Custom Web Application (HTML/PHP) with Database Connection

Now that we have set up a Web Server with PHP and phpMyAdmin and configured an independent database server, we can write our first server application: a simple **Contact List**.

A contact list/phone book application is a classic and simple example, as it covers all the basic database operations (**CRUD**: Create, Read, Update, Delete) and is perfectly suited for a quick PHP implementation.

💡 Example Application: Simple Contact List (CRUD)

This application consists of a single PHP page and one database table.

🎯 The application can: - **READ:** Display all saved contacts in a table. - **CREATE:** Add new contacts (Name, Phone number, Email). - **UPDATE:** Edit existing contacts. - **DELETE:** Delete contacts.

This document describes the fundamental procedure.

Creating the Database and Tables

First, the database is created.

```
CREATE DATABASE contacts_db;

USE contacts_db;

CREATE TABLE contacts (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    phone VARCHAR(20),
    email VARCHAR(100)
);
```

The three SQL commands can be executed directly in the MariaDB console or using phpMyAdmin. To do this, open the "SQL" area in phpMyAdmin, copy the three commands, and confirm with "OK."

This should create the database and the table.

Creating the PHP Application (Apache/PHP Container)

In the Apache HTML directory, create a new folder "Kontaktliste" (ContactList) – `mkdir /var/www/html/Kontaktliste`.

Create a single file, e.g., index.php, in the Document Root of your Apache container (/var/www/html/Kontaktliste/).

```
sudo nano index.php
```

This file must establish the database connection and contain the logic for displaying and processing the forms. In this example, we use the PHP class **PDO (PHP Data Object)** for the data connection.

PHP Documentation PDO

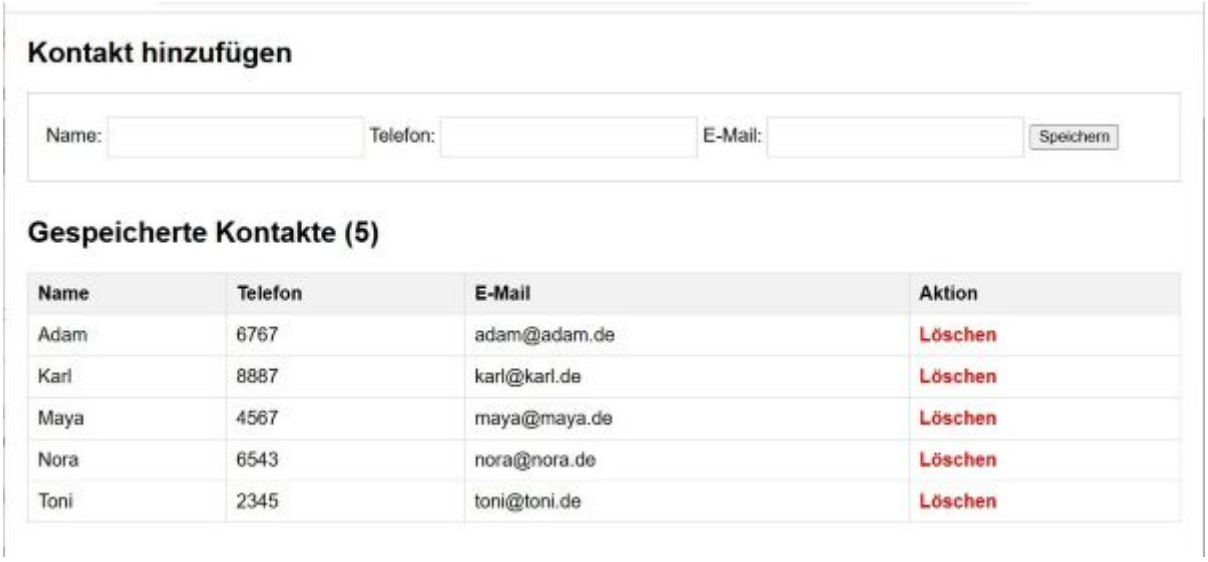
```
<?php
// ============================================================
// 1. CONFIGURATION & DATABASE CONNECTION
// ============================================================
$host = '192.168.137.120'; // IP of your MariaDB container
$db   = 'contacts_db';
$user = 'pdal';          // Your database user
$pass = 'JadeHS20';       // Your password
$charset = 'utf8mb4';

$dsn = "mysql:host=$host;dbname=$db;charset=$charset";
$options = [
    PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    PDO::ATTR_EMULATE_PREPARES   => false,
];

try {
     $pdo = new PDO($dsn, $user, $pass, $options);
} catch (\PDOException $e) {
     throw new \PDOException($e->getMessage(), (int)$e->getCode());
}


// ============================================================
// 2. FORM PROCESSING (CRUD Logic)
// ============================================================

// Example for CREATE (New Contact)
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['action']) &&
$_POST['action'] === 'add') {
    $stmt = $pdo->prepare("INSERT INTO contacts (name, phone, email) VALUES (?, ?,
?)");
    $stmt->execute([$_POST['name'], $_POST['phone'], $_POST['email']]);
    header('Location: index.php'); // Redirects after POST prevent double
submissions
    exit;
}

// Example for DELETE
if (isset($_GET['delete_id'])) {
    $stmt = $pdo->prepare("DELETE FROM contacts WHERE id = ?");
    $stmt->execute([$_GET['delete_id']]);
    header('Location: index.php');
```

```
    exit;
}


// ============================================================
// 3. QUERY DATA (READ)
// ============================================================
$stmt = $pdo->query('SELECT * FROM contacts ORDER BY name');
$contacts = $stmt->fetchAll();

?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Simple Contact List</title>
    <style>
        body { font-family: Arial, sans-serif; margin: 20px; }
        table { width: 100%; border-collapse: collapse; margin-top: 20px; }
        th, td { border: 1px solid #ddd; padding: 8px; text-align: left; }
        th { background-color: #f2f2f2; }
        .form-container { margin-bottom: 30px; padding: 15px; border: 1px solid
#ccc; }
        input[type="text"], input[type="email"] { padding: 8px; margin: 5px 0;
display: inline-block; border: 1px solid #ccc; width: 200px; }
        .delete-btn { color: red; text-decoration: none; font-weight: bold; }
    </style>
</head>
<body>

    <h2>Add Contact</h2>
    <div class="form-container">
        <form method="POST">
            <input type="hidden" name="action" value="add">
            <label for="name">Name:</label>
            <input type="text" id="name" name="name" required>
            <label for="phone">Phone:</label>
            <input type="text" id="phone" name="phone">
            <label for="email">Email:</label>
            <input type="email" id="email" name="email">
            <button type="submit">Save</button>
        </form>
    </div>

    <h2>Saved Contacts (<?= count($contacts) ?>)</h2>
    <?php if (count($contacts) > 0): ?>
        <table>
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Phone</th>
                    <th>Email</th>
                    <th>Action</th>
                </tr>
```

```
            </thead>
            <tbody>
                <?php foreach ($contacts as $contact): ?>
                    <tr>
                        <td><?= htmlspecialchars($contact['name']) ?></td>
                        <td><?= htmlspecialchars($contact['phone']) ?></td>
                        <td><?= htmlspecialchars($contact['email']) ?></td>
                        <td>
                            <a class="delete-btn" href="?delete_id=<?=
$contact['id'] ?>"
                               onclick="return confirm('Are you sure you want to
delete this contact?');">Delete</a>
                        </td>
                    </tr>
                <?php endforeach; ?>
            </tbody>
        </table>
    <?php else: ?>
        <p>No contacts saved yet.</p>
    <?php endif; ?>

</body>
</html>
```

