# Laravel框架\_01

# 今日目标14

* 能够使用composer方式部署Laravel框架项目.
* 能够解释laravel框架中的主要目录结构.
* 能够使用php artisan serve方式运行laravel项目.
* 能够说出Laravel框架中定义路由的文件的名称.
* 能够说出Laravel框架中的4种基础路由.
* 能够说出Laravel框架中的控制器所在的目录.
* 能够说出Laravel框架中控制器文件的命名规则.
* 能够使用命令行创建自定义控制器.
* 能够使用请求类进行数据的获取.
* 能够返回一个json数据.
* 能够掌握Laravel框架中return view()方式加载blade视图模板.
* 能够使用compact函数向视图中分配变量.
* 能够掌握在Laravel框架视图模板中foreach方式遍历数据.
* 能够理解模板继承的概念.

# 一、Laravel简介

官方网站：<https://laravel.com/>

中文官网：<https://laravel-china.org/docs/laravel/5.6>

Laravel是一套**简洁、优雅**的PHP Web开发框架(PHP Web Framework)。它可以让你从面条一样杂乱的代码中解脱出来；它可以帮你构建一个完美的网络APP，而且每行代码都可以简洁、富于表达力。

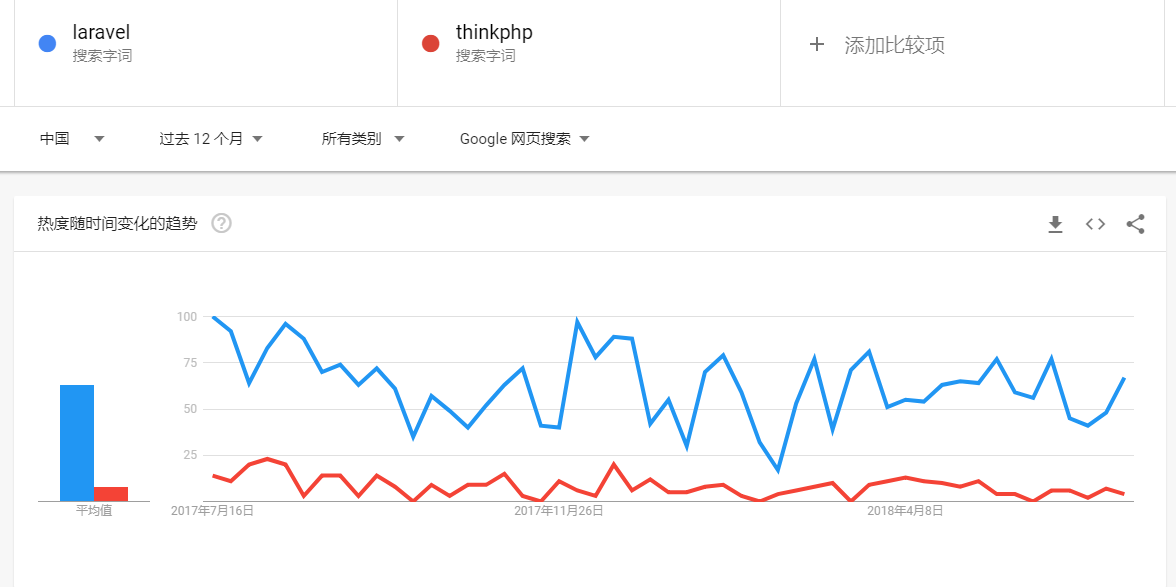
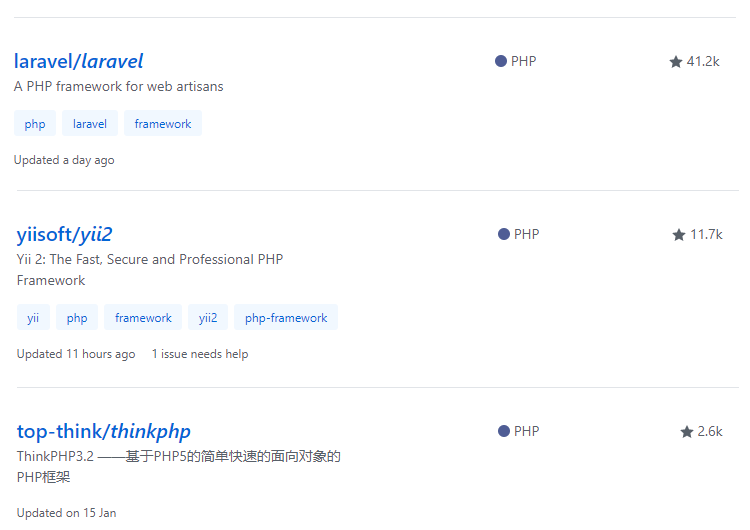
Laravel框架诞生于2011年6月11日

