PHP & Laravel Backend Engineering

Week 4 – Working with Databases using PDO

Program: PHP & Laravel Backend Engineering
Week: 4
Level: Beginner
Duration: 3 Days

Learning Objectives

By the end of this week, you will be able to:

- □ Understand how databases store information
- □ Connect to a database using PHP
- □ Find and change data with SQL
- □ Keep your database safe with PDO
- □ Build a simple contact book

Day 1 – What is a Database?

Think of a database like a digital filing cabinet:

- A phone book stores names and numbers
- A library keeps track of books
- A school saves student records

Let's set up a database:

1. **Install XAMPP** (if you haven't)
    - It's free and includes MySQL
    - Get it from: https://www.apachefriends.org/
2. **Create your first database**
    - Go to http://localhost/phpmyadmin
    - Click "New" and name it "my_contacts"
    - Click "Create"

Basic SQL Commands (The Language for Databases)

```
-- Make a new table
CREATE TABLE contacts (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    phone VARCHAR(20),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Add a new contact
INSERT INTO contacts (name, email, phone)
VALUES ('Ali Khan', 'ali@example.com', '1234567890');

-- See all contacts
SELECT * FROM contacts;

-- Update a contact
UPDATE contacts
SET phone='0987654321'
WHERE email='ali@example.com';

-- Delete a contact
DELETE FROM contacts WHERE id = 1;
```

Practice Time!

1. Make a database called "school"
2. Create a "students" table with: id, name, email, class
3. Add 3 students
4. Try updating and deleting records

Day 2 – Connecting PHP to Your Database

Let's connect PHP to MySQL using PDO:

1. First, create a file called `db.php`:

```
<?php
// Database settings
$host = 'localhost';     // Where the database is
$dbname = 'my_contacts'; // Your database name
$username = 'root';      // Default username for XAMPP
$password = '';          // Default password is empty for XAMPP

try {
    // Make the connection
    $pdo = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);

    // Show errors if something goes wrong
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    echo " Connected to database successfully!";
} catch(PDOException $e) {
    echo " Connection failed: " . $e->getMessage();
}
?>
```

Adding Data to the Database

Create a new file called `add_contact.php`:

```php
<?php
// Include our database connection
require_once 'db.php';

try {
    // Prepare our SQL query
    $sql = "INSERT INTO contacts (name, email, phone)
            VALUES (:name, :email, :phone)";

    // Create a prepared statement
    $stmt = $pdo->prepare($sql);

    // Add our data
    $name = "Sara Ahmed";
    $email = "sara@example.com";
    $phone = "1234567890";

    // Execute the query
    $stmt->execute([
        ':name' => $name,
        ':email' => $email,
        ':phone' => $phone
    ]);

    echo "✅ Contact added successfully!";
} catch(PDOException $e) {
    echo "❌ Error: " . $e->getMessage();
}
?>
```

Viewing Data from the Database

Create a file called `view_contacts.php`:

```php
<?php
require_once 'db.php';

try {
    // Get all contacts
    $stmt = $pdo->query("SELECT * FROM contacts ORDER BY name");
    $contacts = $stmt->fetchAll();

    // Show each contact
    foreach($contacts as $contact) {
        echo "Name: " . htmlspecialchars($contact['name']) . "<br>";
        echo "Email: " . htmlspecialchars($contact['email']) . "<br>";
        echo "Phone: " . htmlspecialchars($contact['phone']) . "<br><br>";
    }
} catch(PDOException $e) {
    echo "❌ Error: " . $e->getMessage();
}
?>
```

Practice Time! 🎯

1. Create a form to add new contacts
2. Show all contacts in a nice table
3. Add a search box to find contacts by name

Day 3 – Build a Simple Contact Book

Let's build a complete contact book with these features:

- Add new contacts
- View all contacts
- Edit existing contacts
- Delete contacts

File Structure

```
/contact_book
  ├── index.php       # Show all contacts
  ├── add.php         # Add new contact
  ├── edit.php        # Edit contact
  ├── delete.php      # Delete contact
  ├── db.php          # Database connection
  └── style.css       # Make it look nice
```

1. First, let's create our database (db.php)

```
<?php
$host = 'localhost';
$dbname = 'my_contacts';
$username = 'root';
$password = '';

try {
    $pdo = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch(PDOException $e) {
    die("□ Could not connect: " . $e->getMessage());
}
?>
```

2. Create the contacts table

Run this in phpMyAdmin (SQL tab):

```
CREATE TABLE IF NOT EXISTS contacts (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    phone VARCHAR(20),
    address TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

3. Create the main page (index.php)

```php
<?php
$title = "My Contact Book";
require_once 'db.php';

// Get all contacts
$stmt = $pdo->query("SELECT * FROM contacts ORDER BY name");
$contacts = $stmt->fetchAll();
?>

<!DOCTYPE html>
<html>
<head>
    <title><?php echo $title; ?></title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <h1> <?php echo $title; ?></h1>

        <a href="add.php" class="btn">➕ Add New Contact</a>

        <?php if (isset($_GET['success'])): ?>
            <div class="success">✅ Contact saved successfully!</div>
        <?php endif; ?>

        <table>
            <tr>
                <th>Name</th>
                <th>Email</th>
                <th>Phone</th>
                <th>Actions</th>
            </tr>

