Разработка роутера в своем MVC фреймворке на PHP

Теперь вам нужно разработать *роутер*. Он представляет собой класс, который будет брать массив роутов, брать запрошенный URL, и определять, какой из роутов соответствует данному урл.

После нахождения соответствующего роута наш класс должен получить части URL, соответствующие параметрам роута.

Пусть своим результатом наш роутер возвращает объект класса Track, содержащего имя контроллера, который должен быть вызван на данный запрос, имя действия, и параметры из URL во).

Пусть наш класс Track имеет свойства $controller, $action и $params, доступные только для чтения:

```
<?php
  namespace Core;

  class Track
  {
    private $controller;
    private $action;
    private $params;

    public function __construct($controller, $ac
      tion, $params)
     {
      $this->controller = $controller;
      $this->action = $action;
      $this->params = $params;
    }

    public function __get($property)
    {
      return $this->$property;
    }
  }
?>
```

Пример

Для примера, пусть в адресной строке вбито /test/1/2/. Пусть у нас есть роут, соответствующий этому адресу:

```
<?php
  new Route('/test/:var1/:var2/', 'test', 'index');
?>
```

Это значит, что имя контроллера будет test, имя действия - index, а массив параметров будет следующий: ['var1' => 1, 'var2' => 2].

Цель данного урока - написать класс Router, возвращающий объект класса Track. Остальное нас пока не касается. Давайте приступим к написанию этого класса.

Использование роутера

Вспомним текущее содержимое файла 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';
?>
```

Пусть теперь далее в index.php мы хотим использовать наш роутер следующим образом:

```php
<?php
  $router = new Router();
  $track  = $router->getTrack($routes, $_SERVER['RE
    QUEST_URI']);
?>
```

Можно переписать более компактно:

```php
<?php
  $track = ( new Router ) -> getTrack($ro
    utes, $_SERVER['REQUEST_URI']);
?>
```

Реализация роутера

Давайте теперь напишем заготовку класса Router в соответствии с нашими вызовами:

```php
<?php
  namespace Core;

  class Router
  {
    private $routes;

    public function getTrack($routes, $uri)
    {
      // тут будет код
    }
  }
?>
```

В методе getTrack мы должны определить, какой из роутов соответствует данному $uri. Для этого нужно перебрать наш массив с роутами циклом:

```php
<?php
  namespace Core;

  class Router
  {
    public function getTrack($routes, $uri)
    {
      foreach ($routes as $route) {
        // проверка $uri и $route
      }
    }
  }
?>
```

Если какой-то роут соответствует URI, мы должны получить из этого URI значения параметров роута и вернуть объект класса Track:

```php
<?php
  namespace Core;

  class Router
  {
    public function getTrack($routes, $uri)
    {
      foreach ($routes as $route) {
        if (проверка соответствия роута и URI) {
          $params = ; // нужно получить параметры из uri
          return new Track($route->controller, $route->action, $params);
        }
      }

      return new Track('error', 'notFound'); // если ни один роут не подойдет
    }
  }
?>
```

Задача 103.1

Скопируйте код моего класса Track и разместите его в файле */core/Track.php*.

Задача 103.2

Скопируйте мою заготовку класса Router и разместите его в файле */core/Router.php*.

Задача 103.3

Реализуйте описанный класс Router, своим результатом возвращающий объект класса Track. Если будете испытывать затруднения (что весьма вероятно), подсмотрите в исходный код фреймворка, по которому вы изучали использование MVC. Там в классе Router вы найдете недостающую часть реализации и мои комментарии к ней.