

Запись на курсы по HTML, CSS, JavaScript, PHP, фреймворкам и CMS,  
а также: помощь в поиске работы и заказов, стажировка на реальных проектах→

урок 105 из 107

[Верстка](#) [JavaScript](#) [PHP](#) [NodeJs](#) [Vue](#) [React](#) [Laravel](#) [WordPress](#) [AJAX](#) [Парсинг](#)

Бесплатный тренинг "Работа с узлами DOM в JavaScript". Начало 10-го декабря→  
Занимательные задачи JavaScript. Перезапуск! Начнем, когда наберется более 100 желающих→

# Разработка диспетчера в своем MVC фреймворке

Вспомним текущее содержимое файла index.php:

```
1  <?php
2      namespace Core;
3
4      error_reporting(E_ALL);
5      ini_set('display_errors', 'on');
6
7      spl_autoload_register(function($class) {
8          // ваша реализация автозагрузки
9      });
10
11     $routes = require $_SERVER['DOCUMENT_ROOT'] .
12         '/project/config/routes.php';
13
14     $track = ( new Router ) -> getTrack($routes, $_SERVER['REQUEST_URI']);
15
16     ?>
```

Как вы видите, пока у нас наш роутер возвращает объект класса Track, содержащий имя контроллера, имя действия и параметры из адресной строки. Кроме того, в предыдущем уроке мы с вами сделали родителя всех контроллеров.

Теперь мы можем сделать так, чтобы происходил вызов метода контроллера, данные которого содержатся в нашей переменной \$track.

Этим будет заниматься специальный класс Dispatcher. Диспетчер будет получать объект класса Track и по его данным создавать объект указанного класса, вызывать метод этого класса, передавая в этот метод параметры.

Давайте добавим вызов диспетчера в файл index.php:

```
1  <?php
2      namespace Core;
3
4      error_reporting(E_ALL);
5      ini_set('display_errors', 'on');
6
7      spl_autoload_register(function($class) {
8          // ваша реализация автозагрузки
9      });
10
11     $routes = require $_SERVER['DOCUMENT_ROOT'] .
12         '/project/config/routes.php';
13
14     $track = ( new Router ) -> getTrack($routes, $_SERVER['REQUEST_URI']);
15
16     // Вызов диспетчера:
17
18     $page = ( new Dispatcher ) -> getPage($track);
19
20     ?>
```

Вызов метода `getPage` нашего диспетчера будет вызывать метод `render` контроллера и возвращать то, что вернул вызов этого метода. Как вы уже знаете из предыдущего урока, метод `render` контроллера возвращает объект класса `Page`.

Вот заготовка нашего класса `Dispatcher`:

```
1  <?php
2      namespace Core;
3
4      class Dispatcher
5      {
6          public function getPage(Track $track)
7          {
8              ...код
9              return контроллер -> render(параметры);
10         }
11     }
12  ?>
```

## Задача 105.1

Используя мою заготовку реализуйте описанный класс `Dispatcher`. Проверьте его работу. При затруднениях посмотрите исходный код в файлах сделанного мною учебного фреймворка.