**号称：为 WEB 艺术家创造的 PHP 框架**

**招聘要求**



**框架的占比**



1、能让我们代码编写更改优雅，2、提高收入 3、框架趋势

# 二、Laravel安装

## 2.1、服务器要求

Laravel5.6 对PHP是有一些要求的。

PHP >= 7.1.3

OpenSSL PHP

PHP PDO 扩展

PHP fileinfo 扩展

PHP mcrypt 扩展

PHP Mbstring 扩展

PHP Tokenizer 扩展

PHP XML 扩展

PHP Ctype 扩展

PHP JSON 扩展

一定要把php配置到环境变量中，让我们PHP命令可以在任意的位置可执行。





## 2.2、通过composer安装laravel

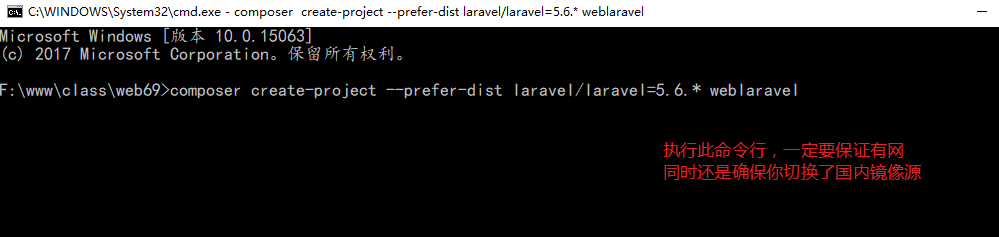
composer create-project --prefer-dist laravel/laravel=5.6.\* weblaravel

