CRUD Application with MVC Architecture and Blade Templates

In this workshop, you will follow the MVC architectural pattern and implement the Blade templating engine.

PART 1: Setup

1. Create employees table in your database to implement CRUD functionality for employee.

```
CREATE TABLE employees (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    title VARCHAR(100),
    skills TEXT
);
```

2. Create a new project inside your htdocs folder (eg.workshop8) and create following subfolders inside it.

/app

/cache

Note: Provide permission to cache folder: `chmod 777 cache`

2. Create following folders inside **/app** directory to implement MVC architecture.

/controllers

/models

/views

2. Place your database connection file (db.php) in the project root directory (inside the workshop8 folder). Use your own database.

```php
<?php

$host = 'localhost';
$db   = 'school_db';
$user = 'root';
$pass = '';

try {
    $pdo = new PDO(
        "mysql:host=$host;dbname=$db;charset=utf8",
        $user,
        $pass,
        [
            PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION
        ]
    );
} catch (PDOException $e) {
    die("Database connection failed");
}
```

PART 2: Using Template Engine

Open your terminal/command prompt and navigate to your project root directory (inside the workshop8 folder).

1. Install the Blade template engine using Composer. Run this command in the project directory.

    - *composer require jenssegers/blade:1.4.0*

This command will create vendor folder with dependencies along with composer.json and composer.lock file in your project folder.

PART 3: Creating Model

1. Create a file app/models/employee.php that will functions related to database operation in employee table.

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

function getAllEmployees()
{
    global $pdo;
    return $pdo->query("SELECT * FROM employees")
               ->fetchAll(PDO::FETCH_ASSOC);
}

function getEmployeeById($id)
{
    global $pdo;
    $stmt = $pdo->prepare("SELECT * FROM employees WHERE id =
?");
    $stmt->execute([$id]);
    return $stmt->fetch(PDO::FETCH_ASSOC);
}


function insertEmployee($name, $title, $skills)
{
    global $pdo;
    $stmt = $pdo->prepare(
          "INSERT INTO employees (name, title, skills) VALUES
(?, ?, ?)"
    );
    $stmt->execute([$name, $title, $skills]);
}

function updateEmployee($id, $name, $title, $skills)
{
    global $pdo;
    $stmt = $pdo->prepare(
```

```
        "UPDATE employees SET name = ?, title = ?, skills = ?
WHERE id = ?"
    );
    $stmt->execute([$name, $title, $skills, $id]);
}

function deleteEmployee($id)
{
    global $pdo;
    $stmt = $pdo->prepare("DELETE FROM employees WHERE id = ?");
    $stmt->execute([$id]);
}
```

PART 4: Creating the Controller

1. Create a file app/controllers/employee_controller.php that will contain the function to control http request related to employee.

```
<?php
require_once __DIR__ . '/../models/employee.php';

function handleEmployeeRequest()
{
    $action = $_GET['action'] ?? 'index';

    if ($action === 'store') {
        insertEmployee(
            $_POST['name'],
            $_POST['title'],
            $_POST['skills']
        );
        header("Location: index.php");
        exit;
    }

    if ($action === 'update') {
        updateEmployee(
            $_POST['id'],
            $_POST['name'],
            $_POST['title'],
          $_POST['skills']
        );
        header("Location: index.php");
        exit;
    }

     if ($action === 'delete') {
        deleteEmployee($_GET['id']);
```

```
        header("Location: index.php");
        exit;
    }

    if ($action === 'edit') {
        return [
            'view'     => 'employee.edit',
            'employee' => getEmployeeById($_GET['id'])
        ];
    }

    if ($action === 'create') {
        return ['view' => 'employee.create'];
    }

    return [
        'view'      => 'employee.index',
        'employees' => getAllEmployees()
    ];
}
```

PART 5: Setting Up Blade

1. Create *index.php* in your root directory and Set up the Blade engine by specifying the paths to your views and cache directories.

```
<?php
require 'vendor/autoload.php';
require 'app/controllers/employee_controller.php';

use Jenssegers\Blade\Blade;

$blade = new Blade('app/views', 'cache');

$data = handleEmployeeRequest();
echo $blade->render($data['view'], $data);
```

PART 6: Creating Blade Views

1. Create **layout.blade.php** with the following code inside views directory.

```
<!DOCTYPE html>
<html>
<head>
    <title>@yield('title')</title>
</head>
<body>

<h2>Simple Employee CRUD</h2>
<hr>

@yield('content')

</body>
</html>
```

2. Create employee folder inside **views** directory and create following blade templates extending layout template:

create.blade.php

```
@extends('layout')

@section('title','Create Employee')

@section('content')
<form method="post" action="index.php?action=store">
    <input name="name" placeholder="Name" required><br><br>
    <input name="title" placeholder="Job Title"
required><br><br>
    <input name="skills" placeholder="Skills (comma
separated)" required><br><br>
    <button>Save</button>
</form>
@endsection
```

edit.blade.php

```
@extends('layout')

@section('title','Edit Employees')

@section('content')
<form method="post" action="index.php?action=update">
    <input type="hidden" name="id" value="{{ $employee['id']
}}">

    <input name="name" value="{{ $employee['name']
}}"><br><br>
    <input name="title" value="{{ $employee['title']
}}"><br><br>
    <input name="skills" value="{{ $employee['skills']
}}"><br><br>

    <button>Update</button>
</form>
@endsection
```

index.blade.php

```
@extends('layout')

@section('title','Employee Database')


@section('content')
    <a href="index.php?action=create">Add Employee</a>

    <ul>
        @foreach ($employees as $e)
            <li>
         <strong>{{ $e['name'] }}</strong> - {{ $e['title'] }}

                <br>
                Skills:
                <ul>
                    @foreach (explode(',', $e['skills']) as
$skill)
                        <li>{{ trim($skill) }}</li>
                    @endforeach
                </ul>

                    <a href="index.php?action=edit&id={{
$e['id'] }}">Edit</a> |
                 <a href="index.php?action=delete&id={{
$e['id'] }}">Delete</a>
             </li>
         @endforeach
    </ul>
@endsection
```

Try to understand the following Blade syntax features and MVC flow.

a. @extends to inherit from the master layout

b. @section and @endsection to define content sections

c. {{ }} to display data

d. @foreach for looping through student records

e. @if, @else, @endif for conditional logic

Your final project structure should look like this: