

# 5. Artisan

## **Введение**

---

В предыдущем уроке мы разбирали контроллеры. Для экспериментов вам был предложен файл контроллера **MainController.php**. Но что, если бы мы его не предложили? Вариант набирать вручную - совсем не вариант). Каждый раз набирать с нуля файлы контроллеров, моделей, посредников не очень удобно (да совсем неудобно). В процессе разработки приложения вам может понадобиться не один файл, например контроллера, создание которого хотелось бы автоматизировать.

В этом уроке разберем средство автоматизации создания полезных шаблонов файлов. Это средство называется artisan.

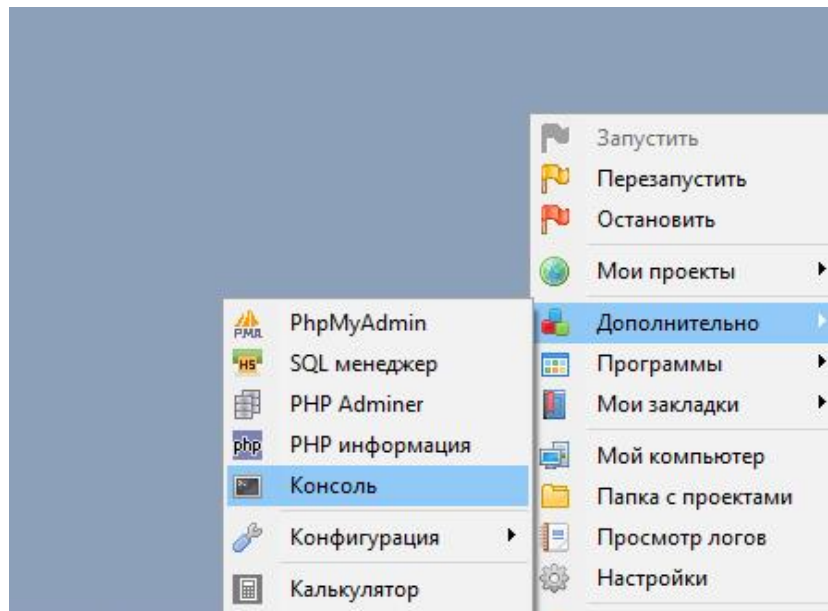
**Artisan** – это ***интерфейс командной строки*** или CLI-приложение, которое позволяет выполнять целое множество полезных команд при разработке веб-приложения на Laravel. Artisan является отличным помощником веб-разработчика, который в значительной степени может сократить некоторые рутинные операции при разработке.

Файл с названием **artisan** физически находится в ***корневой директории сайта Laravel*** (в нашем случае папка host) . При работе в ***консоли*** будем обращаться к нему через **php** с указанием имени этого файла:

```
php artisan.
```

Для демонстрации возможностей запустим **OpenServer**, далее кликаем правой кнопкой по его флажку-пиктограмме и выбираем

**Дополнительно - Консоль.**



В открывшемся окне перейдем в корневую директорию нашего учебного проекта - папку **host**.

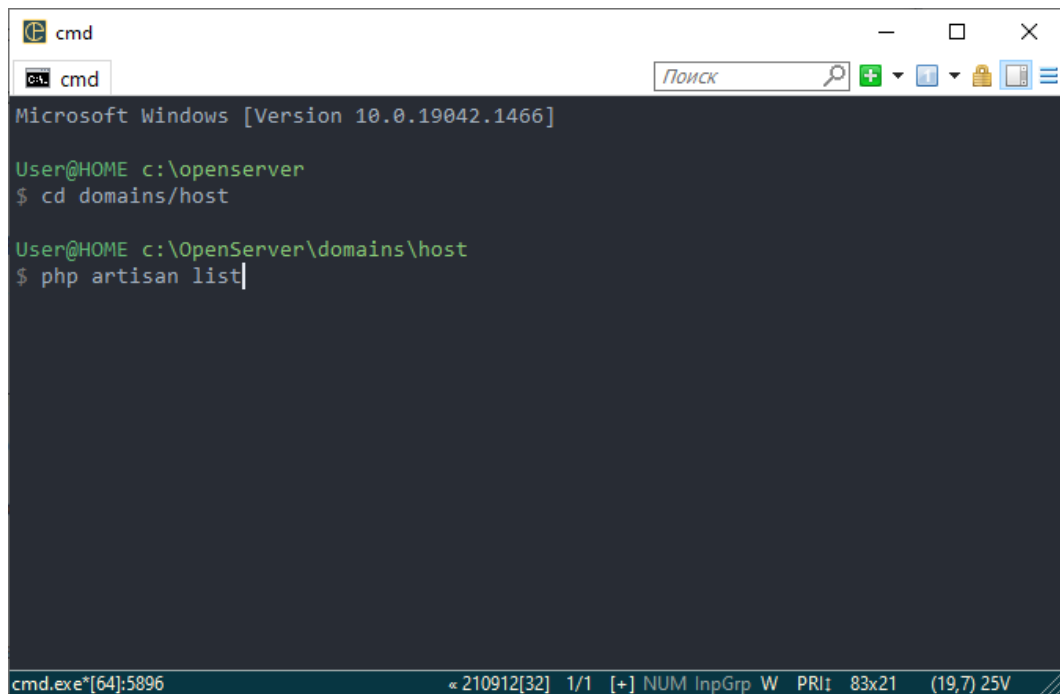
```
Microsoft Windows [Version 10.0.19042.1466]
User@HOME c:\openserver
$ cd domains/host

User@HOME c:\OpenServer\domains\host
$ |
```

Для примера, рассмотрим список всех доступных artisan-команд, вызвав таким образом справку по интерфейсу.

