PHP PDO: Day 2 & Day 3 - Inserting, Fetching, and Functions

This document covers the tasks for Day 2 and Day 3 related to PHP PDO inserting and fetching data, using functions for modularity, and understanding different fetching methods.

# Database Connection Using PHP PDO

<?php  
 // Database configuration  
 $dsn = "mysql:host=localhost;dbname=portfolio\_website"; // Data Source Name  
 $username = "root"; // Database username  
 $password = ""; // Database password  
  
 try {  
 // Create a new PDO instance  
 $pdo = new PDO($dsn, $username, $password);  
   
 // Set the PDO error mode to exception  
 $pdo->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);  
   
 echo "Connected successfully!";  
 } catch (PDOException $e) {  
 // If there is an error, show the message  
 echo "Connection failed: " . $e->getMessage();  
 }  
 ?>

Explanation:  
- \*\*$dsn\*\*: The Data Source Name specifies the connection details for the database, including the database type (mysql), the hostname (localhost), and the database name (portfolio\_website).  
- \*\*$username\*\* and \*\*$password\*\*: Credentials used to access the database. In this case, the default root user with an empty password.  
- \*\*$pdo\*\*: A new PDO instance is created to represent the connection to the database.  
- \*\*Error Handling\*\*: The setAttribute() method is used to set ERRMODE\_EXCEPTION to ensure any SQL errors throw exceptions for easy debugging.  
- \*\*try-catch Block\*\*: Catches exceptions and displays the error message if the connection fails.

# Navbar Section: Creating, Inserting, and Fetching Data

## 1. Creating the Database Table for Navbar

CREATE TABLE navbar (  
 id INT AUTO\_INCREMENT PRIMARY KEY,  
 title VARCHAR(255) NOT NULL,  
 url VARCHAR(255) NOT NULL  
 );

Explanation:  
- \*\*id\*\*: This is the primary key that auto-increments for each new entry in the navbar.  
- \*\*title\*\*: The title or label that will appear for each navigation link (e.g., 'Home', 'About', 'Contact').  
- \*\*url\*\*: The URL that each navigation item will link to.

## 2. Inserting Data into the Navbar Table

<?php  
 // Insert data into the navbar table  
 $sql = "INSERT INTO navbar (title, url) VALUES (:title, :url)";  
 $stmt = $pdo->prepare($sql);  
 $stmt->execute([  
 'title' => 'Home',   
 'url' => '/'  
 ]);  
  
 $stmt->execute([  
 'title' => 'About Us',   
 'url' => '/about'  
 ]);  
  
 $stmt->execute([  
 'title' => 'Contact',   
 'url' => '/contact'  
 ]);  
  
 echo "Navbar items inserted successfully!";  
 ?>

Explanation:  
- \*\*SQL Query\*\*: We're using an \*\*INSERT INTO\*\* statement to add data into the `navbar` table.  
- \*\*Prepared Statements\*\*: The `:title` and `:url` placeholders allow us to securely insert the values into the table without risking SQL injection.  
- \*\*Multiple Inserts\*\*: Each call to `execute()` inserts a new navbar item (e.g., 'Home', 'About Us', 'Contact').

## 3. Fetching Navbar Data from the Database

<?php  
 // Fetch data from the navbar table  
 $sql = "SELECT \* FROM navbar";  
 $stmt = $pdo->query($sql);  
 $navbarLinks = $stmt->fetchAll(PDO::FETCH\_ASSOC);  
  
 // Display the navbar links  
 echo "<ul class='navbar'>";  
 foreach ($navbarLinks as $link) {  
 echo "<li><a href='" . $link['url'] . "'>" . $link['title'] . "</a></li>";  
 }  
 echo "</ul>";  
 ?>

Explanation:  
- \*\*SQL Query\*\*: We're using a \*\*SELECT\*\* statement to retrieve all rows from the `navbar` table.  
- \*\*fetchAll(PDO::FETCH\_ASSOC)\*\*: This fetches all rows as an associative array where the column names are the keys.  
- \*\*HTML Output\*\*: The `foreach` loop iterates through each row and outputs the navbar items as an unordered list (`<ul>`).

