

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

урок 106 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 класса View в своем MVC фреймворке

Сейчас мы с вами сделаем класс View, который будет заниматься представлением данных. Он будет получать параметром объект класса Page, а своим результатом возвращать готовый HTML код страницы, который можно будет выводить на экран.

Посмотрим, как мы будем использовать класс View в файле 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  $track = ( new Router($routes) ) -> get
15      Track($_SERVER['REQUEST_URI']);
16  $ page  = ( new Dispatcher )      -> getPage($tr
17      ack);
18
19  echo (new View) -> render($page); // вот так
20      используем класс View
21  ?>
```

Структура кода класса View будет иметь следующий вид:

```
1  <?php
2  namespace Core;
3
4  class View
5  {
6      public function render(Page $page) {
7          return $this->renderLayout($page, $this->ren
8              derView($page));
9      }
10
11      private function renderLayout(Page $page, $content) {
12
13      }
14
15      private function renderView(Page $page) {
16
17      }
18  }
19  ?>
```



# Метод renderView

Метод renderView будет получать файл представления и подставлять в него значения переменных. Это делается хитрым образом. Как вы знаете, переменные, которые используются в файле с представлением, содержатся в свойстве data объекта класса Page.

Эти переменные представляют собой ассоциативный массив. Нам нужно превратить этот массив в настоящие переменные, а затем подключить файл с представлением через include. В этом случае указанные доступные в этом файле переменные получают свое значение и на выходе мы получим просто HTML код уже с подставленными значениями переменных.

Для того, чтобы преобразовать массив в переменные, используем специальную функцию extract:

```
1  <?php
2  private function renderView(Page $page) {
3      $viewPath = $_SERVER['DOCUMENT_ROOT'] . "
        /project/views/{$page->view}.php";
4
5      if (file_exists($viewPath)) {
6          ob_start();
7          $data = $page->data;
8          extract($data); // массив в переменные
9          include $viewPath; // подключаем файл с п
            редставлением
10         return ob_get_clean();
11     }
12 }
13 ?>
```

Команды **ob\_start** и **ob\_get\_clean** скорее всего вам не знакомы. В данном случае они используются в качестве специального приема, чтобы результат инклюда не сразу вываливался на экран, а записывался в переменную (в данном случае мы его возвращаем через return).

# Метод renderLayout

Давайте теперь сделаем метод renderLayout. Этот метод будет брать файл лэаута и подставлять в него значение переменных \$title и \$content (она будет передаваться параметром метода и будет представлять собой результат работы метода renderView):

```
1  <?php
2  private function renderLayout(Page $page, $content) {
3      $layoutPath = $_SERVER['DOCUMENT_ROOT'] .
        "/project/layouts/{$page->layout}.php";
4
5      if (file_exists($layoutPath)) {
6          ob_start();
7          $title = $page->title;
8          include $layoutPath; // тут будут доступны пер
            еменные $title и $content
9          return ob_get_clean();
10     }
11 }
12 ?>
```

# Итоговый код

Давайте соберем весь наш код вместе:

```
1  <?php
2  namespace Core;
3
4  class View
5  {
6      public function render(Page $page) {
7          return $this->renderLayout($page, $this->ren
            derView($page));
```

```
8      }
9
10     private function renderLayout(Page $page, $content) {
11         $layoutPath = $_SERVER['DOCUMENT_ROOT'] .
12             "/project/layouts/{$page->layout}.php";
13
14         if (file_exists($layoutPath)) {
15             ob_start();
16             $title = $page->title;
17             include $layoutPath;
18             return ob_get_clean();
19         }
20
21     private function renderView(Page $page) {
22         $viewPath = $_SERVER['DOCUMENT_ROOT'] . "
23             /project/views/{$page->view}.php";
24
25         if (file_exists($viewPath)) {
26             ob_start();
27             $data = $page->data;
28             extract($data);
29             include $viewPath;
30             return ob_get_clean();
31         }
32     }
33     ?>
```

## Задача 106.1

Разберите приведенный код класса View. Затем самостоятельно, не подсматривая в мой код, реализуйте такой же класс. Проверьте его работу.

