webmcr_gm_edition Documentation

webmcr_gm_edition

июл. 25, 2018

Оглавление

Публичный проект JM Organization для проекта Grand-Mine. Проект носит кодовое название webmcr_gm_edition. Представляет из себя cms для сайтов проектов игры Minecraft. Основывается уже на готовой cms WebMCR и является улучшенной её версией.

Мы думаем, что эта версия вам понравится куда больше. **Удобстово и красота кода - залог его работы**!

Глава 1

PSR-0, PSR-4

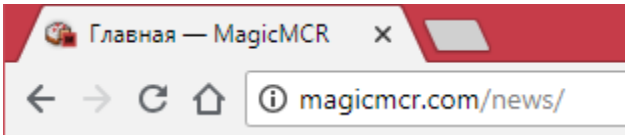
В MagicMCR была интегрирована поддержка стандартов автозагрузки файлов. Вам не нужно беспокоиться о том, будет ли загружен ваш класс. Также, это дало возможность использовать сторонние php-библиотеки, к примеру, FastRoute, которая используется в MagicMCR для маршрутизации.

Глава 2

Маршрутизация

Как уже сказано выше, в движке используется новый алгоритм определения обработчика маршрута. Маршруты были адаптированы под человеко-понятные.

2.1 Маршрутизация



Современные cms имеют поддержку ЧПУ (семантического URL). Такой URL понятен человеку и легко запоминается. Но в его реализации всегда есть спорные моменты и определённые сложности. Но, с библиотекой FastRoute определённые сложности ушли на второй план, а некоторые вовсе не ощущались.

В MagicMCR Семантический URL реализован по методу RESTAPI `__ROUTE__ => Handler->{action}();`

2.1.1 Маршрутизатор

Введение

Для того чтобы при обращении по адресу http://site.com/users/4900.user/ ваш сайт понял какие действия необходимо выполнить вам необходимо в папке, где хранятся маршруты вашего сайта, описать ваш маршрут.

Данная папка находится по пути `MCR_ROOT . configs/routes/`. Когда вы только установили MagicMCR в данной папке будет находится файл `master.php`. Вы можете использовать его для описания ваших маршрутов.

Примечание: Учтите, что файл `master.php` описывает маршруты модулей, которые идут по умолчанию в MagicMCR. Рекомендуется создать свой файл для добавления своих маршрутов.

В MagicMCR нету привязки к имени данного файла - вы можете называть его так, как вам будет удобно. MagicMCR сам проиндексирует данный файл, предварительно загрузив переменную `route_collector $router`.

Регистрация маршрутов (Методы запросов)

Для описания маршрутов в MagicMCR используется библиотека FastRoute. Она предоставляет набор методов для добавления своих маршрутов. Суть добавления маршрута в том, что вы говорите MagicMCR какой обработчик будет обрабатывать указанный вами маршрут. Когда у вас появилась необходимость описать маршрут */news*, вы должны использовать подобную конструкцию:

```php
<?php

use mcr\http\routing\route_collector
/** @var route_collector $router */

$router->addRoute('GET', '/news', '\modules\news@index', 'home');
```

route_ collector::addRoute - добавляет указанный маршрут в коллекцию маршрутов. С этой коллекции будет производится поиск маршрута, к которому обратились в данный момент.

```php
<?php

route_collector::
    addRoute( string $httpMethod, string $route, mixed $handler [, string $name = null] );
```

тип	параметр	Описание
required	**httpMethod**	Имя метода запроса, записанное в формате *ВСЕ ЗАГЛАВНЫЕ*.
required	**route**	Маршрут, который необходимо обработать.
required	**handler**	Имя обработчика / Функция-обработчик, который будет обрабатывать маршрут, когда поступил запрос по данному маршруту
optional	**name**	Имя маршрута.

MagicMCR поддерживает базовые HTTP методы запросов, для них описаны быстрые методы с подобными названиями. Ниже представлены методы с описанием аргументов, которые они принимают.