# Hero Section: Creating, Inserting, and Fetching Data

## 1. Creating the Database Table for Hero Section

CREATE TABLE hero\_section (  
 id INT AUTO\_INCREMENT PRIMARY KEY,  
 title VARCHAR(255) NOT NULL,  
 subtitle VARCHAR(255) NOT NULL,  
 image\_url VARCHAR(255) NOT NULL  
 );

Explanation:  
- \*\*id\*\*: This is the primary key that auto-increments for each new entry in the hero section.  
- \*\*title\*\*: The main title that will appear in the hero section (e.g., 'Welcome to Our Website').  
- \*\*subtitle\*\*: The subtext or description that will appear under the title (e.g., 'We offer professional services').  
- \*\*image\_url\*\*: The URL of the image that will be displayed in the hero section.

## 2. Inserting Data into the Hero Section Table

<?php  
 // Insert data into the hero section table  
 $sql = "INSERT INTO hero\_section (title, subtitle, image\_url) VALUES (:title, :subtitle, :image\_url)";  
 $stmt = $pdo->prepare($sql);  
 $stmt->execute([  
 'title' => 'Welcome to Our Website',   
 'subtitle' => 'We offer professional services tailored to your needs.',  
 'image\_url' => '/images/hero.jpg'  
 ]);  
  
 echo "Hero section data inserted successfully!";  
 ?>

Explanation:  
- \*\*SQL Query\*\*: We're using an \*\*INSERT INTO\*\* statement to add data into the `hero\_section` table.  
- \*\*Prepared Statements\*\*: The placeholders `:title`, `:subtitle`, and `:image\_url` are used to insert the values securely, preventing SQL injection.  
- \*\*Image URL\*\*: The URL points to the image that will be displayed in the hero section.

## 3. Fetching Hero Section Data from the Database

<?php  
 // Fetch data from the hero\_section table  
 $sql = "SELECT \* FROM hero\_section";  
 $stmt = $pdo->query($sql);  
 $hero = $stmt->fetch(PDO::FETCH\_ASSOC);  
  
 // Display the hero section  
 echo "<div class='hero'>";  
 echo "<h1>" . $hero['title'] . "</h1>";  
 echo "<p>" . $hero['subtitle'] . "</p>";  
 echo "<img src='" . $hero['image\_url'] . "' alt='Hero Image'>";  
 echo "</div>";  
 ?>

Explanation:  
- \*\*SQL Query\*\*: We're using a \*\*SELECT\*\* statement to retrieve the hero section data from the database.  
- \*\*fetch(PDO::FETCH\_ASSOC)\*\*: This fetches a single row as an associative array.  
- \*\*HTML Output\*\*: The retrieved data (title, subtitle, and image URL) is used to dynamically populate the hero section in the HTML.

# Services Section: Creating, Inserting, and Fetching Data

## 1. Creating the Database Table for Services Section

CREATE TABLE services (  
 id INT AUTO\_INCREMENT PRIMARY KEY,  
 service\_name VARCHAR(255) NOT NULL,  
 description TEXT NOT NULL,  
 image\_url VARCHAR(255) NOT NULL  
 );

Explanation:  
- \*\*id\*\*: This is the primary key that auto-increments for each new entry in the services section.  
- \*\*service\_name\*\*: The name of the service (e.g., 'Web Development').  
- \*\*description\*\*: A description of the service being offered.  
- \*\*image\_url\*\*: The URL of the image representing the service.

## 2. Inserting Data into the Services Table

<?php  
 // Insert data into the services table  
 $sql = "INSERT INTO services (service\_name, description, image\_url) VALUES (:service\_name, :description, :image\_url)";  
 $stmt = $pdo->prepare($sql);  
 $stmt->execute([  
 'service\_name' => 'Web Development',  
 'description' => 'We provide custom web development solutions tailored to your business needs.',  
 'image\_url' => '/images/web-development.jpg'  
 ]);  
  
 echo "Service data inserted successfully!";  
 ?>

Explanation:  
- \*\*SQL Query\*\*: We're using an \*\*INSERT INTO\*\* statement to add data into the `services` table.  
- \*\*Prepared Statements\*\*: The placeholders `:service\_name`, `:description`, and `:image\_url` are used to insert the values securely.  
- \*\*Image URL\*\*: The URL points to the image representing the service.

