MyMvc Framework Documentation

A Lightweight PHP MVC Framework

Contents

1 Overview

MyMvc is a lightweight PHP MVC framework designed for building modern, efficient web applications. It leverages **Twig** as its templating engine and offers features such as custom routing, a flexible query builder, middleware support, and robust error logging, making it ideal for developers seeking simplicity and flexibility.

2 Features

MyMvc provides a clean and intuitive structure with the following key features:

- **MVC Architecture**: Organizes applications with clear separation of logic, presentation, and data.
- **Twig Templating**: Enables dynamic, secure, and reusable templates with Twig's powerful engine.
- **Custom Routing**: Supports dynamic parameters for clean and flexible URLs.
- **Query Builder**: Simplifies database operations with an intuitive API (to be implemented).
- **Middleware Support**: Allows custom request handling logic (to be implemented).
- **Error Logging**: Automatically logs errors to `error.log` for debugging.

3 Installation

Follow these steps to set up MyMvc on your local environment:

3.1 Download the Framework

- Download the MyMvc archive from the repository: `https://github.com/bappaSoumya/myProjectWorks/blob/master/MyMvc.rar`.
- Unzip the archive to your desired directory (e.g., `D:\xampp\htdocs\MyMvc`).

3.2 Navigate to the Project Directory

```
cd MyMvc
```

3.3 Install Dependencies

- Ensure Composer is installed.
- Run the following command to install Twig and other dependencies:

```
composer install
```

3.4 Configure the Database

- Open `config/config.php` and set your database credentials:

```php
<?php
define('DB_DSN', 'mysql:host=localhost;dbname=your_database;charset=
   utf8mb4');
define('DB_USER', 'your_username');
define('DB_PASS', 'your_password');
define('DB_NAME', 'your_database');
```

3.5 Set Up Constants

- In `config/config.php`, configure the BASE_URL:

```php
<?php
define('BASE_URL', 'http://localhost:8000/');
```

3.6 Start the Development Server

- Run the PHP built-in server:

```
php -S localhost:8000
```

- Access the application at `http://localhost:8000/`.

4 Folder Structure

The MyMvc framework follows a clear and organized folder structure:

```
MyMvc/
├── app/
│   ├── controllers/       # Application controllers (e.g., HomeControlle
│   ├── core/              # Core framework files (Router.php, Controller
│   ├── models/            # Application models (e.g., User.php)
├── config/                # Configuration files (config.php)
├── public/                # Publicly accessible files (optional, if not
├── vendor/                # Composer dependencies (Twig, etc.)
├── resources/
│   └── views/             # Twig templates (layouts/app.twig, home.twig)
├── cache/
│   └── twig/              # Twig cache directory
├── error.log              # Error log file
├── index.php              # Application entry point
├── .htaccess              # URL rewriting rules (for Apache)
└── composer.json          # Composer dependencies
```

5 Usage

5.1 Defining Routes

Routes are currently parsed dynamically based on URL segments (e.g., `/user/123` maps to `UserController::index(123)`). To implement custom routing, extend `Router.php` with a route array. Example:

```
<?php
// app/core/Router.php
protected $routes = [
   '/' => ['controller' => 'HomeController', 'method' => 'index'],
   '/blog' => ['controller' => 'HomeController', 'method' => 'blog'
      ],
   '/user/{id}' => ['controller' => 'UserController', 'method' => '
      profile'],
];
```

5.2 Creating Controllers

Controllers extend the `Controller` class and handle application logic. Example:

```
<?php
// app/controllers/HomeController.php
class HomeController extends Controller {
   public function index() {
      $data = ['title' => 'Welcome to MyMvc'];
      $this->view('home', $data);
   }
}
```

5.3 Using Models

Models extend the `Model` class and use PDO for database interactions. A query builder is planned but not yet implemented. Example:

```
<?php
// app/models/User.php
class User extends Model {
   public function getUsers() {
      return $this->db->query('SELECT id, name FROM users')->
         fetchAll();
      // Future query builder example:
      // return $this->queryBuilder()
      // ->table('users')
      // ->select(['id', 'name'])
      // ->get();
   }
}
```

5.4 Creating Views

Views are Twig templates located in `resources/views/`. Example:

```
<!-- resources/views/home.twig -->
{% extends 'layouts/app.twig' %}

{% block content %}
   <h1>{{ title }}</h1>
   <ul>
      {% for user in users %}
         <li>{{ user.name }}</li>
      {% endfor %}
   </ul>
{% endblock %}
```

5.5 Error Logging

Errors are logged to `error.log` in the root directory. Ensure error reporting is enabled in `index.php`:

```
<?php
// index.php
error_reporting(E_ALL);
ini_set('display_errors', 1);
ini_set('log_errors', 1);
ini_set('error_log', 'D:/xampp/htdocs/MyMvc/error.log');
```

6 Troubleshooting

6.1 Common Issues

6.1.1 Class "TwigFunction" Not Found

- Ensure Twig is installed:

```
composer require twig/twig
```

- Verify `vendor/autoload.php` is included in `index.php`.

6.1.2 Unknown "call" Function in Twig

- The `call` function is not implemented. If needed, add it to `Controller.php`:

```
<?php
use Twig\TwigFunction;
// In Controller::__construct()
$this->twig->addFunction(new TwigFunction('call', function (
   $functionName, ...$args) {
   if (function_exists($functionName)) {
        return call_user_func_array($functionName, $args);
    }
    return ”Error: Function ’$functionName’ does not exist.”;
}));
```

6.1.3 Database Connection Issues

- Verify credentials in `config/config.php`.
- Ensure MySQL is running in XAMPP.
- Create the database:

```
CREATE DATABASE your_database;
```

6.1.4 Routing Issues

- If URLs don't resolve (e.g., `http://localhost:8000/blog`), check `Router.php`'s `parseUrl` method.
- Add debug logging:

```
error_log('REQUEST_URI: ' . ($_SERVER['REQUEST_URI'] ?? ''));
```

6.1.5 Twig Cache Issues

- Clear the Twig cache:

```
del /Q D:\xampp\htdocs\MyMvc\cache\twig\*
```

- Ensure `cache/twig/` is writable:

```
icacls ”D:\xampp\htdocs\MyMvc\cache\twig” /grant Everyone:F /T
```

7 Notes

- **Database Configuration**: The reference mentions `database.php`, but the setup uses `config.php`.
- **Custom Routing**: The route array example is not implemented; the current `Router.php` uses dynamic routing.
- **Query Builder**: Not yet implemented; use PDO queries as a fallback.
- **Middleware**: Not implemented; extend `Router.php` for middleware support.
- **Public Directory**: The setup uses the root `index.php`, not a `public/` directory.

8 License

This project is licensed under the MIT License. See the LICENSE file for details.

9 Contributing

Contributions are welcome! Please submit pull requests or issues to the GitHub repository.

10 Contact

For questions or support, contact the maintainer at [your-email@example.com] or open an issue on GitHub.