# **Lecture:** MySQLi_Connect in PHP (Part 2)

- **Teacher:** Sir Arsalan Shah
- **Typist:** Yousuf Naveed

In this second part of the lecture, we'll continue our exploration of MySQLi_Connect in PHP, focusing on form value insertion into a database using Bootstrap as a boilerplate.

## Connection Setup

Before working with the database, it's essential to set up the connection. We'll use the following code to connect to our MySQL database:

```php
<?php
$server = "localhost";
$username = "root";
$pass = "";
$database = "dbname";
$insert = false;
$con = mysqli_connect($server, $username, $pass, $database);

// Method #1: Check if the connection is successful.
if (!$con) {
    echo "Database connection failed";
    // You can also use an alert here.
}

// Method #2: Using Die Method
if (!$con) {
    die("Connection failed: " . mysqli_connect_error());
}
?>
```

## Database and Table Creation

To prepare the database for data insertion, follow these SQL instructions:

```sql
-- TO CREATE DATABASE
CREATE DATABASE UserAuthentication;

-- TO USE DATABASE
USE UserAuthentication;

-- TO CREATE TABLE
CREATE TABLE INFO (
ID INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
Username VARCHAR(250),
Password INT
);

-- TO DISPLAY THE TABLE
SELECT * FROM INFO;
```

## Form Setup

We'll use a form to insert values into the database. You can either create a separate PHP file for this or include the PHP code at the beginning of your HTML document using **`<?php ?>`** tags.

```html
<form method="post" action="filename.php">
    <!-- Input for username -->
    <input type="text" name="ua" placeholder="Username">

    <!-- Input for password -->
    <input type="password" name="pa" placeholder="Password">

    <!-- Submit button -->
    <input type="submit" value="Submit">
</form>
```

In the form, we've set the method to "post," and the action attribute specifies the PHP file where we'll process the form data. Each input element has a `name` attribute that we'll use to access the submitted values.

## PHP Code to Handle Form Submission

Here's the PHP code that processes the form data and inserts it into the database:

```
    if (isset($_POST["ua"])) {
        $uservalue = $_POST["ua"];
        $passvalue = $_POST["pa"];

        $sql = "INSERT INTO info(username, password) VALUES
('$uservalue', '$passvalue')";

        $res = mysqli_query($con, $sql);

        // If the query is successful, display an alert.
        if ($res) {
            echo "(Bootstrap Alert Code)";
            $insert = true;
        }
    }
```

In this code, we check if the "ua" (username) value is set in the `$_POST` array. If it is, we retrieve both the username and password values. We then construct an SQL query to insert these values into the "info" table of the database.

If the query is successful, we display a Bootstrap alert. You can select an alert from Bootstrap and replace all double inverted commas with single inverted commas before pasting it into the `echo`.

- Don't forget to change the type of your submit button to "submit" within your HTML form.

# Conclusion

In this part of the lecture, we've learned how to set up a connection to a MySQL database, create a form to collect user data, and insert that data into the database. Understanding these concepts is fundamental when building dynamic web applications and handling user data securely.