不管是 Node.js、Python 还是 PHP 的 Web 框架，都提供了通过命令行与应用进行交互的功能，通过这些命令行工具，我们可以完成一些初始化操作，比如创建新应用、执行数据库迁移、或者快速创建类文件等，在 Laravel 中，我们可以通过三种工具实现命令行交互：

* Artisan：Laravel 内置的命令行操作工具集，支持自定义命令；
* Tinker：一个由 [PsySH](https://github.com/bobthecow/psysh) 扩展包驱动的 REPL，允许你通过命令行与整个 Laravel 应用进行交互；
* Laravel 安装器，这个我们在[框架安装部分](https://laravelacademy.org/post/9610.html#toc_1)已经提到过，比较简单，不再单独介绍。

我们首先来介绍 Artisan 及其使用。

**Artisan 命令简介**

我们在前面的教程中已经多次使用过 Artisan 命令，比如创建控制器：

php artisan make:controller PostController

如果你查看应用根目录，会看到一个 artisan 文件，这个就是命令行交互的入口文件，就像 public/index.php 是 Web 请求的入口文件一样。在 artisan 文件中，处理流程会像 Web 请求一样，注册类的自动加载器，初始化容器和异常处理器，获取用户输入，执行处理逻辑，最后发送响应，只不过这一切都是在控制台中完成。我们在执行 php artisan 命令时，artisan 就指的是根目录下的 artisan 文件，所以命令需要在应用根目录下执行，然后 artisan 之后的部门都是作为请求参数被控制台应用获取并处理。

注：Artisan 底层基于 [Symfony Console 组件](https://symfony.com/doc/current/components/console.html)，所以，如果你之前使写过 Symfony Console 命令，会很快熟悉 Artisan 命令的使用。

不同的 Laravel 应用由于安装了不同的扩展包或编写了自定义的 Artisan 命令，所以在当你准备了解一个新应用时，有必要通过 php artisan list 快速浏览该应用支持的所有命令。

**内置 Artisan 命令**

运行 php artisan list 命令后，你会看到新安装的 Laravel 应用默认提供了很多 Artisan 命令，我们不可能把每一个都讲一遍下来（实际上，我们会在每篇教程中分散提到），现在，我们挑几个最基本的命令来看下：

* help：为指定命令提供使用帮助信息，如 php artisan help make:request
* clear-compiled：移除编译过的类文件，比如缓存、Blade视图文件等
* down：将应用切换到维护模式以便查找问题
* up：将应用从维护模式恢复为正常模式
* env：显示应用当前运行环境，如 local、production
* migrate：运行所有数据库迁移
* optimize：优化应用以便提供更好的性能
* serve：在本地 localhost:8000 端口启动 PHP 内置服务器
* tinker：进入 Tinker REPL
* dump-server：启动 dump server 收集 dump 信息
* preset：切换应用前端框架脚手架代码，比如从 Vue 切换到 React

**选项**

在我们继续介绍 Artisan 命令其它内容之前，我们先来看一下在运行 Artisan 命令时可以传入的选项参数：

* -q：禁止所有输出
* -v、-vv、-vvv：命令执行输出的三个级别，分别代表正常、详细、调试
* --no-interaction：不会问任何交互问题，所以适用于运行无人值守自动处理命令
* --env：允许你指定命令运行的环境
* --version：打印当前 Laravel 版本

