4. Контроллеры

Вызов действия контроллера

В предыдущих уроках в результате срабатывания роута вызывалась ***анонимная функция*** (callback). Такой подход используется, как правило, в обучающих целях. Вместо того, чтобы определять логику обработки запросов ***в виде замыканий в файлах маршрутов***, вы можете организовать её с помощью ***классов контроллеров***.

Контроллеры могут группировать связанную с обработкой HTTP-запросов логику в отдельный класс.

> ***Контроллеры*** хранятся в директории **App/Http/Controllers**

Логика обработки запроса просто ***обязана*** быть выполнена с помощью контроллеров. Иначе, зачем нужен фреймворк?

Для этих целей в файлах маршрутов принято указывать ***контроллер*** и ***действие*** (action) этого контроллера, которое выполнится по срабатыванию ***роута*** (другими словами действие контроллера это функция-обработчик маршрута).

Укажем, к примеру, что по обращению к URI '/about' выполнится действие acAbout, контроллера MainController.

Файл **ex_1/web.php**. Вызов контроллера

```
// регистрируем маршрут http://host/about

// обработчик: контроллер - MainController, действие - acAbout

Route::get ('/about', [MainController::class, 'acAbout']);
```

где:

- **/about** - маршрут, который будет обрабатываться контроллером
- **MainController** - контроллер
- **acAbout** - функция-обработчик(действие)роута в контроллере

Вывод в браузер из ***функции замыкания маршрутизатора*** теперь перенесли в ***действие контроллера*** и выглядеть это будет так:

Файл **ex_1/MainController.php**. Обработка роута

```
// обработчик маршрута http://host/about
public function acAbout () {
     echo "<h1>Страница: О нас</h1>";
}
```

Передача в действие контроллера одного параметра

В уроке "Маршруты с параметрами" мы рассмотрели возможность встроить в маршрут некоторую его изменяемую часть - ***параметр***. В качестве параметра можно использовать любой набор символов.

Параметр маршрута необходимо передать в ***действие*** контроллера. В противном случае будет сгенерирована ошибка. Исключения составляют ***необязательные параметры*** маршрутов.

Файл **ex_2/web.php**. Определение параметра в маршруте

```
// регистрируем маршрут http://music/team/{параметр}/albums
// пример: http://host/music/team/4/albums
Route::get ('/music/team/{idTeam}/albums',
[MainController::class, 'acAlbums']);
```

Файл **ex_2/MainController.php**. Передача параметра в действие контроллера

```
// обработчик маршрута http://music/team/{параметр}/albums
public function acAlbums ($par) {
     echo "<h1>Страница: Альбомы</h1>";
     echo "<h2>Параметр маршрута: $par</h2>";
}
```

Передача в действие контроллера двух параметров

Передача двух параметров ничем не отличается от передачи одного параметра. При этом нельзя забывать, что никакой ***связи между именем параметра в маршруте и именем параметра в функции нет***. Важен порядок следования.

Файл **ex_3/web.php**. Определение двух параметров маршрута

```
// регистрируем маршрут:
// http://host/music/team/{парметр1}/album/{параметр2}
// пример: http://host/music/team/21/album/3
Route::get ('/music/team/{idTeam}/album/{idAlbum}',
[MainController::class, 'acAlbum']);
```

Файл **ex_3/MainController.php**. Передача параметров в действие контроллера

```
// обработчик маршрута:
// http://host/music/{парметр1}/album/{параметр2}
public function acAlbum ($par1, $par2) {
     echo "<h1>Страница: Вывод информации об альбоме</h1>";
     echo "<h3>Идентификатор группы: $par1</h3>";
```

```
    echo "<h3>Идентификатор альбома: $par2</h3>";
}
```

Формирование данных

Маршрутизатор вызвал действие контроллера, при этом посредством ***параметров маршрута*** либо ***строкой запроса*** передал ему какие то входные данные. Далее контроллер, используя входные данные, реализует логику работы приложения и вывод ***обработанных данных*** в браузер.

Хотя такая цепочка и возможна, выводить данные в браузер, это не функция контроллера, это обязанность представления (урок 1. Технология MVC). А вот обязанность контроллера - ***сформировать данные для передачи представлению***.

Данные могут быть представлены в виде:

- простой переменной
- массива
- объекта

Массивы и Объекты - основные структуры данных, которые вы будете передавать представлению при разработке приложений.

Файл **ex_5/web.php**. Регистрация маршрута

```
// регистрируем маршрут http://host/music/team/{параметр}
// пример: http://host/music/team/11
Route::get ('/music/team/{idTeam}', [MainController::class, 'acTeam']);
```

Файл **ex_5/MainController.php**. Обработка маршрута, формирование данных

```
// обработчик маршрута http://host/music/team/{параметр}
public function acTeam ($id) {
     // получили id группы, по имеющемуся id можно
     // извлечь из базы различную информацию
     // например название группы
     $name = "Deep Purple";
     dump ($name); // простая переменная
     // или название группы, топовый альбом,
     // исполнителей топ-альбома
     $team["name"] = "Deep Purple";
     $team["album"] = "Deep Purple in Rock";
     $team["composition"] = ["Иэн Гиллан", "Ричи Блэкмор",
     "Роджер Гловер", "Джон Лорд Иэн Пэйс"];
     dump($team); // массив
     // а можем представить все это в виде объекта
     dump ((object)$team); // объект
}
```

Файлы раздаточного материала

ex_1/ - вызов действия контроллера.

ex_2/ - передача в действие контроллера одного параметра.

ex_3/ - передача действие контроллера двух параметров.

ex_4/ - передача в действие контроллера трех параметров.

ex_5/ - формирование данных разных типов.