Просмотр списка доступных команд:

```
php artisan list
```



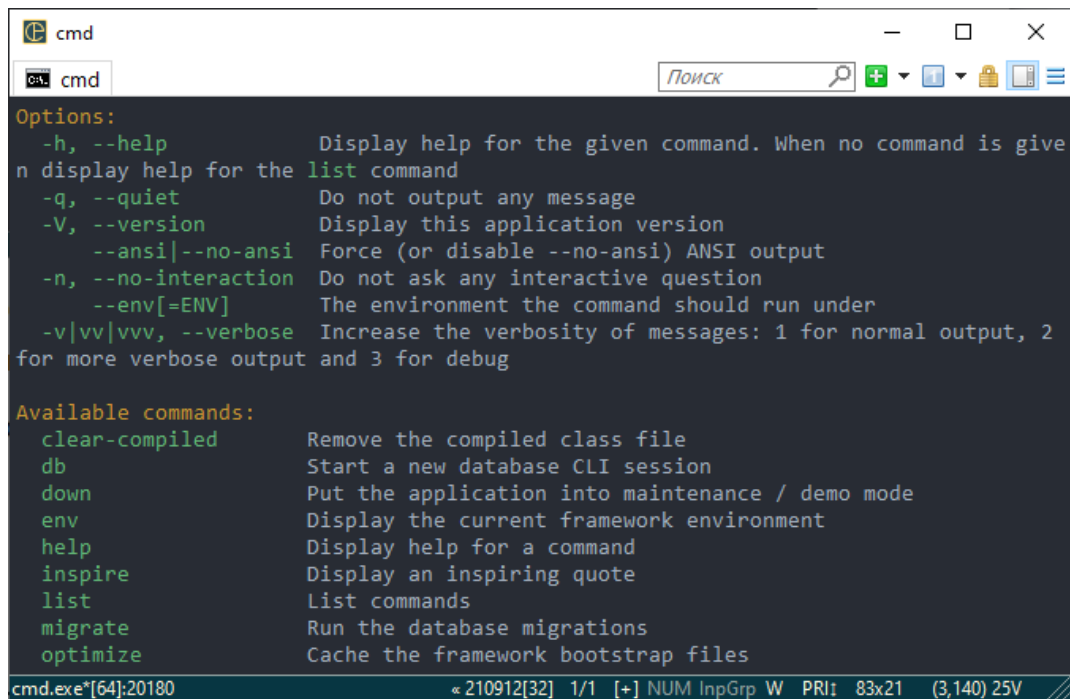
```
cmd
Microsoft Windows [Version 10.0.19042.1466]

User@HOME c:\openserver
$ cd domains/host

User@HOME c:\OpenServer\domains\host
$ php artisan list
```

Список команд целое множество. Однако на практике большинство из них может ни разу не пригодятся.

**Совет:** В языках программирования списки команд, списки функций, списки аргументов (и пр.) приводятся, как правило, не для того чтобы их немедленно учили. Приводят для того, чтобы начинающий разработчик знал, что эта информация есть и ее легко получить в случае необходимости.



```
cmd
cmd
Поиск

Options:
-h, --help            Display help for the given command. When no command is give
n display help for the list command
-q, --quiet           Do not output any message
-V, --version         Display this application version
--ansi|--no-ansi     Force (or disable --no-ansi) ANSI output
-n, --no-interaction Do not ask any interactive question
--env[=ENV]          The environment the command should run under
-v|vv|vvv, --verbose Increase the verbosity of messages: 1 for normal output, 2
for more verbose output and 3 for debug

Available commands:
clear-compiled       Remove the compiled class file
db                   Start a new database CLI session
down                 Put the application into maintenance / demo mode
env                  Display the current framework environment
help                 Display help for a command
inspire              Display an inspiring quote
list                 List commands
migrate              Run the database migrations
optimize             Cache the framework bootstrap files

cmd.exe*[64]:20180      * 210912[32]  1/1  [+] NUM InpGrp W  PRI:  83x21  (3,140) 25V
```

## Некоторые *artisan*-команды

---

Приведем пример команд, которые могут оказаться полезными уже на этапе погружения в технологию.

### Создание файла контроллера

- `php artisan make:controller ContactController`

Будет создана папка **app/Http/Cotrollers** (если не существует) и файл с классом контроллера **ContactController.php**.

### Создание файла валидации данных

- `php artisan make:request ContactRequest`

Будет создана папка **app/Http/Requests** (если не существует), в ней будет создан файл **ContactRequest.php**.

### Создание файла — класс модели

- `php artisan make:model Contact`

В директории **app\Models** будет создан файл **Contact.php**.

### Создание класса посредника

- `php artisan make:middleware CheckAge`

Будет создана папка **app\Http\Middleware** (если не существует), в ней будет создан файл **CheckAge.php**.

### Выполнение миграции базы данных

- `php artisan migrate`

Файлы миграции базы данных располагаются в отдельной директории **/database/migrations/**. Эта команда запускает миграцию данных из этих файлов в базу данных.

### Очистка кэша Laravel

Время от времени кэш Laravel полезно чистить. Особенно это актуально при активном тестировании приложения и частой смене файлов представлений и CSS-стилей.

Но кэшей в Laravel несколько, и вот команды, чтобы очистить их все:

- `php artisan cache:clear`
- `php artisan config:clear`
- `php artisan route:clear`
- `php artisan view:clear`