create-project 新建项目

--prefer-dist 以压缩的方式下载

laravel/laravel=5.6.\* 创建名称和版本号

weblaravel 下载到本机后，所起的项目名称



切换国内镜像源

composer config -g repo.packagist composer <https://packagist.phpcomposer.com>

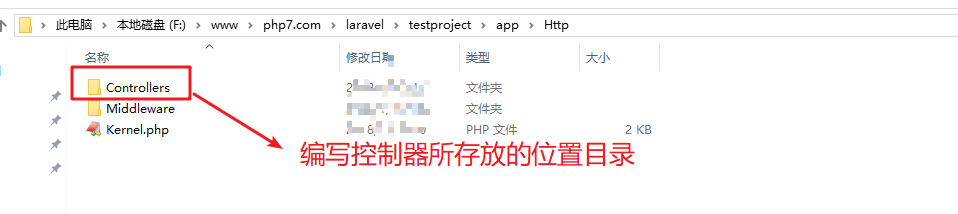
备选

composer config -g repo.packagist composer https://packagist.laravel-china.org

# 三、目录结构

## 3.1、app目录【重要目录】

项目的核心目录，主要用于存放核心代码，你所编写的代码绝大多数也会放到这里，如控制器、模型等。



默认模型存放的位置如下,app根目录下面



但是这种在根目录下面建立模型文档不利于我们后期代码的管理维护，所以我们往往在项目中，手动创建目录，进行目录分隔，即可以在该目录下建立Models目录，如下



## 3.2、bootstrap目录

laravel启动目录，目录包含了框架启动和自动加载设置的文件，autoload.php文件用于自动载入需要的类文件。



## 3.3、config目录【重要目录】

项目的配置目录，主要存放配置文件，比如数据库的配置。



app.php：项目主要配置文件

database.php：针对数据库的配置

## 3.4、database目录

目录包含了数据迁移及填充文件，就是使用文件的方式，来管理数据库，创建一个PHP文件，在文件中设计表结构，运行该文件，完成表的创建。



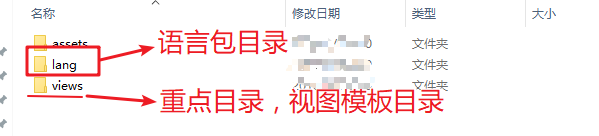
## 3.5、public目录

**laravel项目的web虚拟主机指定的目录，**项目的入口文件和系统的静态资源目录（css,img,js,uploads）**后期使用的外部静态文件（js、css、图片等）都需要放到Public目录下。**



## 3.6、resources目录【重要目录】

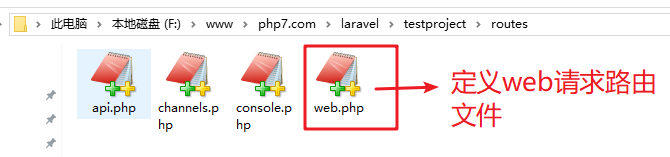
存放视图文件，还有就是语言包文件的目录。



注：laravel中我们的模板引擎为blade模板引擎，后缀为.php，例：【welcome.blade.php】

## 3.7、routes目录【重要目录】

定义路由的目录，laravel中所有的访问请求，都必须事先定义好路由，web.php是定义路由的文件。



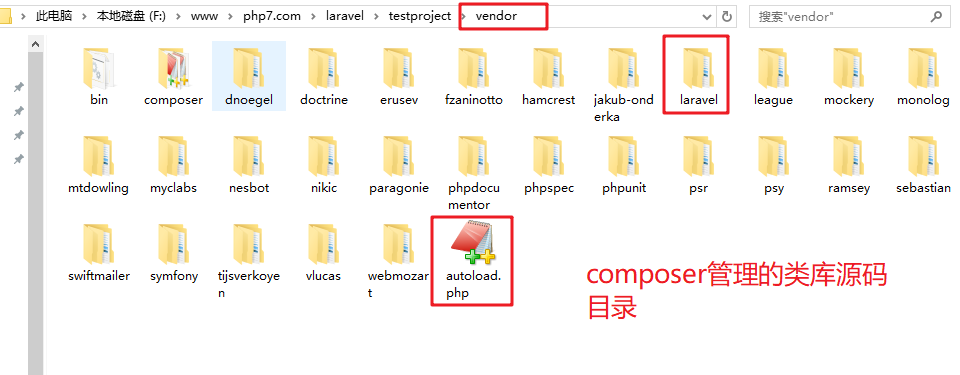
## 3.8、storage目录

主要是存放缓存文件和日志文件，注意，如果在【linux】环境下，该目录需要有【可写】权限。



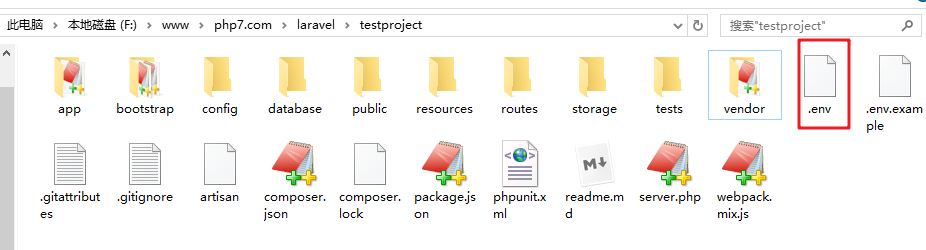
## 3.9、vendor目录

主要是存放第三方的类库文件和laravel框架的源码， composer下载的类库存放在该目录下面。



## 3.10、env文件【重要文件】

主要是设置一些系统相关的环境配置文件信息。config目录里面的文件配置内容一般都是读取该文件里面的配置信息（config里面的配置项的值基本都是来自.env文件）



