УДК 004

Реализация MVC паттерна на PHP

Круглик Роман Игоревич
Приамурский государственный университет им. Шолом-Алейхема
Студент

Аннотация
В статье подробно реализуется основной MVC паттерн на языке программирования PHP. Данный пример является шаблоном и может быть интегрирован в любой проект.
Ключевые слова: MVC, программирование, паттерн.

Implementation of MVC pattern in PHP

Kruglik Roman Igorevich
Sholom-Aleichem Priamursky State University
Student

Abstract
In article details the main MVC pattern in the PHP programming language. This example is a template and can be integrate into any project.
Keywords: MVC, programming, pattern.

На сегодняшний день огромное влияние в мире веб-разработки имеют паттерны проектирования. Самым популярным является MVC. Его используют не только для серверной части, но и при проектировании клиентской. На MVC создан JavaScript фреймворк AngularJs.

Исследования в области применения концепции MVC не заканчиваются и по сей день. В статье К.А. Жерденко [1] рассмотрены современные подходы к разработке клиентской части веб-приложений(MVC). С.М. Гардейчик, А.И. Шербаф [2] приводят краткое описание PHP-фреймворка Laravel, реализующего шаблон MVC. В статье О.Н. Симонова [3] рассказывается о шаблонах проектирования MVC как эффективное средство построения архитектуры программной системы. Р.И. Ибраимов [4] создаёт тесты для spring mvc контроллеров.

В данной статье разработан MVC паттерн на языке программирования PHP. Структура папок будет выглядеть так (см. рис. 1).

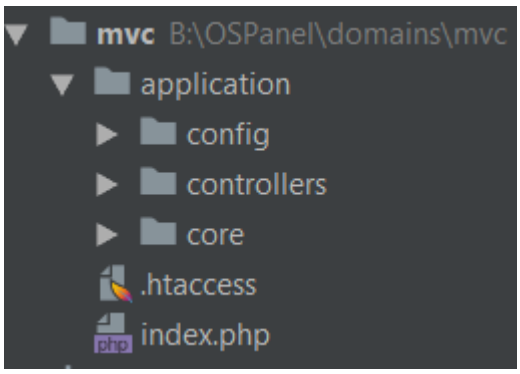

Рисунок 1. Структура папок

Для начала нужно настроить файл .htaccess. Пользователь всегда будет переходить на index.php (см. рис. 2).

```
RewriteEngine on
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ index.php
```

Рисунок 2 .htaccess

MVC будет состоять из главного файла, который подключает все классы. В данном случае создан класс-маршрутизатор(Router.php) (см. рис. 3).

```
<?php
use application\core\Router;
    spl_autoload_register(function ($class) {
       $str =  str_replace( search: '\\', replace: '/',$class);
        include '' . $str . '.php';
    });
$router = new Router();
$router->start();
```

Рисунок 3. Файл index

Чтобы не писать подключение каждого класса отдельно, была разработана функция автоматического подключения всех классов, которые прописываются через use. Далее создаётся экземпляр класса и вызывается метод start. В маршрутизаторе сравниваются путь, по которому пользователь хочет получить страницу и массив путей, который имеется в проекте. Они хранятся в файле path.php (см. рис. 4).

```php
<?php
return [
        '' => [
            'controller' => 'Main',
            'action' => 'index',
        ],
            'posts/index' =>[
            'controller' => 'Post',
            'action' => 'index',
        ]
];
```

Рисунок 4. Файл путей

При попытке зайти на страницу posts/index будет вызван PostController и метод внутри indexAction. Посмотрим на класс маршрутизатор (см. рис. 5,6).

```php
namespace application\core;
class Router
{
    protected $action;
    protected $route = [];
    public function __construct()
    {
        $mass = require 'application/config/routes.php';
        foreach ($mass as $controller => $action)
        {
            $this->route[$controller] = $action;
        }
    }
    public function check()
    {
        $url = trim($_SERVER['REQUEST_URI'], charlist: '/');
        foreach ($this->route as $key =>$val)
        {
            if ($url == $key)
            {
                $this->action = $key;
                return true;
            }
        }
        return false;
    }
```

Рисунок 5. Класс Router

```php
    public function path()
    {
        if ($this->check())
        {
            $path = 'application\controllers\\'. ucfirst($this->route[$this->action]['controller']) . 'Controller';
            if (class_exists($path))
            {
                $action = $this->route[$this->action]['action']  .'Action';
                if (method_exists($path,$action))
                {
                    $result = new $path();
                    $result->$action();
                }
            }
        }
    }
    public function start()
    {
        $this->path();
    }
}
```

Рисунок 6. Класс Router

Устанавливается пространство имён. Переменная action и route хранят данные о путях, которые находятся в route.php. Конструктор подключается сразу при вызове класса. Берётся массив всех путей и перезаписываются в переменную route. Далее из главного файла вызывается метод start, который инициализирует метод path. После проверяется метод check, в котором запрашивается ссылка, где находится пользователь. После через цикл сравнивается с массивом путей, если совпадение, то возвращается true, и записывается метод совпадения. Соответственно, только при совпадении активируется метод path. Внутри собирается путь к контроллеру и action, которые вызываются, если существуют. Теперь перейдём к контроллеру (см. рис. 7).

```php
<?php
namespace application\controllers;
class PostController
{
    public function indexAction()
    {
        echo 'Страница постов';
    }
}
```

Рисунок 7. Класс PostController

Устанавливается пространство имён и функция indexAction(может быть любой, который указывается в путях и вызовах) (см. рис. 8).

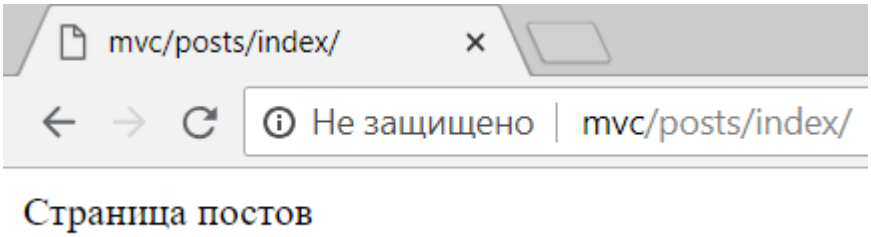

Рисунок 7. Вызов метода index в контроллере

Контроллер не когда не нагружается, сейчас можно подключить html файл с видом и передавать любые переменные, возможно подключить шаблоны, модели с БД. Основные принципы паттерна были разобраны, и на этой основе можно загрузить любой проект.

Библиографический список

1. Жерденко К.А. Применение mvc-паттерна во front-end разработке веб-программ // Синергия Наук. 2018. № 22. С. 653-657.
2. Гардейчик С.М., Шербаф А.И. PHP-фреймворк laravel с использованием архитектурной модели MVC // Перспективные направления развития отечественных информационных технологий Материалы III межрегиональной научно-практической конференции. Научный редактор Б.В. Соколов. 2017. С. 133-135.
3. Симонова О.Н. Шаблон проектирования mvc как эффективное средство построения архитектуры программной системы // Студенческий научный форум - 2014 VI Международная студенческая электронная научная конференция: Электронное издание. 2014.
4. Ибраимов Р.И., Джемалетдинов А.Б., Шевченко А.А. Spring boot: создание тестов для spring mvc контроллеров // Информационно-компьютерные технологии в экономике, образовании и социальной сфере. 2018. № 4 (18). С. 104-111.