

Learning Objectives

After studying this material, participants are expected to be able to:

- Understanding the basic concept of forms in PHP.
- Managing input data from an HTML form using PHP.
- Perform input data validation for security and accuracy.
- Saving data from the form to the database.
- Using GET and POST methods securely.
- Implementing basic security measures such as input validation and sanitization.

2. Requirements

Before starting this lesson, participants are advised to have a basic understanding of:

- HTML dan CSS
- Basics of PHP
- Basic concepts of MySQL database

3. Hardware Specifications

3.1 Minimum Requirements

- **Prosesor:** Intel Core i3 or higher
- **RAM:** 4 GB
- **Penyimpanan:** 10 GB of free space
- **Sistem Operasi:** Windows 7 / macOS Sierra / Linux Ubuntu 16.04

3.2 Recommendation Requirements

- **Prosesor:** Intel Core i5 or higher
- **RAM:** 8 GB or more
- **Penyimpanan:** SSD with 20 GB of free space
- **Sistem Operasi:** Windows 10 / macOS Big Sur / Linux Ubuntu 20.04

4. Required Software

- **Web Server:** XAMPP, WAMP, or LAMP
- **Database Management System:** MySQL or MariaDB
- **Editor Kode:** VS Code, Sublime Text, or PHPStorm
- **Browser:** Google Chrome or Mozilla Firefox
- **PHP:** Version 7.4 or higher

Database connection material Review

The following code is used to connect PHP to a MySQL database using **MySQLi (MySQL Improved Extension)**. Let's break down each part in detail to make it easier for beginners to understand.

1. What is MySQLi?

MySQLi is a PHP extension used to interact with MySQL databases. MySQLi supports two connection methods:

- **Procedural** → Using `mysqli_connect()`
- **OOP (Object-Oriented Programming)** → Using new `mysqli()`

The code we are discussing uses the **OOP** method.

2. Code Explanation

```
<?php
// Database connection
$localhost = "localhost"; // Database server address (usually "localhost")
$username = "root";       // Database username (default is "root" in XAMPP)
$password = "";           // Database password (default is empty in XAMPP)
$db        = "test_db";   // Name of the database to be used
// Creating a connection
$conn = new mysqli($localhost, $username, $password, $db);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
// Closing the connection
$conn->close();
?>
```

3. Explanation of Each Line

Part 1: Initializing Variables

- **\$localhost** → MySQL server address. If the server is on your own computer, use "localhost".
- **\$username** → MySQL username. The default for XAMPP is "root".
- **\$password** → MySQL password. The default for XAMPP is "" (empty).
- **\$db** → Name of the database to be used.

⚠ Note:

Make sure the database **test_db** exists in MySQL. If not, create it using the following SQL command:

```
CREATE DATABASE test_db;
```

Part 2: Creating the Database Connection

```
$conn = new mysqli($localhost, $username, $password, $db);
```

- **new mysqli(\$localhost, \$username, \$password, \$db);**
→ Creates a connection object using **MySQLi OOP**.
 - If successful, the connection is established without any error messages.
-

Part 3: Checking the Connection

```
if ($conn->connect_error) { die("Connection failed: " . $conn->connect_error);}
```

- **\$conn->connect_error** → Checks if there is an error in the connection.
- If there is an error, the program stops (die()) and displays the message "**Connection failed: [error]**".

⚠ Common Errors:

- **Wrong database name** → Unknown database 'test_db'
 - **Incorrect username/password** → Access denied for user 'root'@'localhost'
 - **MySQL is not running** → Can't connect to MySQL server
-

Part 4: Closing the Connection

`$conn->close();`

- **Closes the connection to the database after it is used.**
 - This is important to conserve server resources.
-

Inserta Data Material Review

Previously, we have learned how to connect our application to the database. In this session, we will learn how to add data to the database.

- **The first step is to create a form to store data through a form**

```
<html>
  <body>
    <form action="insert.php" method="post">
      NIP :<input type="text" name="NIP" />
      Initial name: <input type="text" name="initial_name" />
      Last name: <input type="text" name="last_name" />
      Age: <input type="text" name="age" />
      <input type="submit" />
    </form>
  </body>
</html>
```

- **The second step is to save the data**

The INSERT INTO command is used to add a new record to a database table. The INSERT INTO statement can be written in 2 forms: INSERT INTO table_name VALUES (value1, value2, value3,...) or INSERT INTO table_name (column1, column2, column3,...). VALUES (value1, value2, value3,...). Still using the test_db database and the biodata table, we are trying to design a program to input data using the INSERT INTO instruction.

```
<?php
$servername = "localhost";
$username   = "root"; // Replace with your database username
$password   = ""; // Replace with your database password
$dbname     = "test_db"; // Replace with the name of the database used

// Create a connection to the database
$conn = new mysqli($servername, $username, $password, $dbname);

// Check the connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
```

```
// Data to be entered
$name = "John Doe";
$age = 25;
$address = "Jl. Contoh No. 123";

// Query to enter data
$sql = "INSERT INTO biodata (name, age, address) VALUES ('$name', $age, '$address')";

// Execute the query and check the result
if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

// Close the connection
$conn->close();
?>
```

○ **Explanation**

- Determining the data to be input.
 - Writing the INSERT INTO command to add data to the biodata table.
 - Execute the command with \$conn->query(\$sql).
 - Display a success message if successful, or an error message if it fails.
-

Update Data Material Review

The **UPDATE** command in SQL is used to update existing data in a table. The basic format of the **UPDATE** command is:

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

Note:

Use **WHERE** to update only certain rows.

If **WHERE** is not used, all data in the table will be updated.

```
<?php
$servername = "localhost";
$username   = "root"; // Replace with your database username
$password   = "";     // Replace with your database password
$dbname     = "test_db"; // Replace with the name of the database used
```

```
// Create a connection to the database
$conn = new mysqli($servername, $username, $password, $dbname);

// Check the connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Data to be entered
$id = 1;
$name = "John Doe";
$age = 25;
$address = "Jl. Contoh No. 123";

// Query to enter data
$sql = "UPDATE biodata SET name='$new_name', age=$new_age, address='$new_address' WHERE id=$id";

// Execute the query and check the result
if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

// Close the connection
$conn->close();
?>
```

Delete Data Material Review

The DELETE command is used to delete one or more rows in a table in MySQL. The basic format of the DELETE command is:

```
DELETE FROM table_name WHERE condition;
```

Note:

Use **DELETE ... WHERE** to delete specific data.

Be careful when using **DELETE** without **WHERE**, because **it will delete all data in the table**.

Use Prepared Statements to prevent SQL Injection.

```
<?php
$servername = "localhost";
$username   = "root"; // Replace with your database username
$password   = "";     // Replace with your database password
$dbname     = "test_db"; // Replace with the name of the database used
```

```
// Create a connection to the database
$conn = new mysqli($servername, $username, $password, $dbname);

// Check the connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$id = 1;

// Query to enter data
$sql = "DELETE FROM biodata WHERE id=$id";

// Execute the query and check the result
if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

// Close the connection
$conn->close();
?>
```
