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:

```php
<?php
  namespace Core;

  error_reporting(E_ALL);
  ini_set('display_errors', 'on');

  spl_autoload_register(function($class) {
    // ваша реализация автозагрузки
  });

  $routes = require $_SERVER['DOCUMENT_ROOT'] .
     '/project/config/routes.php';

  $track = ( new Router($routes) ) -> get
    Track($_SERVER['REQUEST_URI']);
  $ page  = ( new Dispatcher )      -> getPage($tr
    ack);

  echo (new View) -> render($page); // вот так
     используем класс View
?>
```

Структура кода класса View будет иметь следующий вид:

```php
<?php
  namespace Core;

  class View
  {
    public function render(Page $page) {
      return $this->renderLayout($page, $this->ren
        derView($page));
     }

    private function renderLayout(Page $page, $content) {

    }

    private function renderView(Page $page) {

    }
  }
?>
```

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 получат 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:

```php
<?php
  private function renderView(Page $page) {
    $viewPath = $_SERVER['DOCUMENT_ROOT'] . "
      /project/views/{$page->view}.php";

    if (file_exists($viewPath)) {
      ob_start();
        $data = $page->data;
        extract($data);    // массив в переменные
        include $viewPath; // подключаем файл с п
          редставлением
      return ob_get_clean();
    }
  }
?>
```

Команды **ob_start** и **ob_get_clean** скорее всего вам не знакомы. В данном случае они используются в качестве специального приема, чтобы результат инклуда 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 лэйаута и подставлять в него значение переменных $title и $content (она будет передаваться параметром метода и будет представлять собой результат работы метода renderView):

```php
<?php
  private function renderLayout(Page $page, $content) {
    $layoutPath = $_SERVER['DOCUMENT_ROOT'] .
       "/project/layouts/{$page->layout}.php";

    if (file_exists($layoutPath)) {
      ob_start();
        $title = $page->title;
        include $layoutPath; // тут будут доступны пер
          еменные $title и $content
      return ob_get_clean();
    }
  }
?>
```

Итоговый код

Давайте соберем весь наш код вместе:

```php
<?php
  namespace Core;

  class View
  {
    public function render(Page $page) {
      return $this->renderLayout($page, $this->ren
        derView($page));
```

```
    }

    private function renderLayout(Page $page, $content) {
      $layoutPath = $_SERVER['DOCUMENT_ROOT'] .
         "/project/layouts/{$page->layout}.php";

      if (file_exists($layoutPath)) {
        ob_start();
          $title = $page->title;
          include $layoutPath;
        return ob_get_clean();
      }
    }

    private function renderView(Page $page) {
      $viewPath = $_SERVER['DOCUMENT_ROOT'] . "
        /project/views/{$page->view}.php";

      if (file_exists($viewPath)) {
        ob_start();
          $data = $page->data;
          extract($data);
          include $viewPath;
        return ob_get_clean();
      }
    }
  }
?>
```

Задача 106.1

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