上述选项可以单独运行，也可以和具体命令一起运行。

**分组命令**

php artisan list 罗列出的其它命令都是被分门别类的，我们不会详细介绍所有命令，大致看一下分组：

* app：只包含 app:name 命令，用于替换应用默认命名空间 App\
* auth：只包含 auth:clear-resets，用于从数据库清除已过期的密码 Token
* cache：应用缓存相关命令
* config：config:cache 用于缓存应用配置，config:clear 用于清除缓存配置
* db：db:seed 用于通过填充器填充数据库（如果编写了填充器的话）
* event：event:generate 用于根据注册信息生成未创建的事件类及监听器类
* key：key:generate 用于手动设置应用的 APP\_KEY
* make：用于根据模板快速生成应用各种脚手架代码，如认证、模型、控制器、数据库迁移文件等等等，我们会将每个命令穿插在相应教程中介绍
* migrate：数据库迁移相关命令（数据库教程中会详细介绍）
* notifications：notifications:table 用于生成通知表
* optimize：optimize:clear 用于清除缓存的启动文件
* package：package:discover 用于重新构建缓存的扩展包 manifest
* queue：队列相关命令（队列教程中会详细介绍）
* route：路由相关命令，route:cache 和 route:clear 分别用于缓存路由信息和清除路由缓存，route:list 用于列出应用所有路由信息
* schedule：调度任务相关命令（调度任务教程中会介绍）
* session：对于数据库驱动的 Session，我们通过 session:table 生成 sessions 数据表
* storage：storage:link 生成一个软链 public/storage 指向 storage/app/public
* vendor：vendor:publish 用于发布扩展包中的公共资源
* view：view:cache 用于编译应用所有 Blade 模板，view:clear 用于清除这些编译文件

**编写第一个 Artisan 命令**

介绍完系统内置的所有命令之后，我们接下来看看如何编写自定义的 Artisan 命令，这种需求在实际项目开发中还是常有的，比如数据迁移、数据修复、定时任务等。

**创建命令类**

首先，我们要了解专门有一个系统自带命令 make:command 来创建自定义命令文件：

php artisan make:command WelcomeMessage --command=welcome:message

该命令的第一个参数就是要创建的 Artisan 命令类名，还可以传递一个选项参数 --command 用于自定义该命令的名称（不指定的话会系统会根据类名自动生成）。执行完该命令后，会在 app/Console/Commands 目录下创建一个 WelcomeMessage.php 文件：

<?php

namespace App\Console\Commands;

use Illuminate\Console\Command;

class WelcomeMessage extends Command

{

/\*\*

\* 命令名称，在控制台执行命令时用到

\*

\* @var string

\*/

protected $signature = 'welcome:message';

/\*\*

\* 命令描述

\*

\* @var string

\*/

protected $description = 'print welcome message';

/\*\*

\* Create a new command instance.

\*

\* @return void

\*/

public function \_\_construct()

{

parent::\_\_construct();

}

/\*\*

\* 命令具体执行逻辑放在这里

\*

\* @return mixed

\*/

public function handle()

{

//

}

}

**注册命令类**

创建完 Artisan 命令类后，还需要在 app/Console/Kernel.php 中注册才能使用，就像编写好控制器方法后，需要在路由文件中定义路由指向它一样。

打开 app/Console/Kernel.php，将新创建的命令类 WelcomeMessage 添加到 $commands 完成注册：

protected $commands = [

App\Console\Commands\WelcomeMessage::class

];

现在，运行 php artisan list 就可以看到 welcome:message 命令了。

**编写一个简单示例**

由于我们的命令类还没有实现任何实际功能，下面我们来编写一个简单示例，打开 app/Console/Commands/WelcomeMessage.php，Artisan 命令的具体业务逻辑要定义到 handle 方法中，这里我们先简单打印一条欢迎信息：

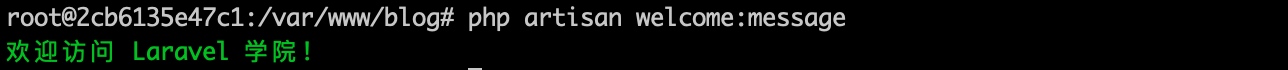
public function handle()

{

$this->info('欢迎访问 Laravel 学院!');

}

这样，在应用根目录下运行 php artisan welcome:message 的话就可以打印上面这条欢迎信息了：



**基于闭包实现 Artisan 命令**

当然，就像我们可以在 routes/web.php 中基于闭包实现简单业务逻辑一样，对于这么简单的命令，我们也可以在 routes/console.php 中基于闭包实现：

Artisan::command('welcome:message\_simple', function () {

$this->info('欢迎访问 Laravel 学院!');

})->describe('打印欢迎信息');

这样我们就可以在命令行运行 php artisan welcome:message\_simple 打印欢迎信息了，效果和上面通过命令类实现的一样。

至此，你应该对 Artisan 命令的使用和编写有了一个初步的概念，下一篇我们将介绍如何通过命令类编写更加复杂的交互命令。