[GET]

Метод GET запрашивает представление ресурса. Запросы с использованием этого метода могут только извлекать данные.

```
<?php route_collector::get( string $route, mixed $handler [, string $name = null ] )
```

[HEAD]

HEAD запрашивает ресурс так же, как и метод GET, но без тела ответа.

```
<?php route_collector::head( string $route, mixed $handler [, string $name = null ] )
```

[POST]

POST используется для отправки сущностей к определённому ресурсу. Часто вызывает изменение состояния или какие-то побочные эффекты на сервере.

```
<?php route_collector::post( string $route, mixed $handler [, string $name = null ] )
```

[PUT]

PUT заменяет все текущие представления ресурса данными запроса.

```
<?php route_collector::put( string $route, mixed $handler [, string $name = null ] )
```

[PATCH]

PATCH используется для частичного изменения ресурса.

```
<?php route_collector::patch( string $route, mixed $handler [, string $name = null ] )
```

[DELETE]

DELETE удаляет указанный ресурс.

```
<?php route_collector::delete( string $route, mixed $handler [, string $name = null ] )
```

Все вышеперечисленные методы принимают схожие параметры, которые описаны ниже.

тип	параметр	Описание
required	**route**	Маршрут, который необходимо обработать.
required	**handler**	Имя обработчика / Функция-обработчик, который будет обрабатывать маршрут, когда поступил запрос по данному маршруту
optional	**name**	Имя маршрута.

Примечание: Если появилась необходимость в добавлении другого метода используйте функцию `addRoute` из класса `route_collector`. `addRoute( string $httpMethod, string $route, mixed $handler [, string $name = null] )`, где $httpMethod необходимо передавать имя метода в формате uppercase.

handler - Как уже оговорено - это обработчик, который будет обрабатывать запрос, поступивший на данный маршрут. В роли обработчика может выступать, как функция, так и её имя. Но данное имя должно иметь специальный формат записи. `{module_full_name}@{handler_name}` - означает, что в модуле *{module_full_name}* будет вызван метод *{handler_name}*. Но если будет передана сама функция, то будет выполнена эта функция.

name - Не обязательный параметр. Имя маршрута. Если вы хотите генерировать ссылку на данный маршрут, то вам необходимо указать имя этого маршрута. Формат записи имени четко не регламентирован - вы можете использовать удобную для вас запись.

В результате вам необходимо записать свои маршруты таким образом (ниже приведён пример файла с описанием маршрутов):

custom.php

```php
<?php

use mcr\http\request;
use mcr\http\routing\route_collector;

/** @var route_collector $router */

$router->addGroup('admin/', function(route_collector $router) {
    $router->get('news', '\modules\admin\news@news_controll', 'admin_cp.news.index');
    $router->post(
        'news[/{id}[/{action}]]', '\modules\admin\news@news_controll', 'admin_cp.news.index'
    );
});

$router->get('/', '\modules\news@index', 'home');

$router->get('/users[/{id}]', '\modules\news@index', function (request $request) {
    $user = $request->id;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
19
20      if (!empty($user)) return 'User id: ' . $user;
21
22      return 'User list';
23  });
```

Описание маршрутов Defining routes

route - принимает строку, в которой содержится маршрут, который необходимо обработать. Формат записи данного маршрута позволяет использовать нативные регулярные выражения.

`$router->addRoute('GET', '/user/{id:\d+}', 'handler');` - будет обрабатывать маршрут в котором параметр id любое число.

По умолчанию route использует синтаксис, где `{foo}` указывает на переменную с именем `foo` и сопоставляет регулярное выражение `[^/]+`. Чтобы настроить шаблон, вы можете указать собственный шаблон, написав `{bar:[0-9]+}`.

Для того чтобы записать маршрут, в котором есть необязательные его части используйте [*optional*] `$router->addRoute('GET', '/user[/{id:\d+}]', 'handler');`

```php
<?php

// Пример маршрута с не обязательной частью
$router->addRoute('GET', '/user/{id:\d+}[/{name}]', 'handler');
// Эквивалентен этим двум маршрутам
$router->addRoute('GET', '/user/{id:\d+}', 'handler');
$router->addRoute('GET', '/user/{id:\d+}/{name}', 'handler');

// Возможны также множественные вложенные дополнительные части
$router->addRoute('GET', '/user[/{id:\d+}[/{name}]]', 'handler');

// Этот маршрут НЕ является допустимым, поскольку дополнительные части могут появляться только в␣
↪конце
$router->addRoute('GET', '/user[/{id:\d+}]/{name}', 'handler');
```

Рекомендации

1. Файл с вашими маршрутами рекомендуется называть таким образом: `{module_name}_routes.php` / `custom.php`.
2. Имена маршрутов формируйте по такому шаблону: `{module_name}.{submodule}.{action}`.