## 3. Fetching Services Data from the Database

<?php  
 // Fetch data from the services table  
 $sql = "SELECT \* FROM services";  
 $stmt = $pdo->query($sql);  
 $services = $stmt->fetchAll(PDO::FETCH\_ASSOC);  
  
 // Display the services  
 foreach ($services as $service) {  
 echo "<div class='service'>";  
 echo "<img src='" . $service['image\_url'] . "' alt='Service Image'>";  
 echo "<h2>" . $service['service\_name'] . "</h2>";  
 echo "<p>" . $service['description'] . "</p>";  
 echo "</div>";  
 }  
 ?>

Explanation:  
- \*\*SQL Query\*\*: We're using a \*\*SELECT\*\* statement to retrieve the services data from the database.  
- \*\*fetchAll(PDO::FETCH\_ASSOC)\*\*: This fetches all rows as an associative array.  
- \*\*HTML Output\*\*: The retrieved data (service name, description, and image URL) is used to dynamically populate the services section in the HTML.

# PHP PDO Fetch Methods

Here are the different fetch modes in PHP PDO that can be used to retrieve data from the database:

1. \*\*PDO::FETCH\_ASSOC\*\* - Fetches data as an associative array, where column names are the keys.  
Example:  
<?php  
$stmt = $pdo->query('SELECT \* FROM users');  
$users = $stmt->fetchAll(PDO::FETCH\_ASSOC);  
foreach ($users as $user) {  
 echo $user['name'];  
}  
?>

2. \*\*PDO::FETCH\_OBJ\*\* - Fetches data as an object, where columns are accessed as properties.  
Example:  
<?php  
$stmt = $pdo->query('SELECT \* FROM users');  
while ($user = $stmt->fetch(PDO::FETCH\_OBJ)) {  
 echo $user->name;  
}  
?>

3. \*\*PDO::FETCH\_CLASS\*\* - Maps the result set to an instance of a class.  
Example:  
class User {  
 public $name;  
 public $email;  
}  
$stmt = $pdo->query('SELECT name, email FROM users');  
$stmt->setFetchMode(PDO::FETCH\_CLASS, 'User');  
while ($user = $stmt->fetch()) {  
 echo $user->name;  
}  
?>

# Working with Functions in PHP

To keep the code clean, modular, and reusable, we will create a \*\*functions.php\*\* file to handle the insertion and fetching of data for all sections (Navbar, Hero, Services). This file will contain functions that can be called whenever we need to interact with the database. This approach separates the business logic from the presentation layer (HTML).

## Creating Functions for Navbar, Hero, and Services

1. \*\*Insert and Fetch for Navbar\*\*  
   
 Create a file called \*\*functions.php\*\* and add the following code for inserting and fetching data for the navbar:  
   
 ```php  
 // Function to insert data into the navbar  
 function insertNavbar($pdo, $title, $url) {  
 $sql = "INSERT INTO navbar (title, url) VALUES (:title, :url)";  
 $stmt = $pdo->prepare($sql);  
 $stmt->execute([  
 'title' => $title,  
 'url' => $url  
 ]);  
 }  
  
 // Function to fetch navbar data  
 function fetchNavbar($pdo) {  
 $sql = "SELECT \* FROM navbar";  
 $stmt = $pdo->query($sql);  
 return $stmt->fetchAll(PDO::FETCH\_ASSOC);  
 }  
 ```  
   
 To display the navbar in your HTML, you can call the fetch function and embed it in the HTML as follows:  
   
 ```php  
 $navbarLinks = fetchNavbar($pdo);  
 echo "<ul class='navbar'>";  
 foreach ($navbarLinks as $link) {  
 echo "<li><a href='" . $link['url'] . "'>" . $link['title'] . "</a></li>";  
 }  
 echo "</ul>";  
 ```

2. \*\*Insert and Fetch for Hero Section\*\*  
   
 Add the following functions to handle inserting and fetching data for the hero section:  
   