You can insert the script directly using `nano` or create the index file on the client computer and then upload it to the server using `scp` or an FTP tool.

After creating the script, test the application – `http://<IP-Apache-Server>/Kontaktliste/`.



Further Development

If an application has multiple scripts, it doesn't make sense to define the database connection every time. Therefore, the database connection is separated and included in the script using the `include` or `require_once` statement.

> Hint: The difference between an `include` and `require_once` statement is that `require_once` aborts the script if an error occurs. This is always advisable when dealing with a critical section – in this case, the database connection.

To do this, create a file named `db_config.php`:

```php
<?php
// ============================================================
// DATABASE CONFIGURATION
// ============================================================

$host = '192.168.137.120'; // IP of your MariaDB container
$db   = 'contacts_db';
$user = 'pdal';           // Your database user
$pass = 'JadeHS20';        // Your password
$charset = 'utf8mb4';

$dsn = "mysql:host=$host;dbname=$db;charset=$charset";
$options = [
    // Output errors as PDOException
    PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
    // Default fetch mode: associative array
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    // Disable emulation of prepared statements (better performance and security)
    PDO::ATTR_EMULATE_PREPARES   => false,
];

try {
     // The global database connection ($pdo) is established
     $pdo = new PDO($dsn, $user, $pass, $options);
} catch (\PDOException $e) {
     // In case of connection errors (e.g., wrong IP or password)
     // In a production environment, only a general error message should be
displayed here!
     die("Database connection failed: " . $e->getMessage());
}
```

Now the database connection can be imported:

```php
<?php
// ============================================================
// 1. IMPORT DATABASE CONNECTION
// ============================================================
require_once 'db_config.php';

// ============================================================
// 2. FORM PROCESSING (CRUD Logic)
// The $pdo variable is now available
// ============================================================
```

```php
// Example for CREATE (New Contact)
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['action']) &&
$_POST['action'] === 'add') {
    $stmt = $pdo->prepare("INSERT INTO contacts (name, phone, email) VALUES (?, ?,
?)");
    $stmt->execute([$_POST['name'], $_POST['phone'], $_POST['email']]);
    header('Location: index.php');
    exit;
}

// ... Rest of the code remains unchanged ...

// ============================================================
// 3. QUERY DATA (READ)
// ============================================================
$stmt = $pdo->query('SELECT * FROM contacts ORDER BY name');
$contacts = $stmt->fetchAll();

// ... HTML code ...
```

In this way, the database connection is defined only once and can be imported as often as needed. If changes become necessary, they are only implemented in one place.

Adjust the code accordingly.

> Hint: PHP provides several drivers and plugins for accessing and handling MySQL/MariaDB. See e.g.: PHP Drivers for MySQL.

> Hint 2: In production systems, the database configuration and other critical scripts would not be placed in the web directory. The import must then be done with the full path – e.g., `include /var/config/db_config.php`.

Sources

PHP Documentation PDO

PHP Drivers for MySQL