## 3.11、artisan脚手架文件

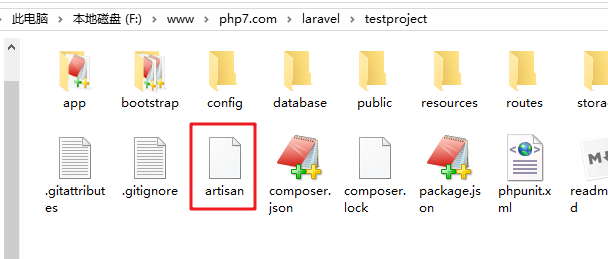
主要用于生成的代码的（自动生成），比如生成控制器，模型文件等。

执行命令：php artisan 需要执行的指令

执行此命令有2点要求

（1）、php必须添加环境变量，并且保证版本能是laravel所要求的版本。

（2）、执行命令时，artisan文件必须存在当前命令路径下。



# 四、运行laravel

## 4.1、方式一 命令【了解】

**php artisan serve**

**或指定端口来运行**

php artisan serve **--port**=8000

需要注意：一旦使用该方式则小黑窗（cmd）不能关闭的，一旦关闭则服务也跟随关闭，网站也就访问不了。修改了.env等一些配置文件，还需要ctrl+c结束进程，重启运行命令，不然修改不生效。

## 4.2、方式一 Apache【推荐】

# 五、CSRF攻击

## 5.1、CSRF是什么

CSRF（Cross-site request forgery），中文名称：跨站请求伪造。

跨站请求攻击，简单地说，是攻击者通过一些技术手段欺骗用户的浏览器去访问一个自己曾经认证过的网站并执行一些操作（如发邮件，发消息，甚至财产操作如转账和购买商品）。由于浏览器曾经认证过，所以被访问的网站会认为是真正的用户操作而去执行。

## 5.2、Laravel框架如何处理CSRF

原理图



## 5.3、取除指定路由中不进行csrf验证

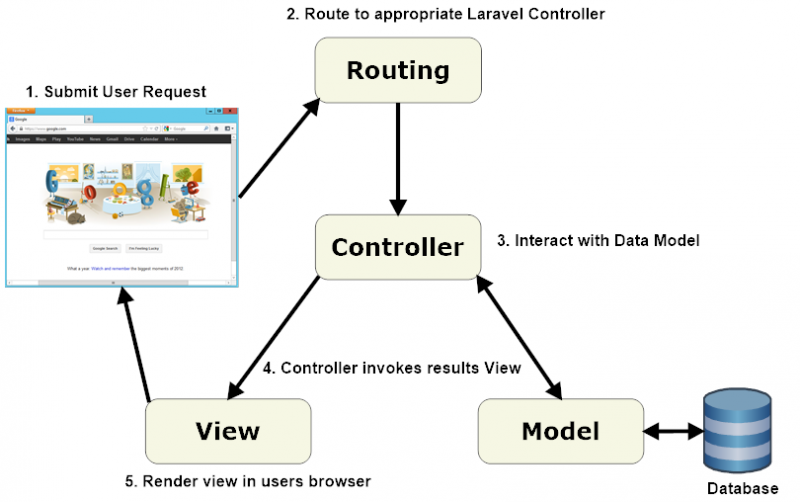
并不是所有请求都需要避免CSRF攻击，比如去第三方API获取数据的请求。

可以通过在VerifyCsrfToken（app/Http/Middleware/VerifyCsrfToken.php）中间件中将要排除的请求URL添加到$except属性数组中。

# 六、Laravel路由

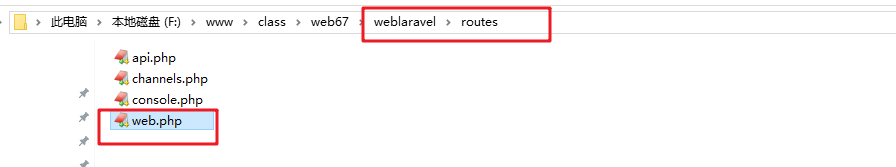
**什么是路由**

将用户的请求按照事先规划的方案提交给指定的控制器和方法来进行处理。



## 6.1、路由配置文件

路由文件的位置:项目目录/routes/web.php文件



## 6.2、4种基础路由的定义（重点）

Route::请求方式('url',匿名函数);

