?
            'array is set.' :  'array is not set.';
?>
```

empty() function

- The empty() function is a language construct to determine whether the given variable is empty or NULL.
- The !empty() function is the negation or complement of empty() function.
- The empty() function is considerably equal to !isset() function and !empty() function is equal to isset() function.

```
<?php
// PHP program to illustrate empty() function

$temp = 0;

// It returns true because $temp is empty
if (empty($temp)) {
    echo $temp . ' is considered empty';
}

// It returns true since $new exist
$new = 1;
if (!empty($new)) {
    echo $new . ' is considered set';
}
?>
```

Insert Data

- To insert values into table query() method of conn object can be used.

```
$conn = new mysqli($hn, $un, $pw, $db);
$query = "INSERT INTO cats VALUES(101, 'Lion', 'Leo', 4)";
$result = $conn->query($query);
if (!$result) die ("Database access failed");
```

- Prepared Statement:
  - A prepared statement is a feature used to execute the same (or similar) SQL statements repeatedly with high efficiency.
  - Prepared statements basically work like this:
    - Prepare: An SQL statement template is created and sent to the database. Certain values are left unspecified, called parameters (labeled "?"). Example: INSERT INTO MyGuests VALUES(?, ?, ?)
    - The database parses, compiles, and performs query optimization on the SQL statement template, and stores the result without executing it
    - Execute: At a later time, the application binds the values to the parameters, and the database executes the statement. The application may execute the statement as many times as it wants with different values

- Example:

```
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
}

$stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname,
                email) VALUES (?, ?, ?)");
$stmt->bind_param("sss", $firstname, $lastname, $email);

$firstname = "John";
$lastname = "Doe";
$email = "john@example.com";
$stmt->execute();
```

i - integer
d - double
s - string

Updating Data

- Changing data that you have already inserted is also quite simple

```
<?php
    require_once 'login.php';
    $conn = new mysqli($hn, $un, $pw, $db);
    if ($conn->connect_error) die("Fatal Error");
            $query = "UPDATE cats SET name='Charlie' WHERE
                    name='Charly'";
    $result = $conn->query($query);
    if (!$result) die ("Database access failed");
?>
```

Deleting Data

```
<?php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
if ($conn->connect_error) die("Fatal Error");
$query = "DELETE FROM cats WHERE name='Growler'";
$result = $conn->query($query);
if (!$result) die ("Database access failed");
?>
```