            <?php foreach ($contacts as $contact): ?>
            <tr>
                <td><?php echo htmlspecialchars($contact['name']); ?></td>
                <td><?php echo htmlspecialchars($contact['email']); ?></td>
                <td><?php echo htmlspecialchars($contact['phone']); ?></td>
                <td class="actions">
                    <a href="view.php?id=<?php echo $contact['id']; ?>" class="btn">👁 View</a>
                    <a href="edit.php?id=<?php echo $contact['id']; ?>" class="btn">✏ Edit</a>
                    <a href="delete.php?id=<?php echo $contact['id']; ?>"
                       class="btn delete"
                       onclick="return confirm('Are you sure you want to delete this contact?')">
                       🗑 Delete
                    </a>
                </td>
            </tr>
            <?php endforeach; ?>
        </table>
    </div>
</body>
</html>
```

4. Add a new contact (add.php)

```php
<?php
$title = "Add New Contact";
require_once 'db.php';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    try {
        $sql = "INSERT INTO contacts (name, email, phone, address)
                VALUES (:name, :email, :phone, :address)";
```

```
        $stmt = $pdo->prepare($sql);

        $stmt->execute([
            ':name' => $_POST['name'],
            ':email' => $_POST['email'],
            ':phone' => $_POST['phone'],
            ':address' => $_POST['address']
        ]);

        header('Location: index.php?success=1');
        exit();

    } catch(PDOException $e) {
        $error = "Error: " . $e->getMessage();
    }
}
?>

<!DOCTYPE html>
<html>
<head>
    <title><?php echo $title; ?></title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <h1> <?php echo $title; ?></h1>

        <?php if (isset($error)): ?>
            <div class="error"> <?php echo $error; ?></div>
        <?php endif; ?>

        <form method="POST">
            <div class="form-group">
                <label>Name:</label>
                <input type="text" name="name" required>
            </div>

            <div class="form-group">
                <label>Email:</label>
                <input type="email" name="email" required>
            </div>

            <div class="form-group">
                <label>Phone:</label>
                <input type="tel" name="phone">
            </div>

            <div class="form-group">
                <label>Address:</label>
                <textarea name="address"></textarea>
            </div>

            <button type="submit" class="btn"> Save Contact</button>
            <a href="index.php" class="btn">↩ Back to Contacts</a>
        </form>
    </div>
</body>
</html>
```

5. Make it look nice (style.css)

```
/* Basic styling for our contact book */
```

```
body {
    font-family: Arial, sans-serif;
    line-height: 1.6;
    margin: 0;
    padding: 20px;
    background-color: #f5f5f5;
}

.container {
    max-width: 1000px;
    margin: 0 auto;
    background: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 2px 10px rgba(0,0,0,0.1);
}

h1 {
    color: #333;
    text-align: center;
    margin-bottom: 30px;
}

.btn {
    display: inline-block;
    background: #4CAF50;
    color: white;
    padding: 10px 15px;
    text-decoration: none;
    border-radius: 4px;
    margin: 5px;
    border: none;
    cursor: pointer;
    font-size: 14px;
}

.btn:hover {
    background: #45a049;
    transform: translateY(-1px);
}

.delete {
    background: #f44336;
}

.delete:hover {
    background: #d32f2f;
}

table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 20px;
    box-shadow: 0 1px 3px rgba(0,0,0,0.1);
}

th, td {
    padding: 12px;
    text-align: left;
    border-bottom: 1px solid #ddd;
}

th {
    background-color: #f2f2f2;
    font-weight: bold;
}
```

```
tr:hover {
    background-color: #f9f9f9;
}

.form-group {
    margin-bottom: 15px;
}

label {
    display: block;
    margin-bottom: 5px;
    font-weight: bold;
}

input[type="text"],
input[type="email"],
input[type="tel"],
textarea {
    width: 100%;
    padding: 8px;
    border: 1px solid #ddd;
    border-radius: 4px;
    box-sizing: border-box;
    font-size: 16px;
}

textarea {
    height: 100px;
    resize: vertical;
}

.success {
    background-color: #dff0d8;
    color: #3c763d;
    padding: 10px;
    margin: 15px 0;
    border-radius: 4px;
    text-align: center;
}

.error {
    background-color: #f2dede;
    color: #a94442;
    padding: 10px;
    margin: 15px 0;
    border-radius: 4px;
    text-align: center;
}

.actions {
    white-space: nowrap;
}

.actions .btn {
    padding: 5px 10px;
    margin: 2px;
    font-size: 12px;
}
```

6. Delete a contact (delete.php)

```php
<?php
require_once 'db.php';

if (isset($_GET['id'])) {
    try {
        $stmt = $pdo->prepare("DELETE FROM contacts WHERE id = ?");
        $stmt->execute([$_GET['id']]);

        header('Location: index.php?deleted=1');
        exit();
    } catch(PDOException $e) {
        die("Error deleting record: " . $e->getMessage());
    }
} else {
    header('Location: index.php');
    exit();
}
?>
```

Practice Time!

1. Add a search box to filter contacts
2. Create a "View Details" page for each contact
3. Add form validation (check if email is valid, etc.)
4. Add a "Last Updated" timestamp that updates when a contact is edited

Tips for Success

- Always use prepared statements to prevent SQL injection
- Validate all user input
- Keep your database credentials secure
- Make regular backups of your database
- Test your code after each step

Day 3: Student Management System Project

Project Requirements:

1. Database design for student management
2. CRUD operations for students
3. Search and filter functionality
4. User interface for managing students
5. Data validation and security

Deliverables:

- Database schema and SQL dump
- Complete PHP application
- Documentation of database design
- User manual

Presentation Guidelines:

- 10-15 minute presentation
- Demonstrate working application
- Explain technical decisions
- Be prepared for Q&A

Tips:

- Follow coding standards
- Write clean, maintainable code
- Document your code
- Test thoroughly

Resources:

- Laravel Documentation