Route::请求方式('url','控制器名称@操作方法');

#从服务器取出资源（一项或多项） select

Route::**get**($uri, $callback);

#在服务器新建一个资源 insert

Route::**post**($uri, $callback);

#在服务器更新资源 update

Route::**put**/patch($uri, $callback);

#从服务器删除资源 delete

Route::**delete**($uri, $callback);

4种基础路由中，只有GET请求方式不进行CSRF验证，其它请求方式则需要CSRF验证

## 6.3、其它路由

Route::match([请求方式,多个以逗号隔开],$uri, $callback);

# 匹配所有请求 工作不用

Route::any($uri, $callback);

## 6.4、路由参数

# 必填

Route::get('URL/**{参数名称}**','闭包函数或控制器响应方法标识');

# 可选

Route::get('URL/**{参数名称?}**','闭包函数或控制器响应方法标识');

# 参数限制

Route::get('URL/**{参数名称}**','闭包函数或控制器响应方法标识')->where(['参数名称'=>'正则']);

## 6.5、路由别名【掌握】

作用：

* 方便管理生成的URL地址
* 权限控制

Route::get('/user/{id}',function($id){

return "当前用户id是：" . $id;

}) **-> name('名字');**

# 生成了对应的URL地址

$url = route('路由名称');

## 6.6、路由组之路由前缀

比如后台有如下路由

/admin/login

/admin/logout

/admin/index

/admin/user/add

/admin/user/del

# 上在的路由地址中admin为相同的路由前缀

Route::group(['**prefix**' => 'admin'], function () {

Route::get('users', function () {

// 匹配 "/admin/users" URL

});

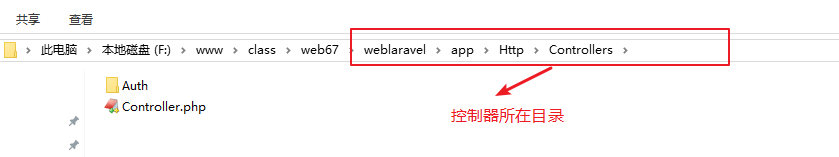
});

## 6.7、查看定义好的路由

php artisan route:list

# 七、控制器

## 7.1、控制器文件写在哪里？



## 7.2、控制器文件如何命名和创建

注意：**使用大驼峰命名，同时注意命名空间和基类控制器的引入**

php artisan make:controller 控制器名Controller

## 7.3、定义路由访问控制器中的方法

Route::请求方法(URL,'控制器类名@方法名')

# 八、请求

laravel框架为我们提供了多种获取数据的形式，我们常用的形式如下两种

## 8.1、Input类来获取数据

通过接收用户输入的类：Illuminate\Support\Facades\Input来进行获取数据。

Input::get(‘参数的名字’, ‘如果参数没有被传递使用该默认值’)

Input::all(): 获取所有的用户的输入

Input::only([]): 获取指定几个用户的输入

Input::except([]): 获取指定几个用户的输入以外的所有的参数

Input::has('名称')：判断某个输入的参数是否存在

上述方法即既可以获取get中的信息，也可以获取post中信息。

## 8.2、Request 获取数据【掌握】

获取数据功能太过于常用，所以laravel又以一个依赖注入的方式，帮我们实例获取对象。

# 九、响应

当服务器收到浏览器的请求后，会发送响应消息给浏览器。

## 9.1、返回字符串

在控制器或路由中 echo 或 return 一个字符串就可以了

return “PHP是世界上最好的语言”;

## 9.2、设置cookie

laravel框架为了安全，它的cookie是加密的

# 设置cookie

return response('')->cookie('id',111,10,'/');

##cookie($name, $value, $minutes, $path, $domain, $secure, $httpOnly)

# 读取cookie

echo $request->cookie('id');

## 9.3、重定向

# 路由别名

return redirect()->route('路由别名');

## 9.4、json数据返回

return response()->json([

'name' => 'zhangsan',

'age' => 22

]);

