Публикация на тему

Artisan

Artisan - интерфейс командной строки Laravel

Автор
Михалькевич Александр Викторович

Публикация

Наименование Artisan
Автор А.В.Михалькевич
Специальность Artisan - интерфейс командной строки Laravel,
Анотация
Anotation in English
Ключевые слова
Количество символов 15186

Содержание

Введение

1 Основы использования

Artisan - название интерфейса командной строки, входящей в состав Laravel. Он предоставляет полезные команды для использования во время разработки приложения. Работает на основе мощного компонента SymfonyConsole.

Чтобы вывести все доступные команды Artisan, можно воспользоваться командной list:

Вывод всех команд Artisan. Листинг 1.1

php **artisan list**

Доступные команды:

make:command — создаёт новый класс команды

make:console — создаёт новую команду Artisan

make:controller — создаёт новый класс контроллера ресурса

make:event — создаёт новый класс события

make:middleware — создаёт новый класс промежуточного ПО

make:migration — создаёт новый файл миграции

make:model — создаёт новый класс модели Eloquent и миграцию

make:provider — создаёт новый класс поставщика услуг

make:request — создаёт новый класс запроса формы

event:generate — генерирует пропущенные события и обработчики

Каждая команда также включает и инструкцию, которая отображает и описывает доступные аргументы и опции для команды. Чтобы её вывести, необходимо добавить слово help перед командой:

Просмотр подсказок для текущей команды. Листинг 1.2

php artisan **help** migrate

Возможно также указать среду, в которой будет выполнена команда при помощи опции --env:

Использование среды. Листинг 1.3

php artisan migrate **--env=local**

Чтобы определить текущую версию Laravel, можно воспользоваться опцией--version

Определение текущей версии Laravel. Листинг 1.4

php artisan --version

2 Разработка команд artisan

В дополнение к командам, предоставляемых фрэймворком, artisan предоставляет возможность создавать свои собственные команды, необходимые разрабатываемому приложению. Свои команды можно хранить как в директории app/Console, так и самостоятельно выбирать место для хранения, прежде убедившись, что команды будут автоматически загружены, основываясь на настройках composer.json.

Для создания новой команды воспользуемся командой artisan-а make:console, которая сгенерирует макет класса:

Создание нового класса команды. Листинг 2.1

php artisan **make:console** FooCommand

Команда выше сгенерирует класс app/Console/FooCommand.php.

Создавая команду, опция --command может быть использована для назначения имени команды в консоли:

Использование опции --command. Листинг 2.2

```
php artisan make:console AssignUsers --command=users:assign
```

Как только команда будет сгенерирована, необходимо заполнить свойства класса name и description, которые будут использованы при отображении команды в списке.

Метод fire будет вызван как только команда будет запущена. Вы можете поместить в этот метод любую логику.

В методах getArguments и getOptions можно определить любые аргументы или опции, которые будет принимать команда. Оба этих метода возвращают массив команд, описываемых списком полей массива.

Массив, определяющий аргументы, выглядит так:

Массив определяющий аргументы. Листинг 2.3

```
array($name, $mode, $description, $defaultValue)
```

Аргумент mode может принимать одно из следующих значений: InputArgument::REQUIRED (обязательный) или InputArgument::OPTIONAL (необязательный).

Массив, определяющий опции, выглядит следующим образом:

Массив, определяющий аргументы. Листинг 2.4

```
array($name, $shortcut, $mode, $description, $defaultValue)
```

Для опций аргумент mode может быть: InputOption::VALUE_REQUIRED (значение обязательно), InputOption::VALUE_OPTIONAL (значение необязательно), InputOption::VALUE_IS_ARRAY (значение - это массив), InputOption::VALUE_NONE (нет значения).

Режим VALUE_IS_ARRAY обозначает, что этот переключатель может быть использован несколько раз при вызове команды:

Вызов команды с использованием режима VALUE_IS_ARRAY. Листинг 2.5

```
php artisan foo --option=bar --option=baz
```

Значение VALUE_NONE означает, что опция просто используется как "переключатель":

Вызов команды с использованием опции переключателя. Листинг 2.6

```
php artisan foo --option
```

Во время выполнения команды, очевидно, потребуется получать значения переданных аргументов и опций. Для этого можно воспользоваться методами argument и option:

Вывод переданных значений аргументов и опций. Листинг 2.7

```
//Получение значения аргумента команды
$value = $this->argument('name');
//Получение всех аргументов
$arguments = $this->argument();
//Получение значения опции команды
$value = $this->option('name');
//Получениевсехопций
$options = $this->option();
```

Для вывода данных в консоль вы можете использовать методы info (информация), comment (комментарий), question (вопрос) и error (ошибка). Каждый из этих методов будет использовать цвет по стандарту ANSI, соответствующий смыслу метода.

Вывод команды. Листинг 2.8

```
//Выводинформациивконсоль
$this->info('Display this on the screen');
//Вывод сообщений об ошибке в консоль
$this->error('Something went wrong!');
```

Для обеспечения пользовательского ввода, можно воспользоваться методами ask иconfirm.

Пользовательский ввод. Листинг 2.9

```
//Попросить пользователя ввести данные:
$name = $this->ask('What is your name?');
//Попросить пользователя ввести секретные данные:
$password = $this->secret('Whatisthepassword?');
Попросить пользователя подтвердить что-то:
if ($this->confirm('Doyouwishtocontinue? [yes|no]'))
{
//
}
```

Также можно указать ответ по умолчанию для метода confirm. Это должно быть true или false:

Ответ по умолчанию для метода confirm. Листинг 2.10

```
$this->confirm($question, true);
```

Как только команда будет готова, нужно зарегистрировать её в Artisan-е, чтобы она была доступна для использования. Обычно это делается в файле app/Providers/ArtisanServiceProvider.php. В этом же файле команду можно зарегестрировать в контейнере IoC. Для регистрации имеется специальный метод commands. По умолчанию, образец регистрации команды включен в сервис-провайдер. Например:

Пример регистрации команды. Листинг 2.11

```
$this->app->bindShared('commands.inspire', function()
{
return new InspireCommand;
});
```

Как только команда зарегистрирована в IoC контейнере, воспользуемся commands сервис-провайдера, чтобы сделать её доступной в artisan-е. Необходимо передать название, использованное при регистрации команды в IoC контейнере:

Передача названия, использованное при регистрации команды. Листинг 2.12

```
$this->commands('commands.inspire');
```

Иногда может потребоваться вызвать другую команду из вашей команды. Это можно сделать, используя метод call:

Вызов другой команды из текущей команды. Листинг 6.17

```
$this->call('command:name', array('argument' => 'foo', '--option' => 'bar'));
```

Заключение

Список использованных источников

Приложения