 ```php  
 // Function to insert data into the hero section  
 function insertHero($pdo, $title, $subtitle, $image\_url) {  
 $sql = "INSERT INTO hero\_section (title, subtitle, image\_url) VALUES (:title, :subtitle, :image\_url)";  
 $stmt = $pdo->prepare($sql);  
 $stmt->execute([  
 'title' => $title,  
 'subtitle' => $subtitle,  
 'image\_url' => $image\_url  
 ]);  
 }  
  
 // Function to fetch hero section data  
 function fetchHero($pdo) {  
 $sql = "SELECT \* FROM hero\_section";  
 $stmt = $pdo->query($sql);  
 return $stmt->fetch(PDO::FETCH\_ASSOC);  
 }  
 ```  
   
 To display the hero section in your HTML, you can call the fetch function and embed it in the HTML as follows:  
   
 ```php  
 $hero = fetchHero($pdo);  
 echo "<div class='hero'>";  
 echo "<h1>" . $hero['title'] . "</h1>";  
 echo "<p>" . $hero['subtitle'] . "</p>";  
 echo "<img src='" . $hero['image\_url'] . "' alt='Hero Image'>";  
 echo "</div>";  
 ```

3. \*\*Insert and Fetch for Services\*\*  
   
 Add the following functions to handle inserting and fetching data for the services section:  
   
 ```php  
 // Function to insert data into the services section  
 function insertService($pdo, $service\_name, $description, $image\_url) {  
 $sql = "INSERT INTO services (service\_name, description, image\_url) VALUES (:service\_name, :description, :image\_url)";  
 $stmt = $pdo->prepare($sql);  
 $stmt->execute([  
 'service\_name' => $service\_name,  
 'description' => $description,  
 'image\_url' => $image\_url  
 ]);  
 }  
  
 // Function to fetch services data  
 function fetchServices($pdo) {  
 $sql = "SELECT \* FROM services";  
 $stmt = $pdo->query($sql);  
 return $stmt->fetchAll(PDO::FETCH\_ASSOC);  
 }  
 ```  
   
 To display the services section in your HTML, you can call the fetch function and embed it in the HTML as follows:  
   
 ```php  
 $services = fetchServices($pdo);  
 foreach ($services as $service) {  
 echo "<div class='service'>";  
 echo "<img src='" . $service['image\_url'] . "' alt='Service Image'>";  
 echo "<h2>" . $service['service\_name'] . "</h2>";  
 echo "<p>" . $service['description'] . "</p>";  
 echo "</div>";  
 }  
 ```

Explanation:  
- By creating functions for each section, you make the code more modular and easier to maintain. Each function handles specific tasks like inserting or fetching data for different sections, and these functions can be reused throughout the project.  
- The HTML embedding shows how you can dynamically populate sections of your website with data fetched from the database.

**References for PHP PDO:**

1. **PHP PDO Documentation (Official PHP Manual):**
   * <https://www.php.net/manual/en/book.pdo.php>
   * This is the official PHP documentation for PDO. It covers all PDO functions, error handling, and security best practices.
2. **PHP PDO Prepared Statements (Official PHP Manual):**
   * <https://www.php.net/manual/en/pdo.prepared-statements.php>
   * Prepared statements allow you to execute queries securely, preventing SQL injection.
3. **PDO::FETCH Modes (Official PHP Manual):**
   * <https://www.php.net/manual/en/pdostatement.fetch.php>
   * This section explains the different fetch modes available in PDO, such as PDO::FETCH\_ASSOC, PDO::FETCH\_OBJ, and PDO::FETCH\_CLASS.
4. **Working with PDO in PHP (Tutorial by DigitalOcean):**
   * <https://www.digitalocean.com/community/tutorials/how-to-use-pdo-to-perform-crud-operations-in-php>
   * A beginner-friendly tutorial that walks through the basics of PDO and how to perform CRUD operations (Create, Read, Update, Delete) using PDO.
5. **PHP PDO Crash Course by Traversy Media (YouTube Video):**
   * <https://www.youtube.com/watch?v=kEW6f7Pilc4>
   * A great YouTube tutorial that covers the essentials of PHP PDO, including setting up a database connection and performing CRUD operations.
6. **MySQLi vs. PDO: What's the Difference? (Comparison Article):**
   * <https://www.tutorialrepublic.com/mysql-tutorial/mysql-mysqli-vs-pdo.php>
   * This article explains the differences between **MySQLi** and **PDO** in PHP, highlighting the advantages of each.