# 十、视图

视图的用途是用来存放应用程序中 HTML 内容，并且能够将你的控制器层与展示层分开。

我们在控制器中使用助手函数【**view()**】来加载视图模板。

## 10.1、视图目录位置与命名

## 10.2、分配数据到模板

语法：

view(模板文件名称,关联数组)

view(模板文件名称,compact('变量名1','变量名2'))

view(模板文件名称)->with(关联数组)

## 10.3、模板中输出变量

### 10.3.1、变量

{{$变量名}}

例

{{$name}}

### 10.3.2、三元运算

{{ $name or 'Default' }} 等价于<?php echo isset( $name)?$name:'default'?>

### 10.3.3、未转义输出

如果变量信息里边如果有html标记信息，在输出的时候html标记被转化为符号实体了，而没有被浏览器解析掉，如果希望看到被浏览器解析后的内容，就需要设置两个感叹号。

例如：

$title = "<a href='http://www.baidu.com'>百度</a>";

{!!$变量!!}

### 10.3.4、原始形态输出

由于很多 JavaScript 框架都使用花括号来表明所提供的表达式，所以你可以使用 @ 符号来告知 Blade 渲染引擎你需要保留这个表达式原始形态。

<h1>Laravel</h1>

Hello, **@**{{ $变量名 }}

## 10.4、使用函数

可以在blade模板中直接使用php函数

{{php函数名()}}

## 10.5、if语句

@if (count($records) === 1)

我有一条记录！

@elseif (count($records) > 1)

我有多条记录！

@else

我没有任何记录！

@endif

## 10.6、循环

@foreach ($users as $user)

<p>此用户为 {{ $user->id }}</p>

# 如果传过来的就是数组那么我们就用

<p>{{ $user['id'] }}</p>

@endforeach

---------------------------------------

@forelse ($users as $user)

<li>{{ $user->name }}</li>

# 如果传过来的就是数组那么我们就用

<p>{{ $user['id'] }}</p>

@empty

<p>没有用户</p>

@endforelse

注：foreach有循环变量，$loop，$loop->index 当前循环所迭代的索引，起始为 0。

## 10.7、模板包含

@include('public.header')

# public 表示 views下面的public目录

# header 表示在views/public/header.blade.php文件

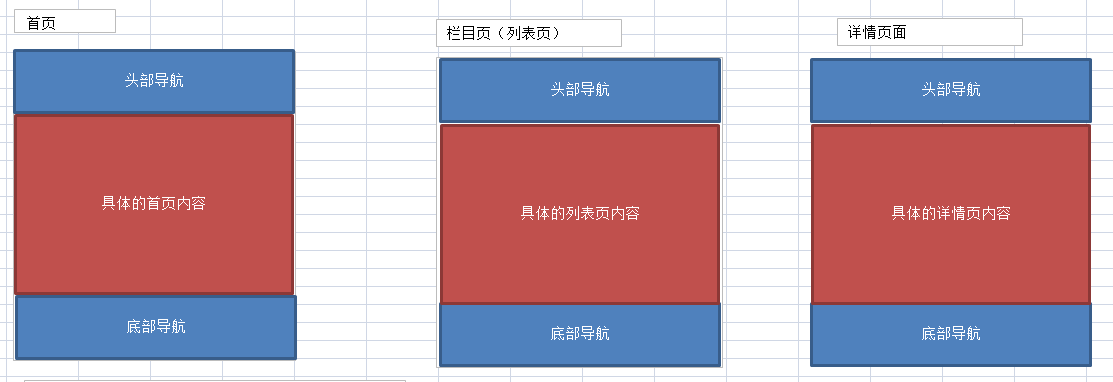
## 10.8、模版继承

在一个项目中有许多模板文件，它们有一个特点：拥有共同的头部 和 脚部内容

为了避免相同代码重复开发、维护 造成工作效率低下

可以把共同的头部和脚部内容集中到一个布局文件中，之后各个具体模板文件去继承该布局文件而使用头部和脚步内容。这个过程成为模板继承。

布局文件中相同的代码只维护一份，会大大提升项目开发效率



@extends('laoyouts.home') #表示继承resource/views/layouts/home.blade.php布局文件

使用section标签替换布局模板中可变区域；

@section('content')

给布局文件yield('content ')的区域进行填充的内容

@endsection