6. Представления

Вывод представления

Представления (**views**), они же ***макеты***, они же ***шаблоны***, содержат HTML-код, передаваемый вашим приложением. Это удобный способ разделения логики ***получения*** и логики ***отображения*** информации.

> Представления находятся в каталоге **resources/views**.

Laravel использует ***шаблонизатор Blade***, поэтому представления имеют два расширения: первое **.php** и второе **.blade**. Второе расширение показывает, что мы в нашем файле можем ***использовать команды шаблонизатора***.

Примеры именования представлений:

- lesson.blade.php
- team.blade.php

Для того, чтобы получить представление, используется функция **view**. Эта функция параметром принимает ***название файла представления*** и возвращает его текст.

```
public function acTest () {
    // ...
    return view("page");
}
```

где: **page** - имя файла представления **page.blade.php** директории **views**.

> ***Важно***: В функцию **view** нужно передавать ***только имя файла***, без расширений

Приведем пример обработки запроса от маршрутизатора до представления.

Файл **ex_1/web.php**. Регистрация маршрута

```
// регистрируем маршрут http://host/music
Route::get ('/music', [MainController::class, 'acMusic']);
```

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

```
// обработчик маршрута http://host/music
public function acMusic () {
     // получаем вывод представления
     return view("page");
}
```

Контроллер передает какие-то данные в представление, а представление ***оформляет данные нужным HTML кодом***. Пока наш контроллер не передал никаких данных, а просто попросил представление вывести свой HTML код - так тоже бывает.

Как получить представление мы разобрались. Теперь самое главное - рассмотрим, как ***передать данные из контроллера в представление***.

Передача простых переменных

Для передачи переменных предназначен ***второй параметр функции view***. В этот параметр мы можем передать, например, ***ассоциативный массив***.

Файл **ex_2/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_2/MainController.php**. Обработка маршрута

```
// 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
     // создаем простую переменную
     $name = "Deep Purple";
     // передаем переменную в представление
     return view("page", ["name" => $name]);
}
```

Файл **ex_2/page.blade.php**. Вывод переменной в представлении

```
<body>
     <h1>{{ $name }}</h1>
</body>
```

Переменные, которые передаются в представление из действия контроллера, обычно выводятся внутри каких-нибудь тегов. Переменная представления помещается в двойные фигурные скобки **{{** $var **}}**.

Передача массивов

Рассмотрим, каким образом в представление можно передать и вывести ***массив***. Передавать будем ***ассоциативный массив***, индексный массив передается аналогично.

Здесь и далее данные в представление будем передавать альтернативным способом, не через ***второй параметр функции view***, а *используя* ***цепочку методов***:

```
return view("page")->with([ ... ]);
```

Файл **ex_3/MainController.php**. Обработка маршрута

```
// 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, топовый альбом, исполнителей топ-альбома
     // формируем массив
     $team["name"] = "Deep Purple";
     $team["album"] = "Deep Purple in Rock";
     $team["composition"] = ["Иэн Гиллан", "Ричи Блэкмор",
     "Роджер Гловер", "Джон Лорд Иэн Пэйс"];
     // передаем массив в представление
     return view("page")->with(["team" => $team]);
}
```

Файл **ex_3/page.blade.php**. Вывод массива в представлении

```
<body>
     <h1>Группа: {{ $team["name"] }}</h1>
```

```
    <h2>Альбом: {{ $team["album"] }}</h2>
    <h2>Состав:</h2>
    <ul>
        <li>{{ $team["composition"][0] }}</li>
        <li>{{ $team["composition"][1] }}</li>
        <li>{{ $team["composition"][2] }}</li>
        <li>{{ $team["composition"][3] }}</li>
    </ul>
</body>
```

Передача объектов

Рассмотрим, каким образом в представление можно передать и вывести ***объект***.

Файл **ex_4/MainController.php**. Обработка маршрута

```
// 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, топовый альбом, исполнителей топ-альбома
    // формируем массив
    $team["name"] = "Deep Purple";
    $team["album"] = "Deep Purple in Rock";
    $team["composition"] = ["Иэн Гиллан", "Ричи Блэкмор",
    "Роджер Гловер", "Джон Лорд Иэн Пэйс"];
    // в представление передаем объект
    return view("page")->with(["team" => (object)$team]);
```

```
}
```

Файл **ex_4/page.blade.php**. Вывод объекта в представлении

```
<body>
     <h1>Группа: {{ $team->name }}</h1>
     <h2>Альбом: {{ $team->album }}</h2>
     <h2>Состав:</h2>
     <ul>
          <li>{{ $team->composition[0] }}</li>
          <li>{{ $team->composition[1] }}</li>
          <li>{{ $team->composition[2] }}</li>
          <li>{{ $team->composition[3] }}</li>
     </ul>
</body>
```

Вывод неэкранированных данных

Всякий раз, когда пользователь отправляет текст на ваш сайт (имя, логин, пароль или любая другая информация), вы должны быть уверенны, что ваши скрипты не содержат дыры с точки зрения безопасности сайта, которые злоумышленники могут использовать для взлома.

Если все же нужно ***получать от пользователя данные***, то обязательно используйте функцию **htmlentities**, чтобы предотвратить запуск HTML-кода или скриптов, которые могут быть потенциально опасны и представлять угрозу вашему сайту.

Важно. **Htmlentities()** — преобразует все возможные ***символы*** в

> соответствующие ***HTML-сущности***

По умолчанию Blade-оператор **{{ }}** автоматически отправляется через PHP-функцию **htmlentities()**. Если вы не хотите экранировать данные, используйте такой синтаксис:

```
{!! $name !!}
```

Комментарии в представлениях

В представлениях допускается использование привычных HTML-комментариев:

```
<!-- это обычный HTML комментарий -->
```

Blade также позволяет определить комментарии в создаваемых представлениях. Но в отличие от HTML-комментариев, ***Blade-комментарии*** не включаются в HTML-код, возвращаемый вашим приложением:

```
{{-- этого комментария не будет в итоговом HTML --}}
```

Файлы раздаточного материала

ex_1/ - вывод представления.

ex_2/ - передача в представление простой переменной.

ex_3/ - передача в представление массива.

ex_4/ - передача в представление объекта.

ex_5/ - передача в представление смешанных данных.