**php**

1. php：超文本预处理器

运行在服务器端，内嵌在html中的脚本语言。

集成开发环境：wampserver、xmapp、phpstudy、phpserver

架构：lamp、lnmp

1. 显示所有php信息：phpinfo();

php是模块化的，可以在php.ini开启需要模块。

### 一、基础

1. 标签

|  |
| --- |
| **<?php** *phpinfo*(); |
| **<?php** *phpinfo*(); **?>** |
| // 需要在php.ini文件中开启短标签风格：short\_open\_tag = On,然后重启所有服务。  **<?** *phpinfo*(); **?>** |

1. 常用函数
2. isset()：判断一个变量是否存在
3. unset()：销毁变量
4. 单引号不解释变量，双引号解释变量。尽量使用单引号

|  |
| --- |
| **echo '\\'**; **echo "\n"**; **echo '\''**;  **echo "**{$x} **+** {$y} **=** {$z}**"**; **echo "**$x **+** $y **=** $z**"**; |

1. 标量：整型、浮点型、布尔类型、字符串

混合类型：数组、对象

特殊类型：空、资源

1. 定义常量：define(‘ABC’, ‘abc’)，常量名ABC前不需要加$,’abc’为常量值。常量值只能是标量，常量的命名都大写。

defined(ABC)：判断常量是否被定义。

系统常量：\_\_FILE\_\_、\_\_LINE\_\_、PHP\_VERSION、\_\_DIR\_\_、\_\_FUNCTION\_\_、

PHP\_OS（获取系统信息）、M\_PI（圆周率）、\_\_METHOD\_\_（获取当前成员方法名）、\_NAMESPACE\_\_（获取当前命名空间名字）、\_\_TRAIT\_\_（获取当前TRAIT名字、多继承）、\_\_CLASS\_\_（获取当前类名）

1. gettype($name)：获取数据类型
2. intval($name)：转换为int

floatval($name)、strval($name)、boolval($name)

1. var\_dump($name)：获取类型和值。
2. $num = ‘0’ // false

$num = ‘’ // false

$num = ‘ ’ // true

$num = ‘0.0’ // true

1. mt\_rand(1, 6) // 获取1~6的随机整数
2. 函数名不区分大小写
3. 函数可以有默认参数 function test($name = ‘张三’){}

## 二、Laravel

Lavael是一套简洁、优雅的PHPWeb开发框架。LTS(Long Time Support)

php框架：ThinkPHP、CI、Yll等。

特点：单入口，所有的请求必须从单入口开始，主要是便于管理（统一的参数过滤）

Laravel框架中，所有的URL访问都必须事先定义好路由规则。

1. 配置
2. php.ini配置文件需要开启的扩展：

extension=php\_openssl.dll

extension=php\_pdo\_mysql.dll

extension=php\_mbstring.dll

extension=php\_fileinfo.dll（验证码代码依赖需要该扩展）

extension=php\_curl.dll（主要用于请求的发送）

1. httpd.conf配置文件需要开启的模块

LoadMoudle deflate\_module module/mod\_deflate.so

LoadMoudle.rewrite\_module module/mod\_rewrite.so

1. 添加php环境变量
2. composer：php中用来管理依赖关系的工具。

官网下载安装：<https://getcomposer.org/download/> 注意版本，不填写代理。

命令行打入composer测试是否成功。

若不成功可自行添加环境变量：C:\ProgramData\ComposerSetup\bin

1. 使用composer方式部署laravel项目：
2. 配置国内镜像：

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

1. 创建一个名为laravel的项目：

切换到创建目录下，执行：composer create-project laravel/laravel –prefer-dist ./test

--prefer-dist：表示优先下载压缩包，而不是直接从github上下载源码

test：为目录名。

指定版本号下载：**composer create-project laravel/laravel=5.4.\* –prefer-dist ./demo01** 【下载时，如果不指定版本号，默认下载最新版本。】

composer更新命令：composer self-update

1. 目录结构：

app：项目的核心目录，主要存放核心代码、控制器、模型。

User.php：默认的模型文件。

bootstrap目录：lavael启动目录，存放框启动需要的目录

config目录：项目的配置文件，主要存放配置文件。

app.php：项目主要配置文件。

auth.php：用于定义用户认证（登录）的配置文件。

database.php：针对数据库的配置。

filesystems.php：上传文件、文件储存需要使用到的配置文件。

database目录：数据库迁移文件。

migrations：建表操作。

seeds：种子文件，存放数据表的数据填充文件。

public目录：项目的入口文件和系统的静态资源目录（后期使用的外部静态文件js/css/图片等都放到该目录下）

index.php：项目单一入口文件

resources目录

lang：语言目录（如果项目需要本地化则需要配置语言包）

views：视图文件存储目录

routes目录：定义路由的目录，

web.php：定义路由的文件

storage目录：主要用于存放缓存文件和日志文件。

app：存放用户上传的文件

framework：框架运行时的缓存文件

logs：日志文件

vendor：第三方的代码依赖

.env：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）。

artisar：脚手架文件，主要用于代码的自动生成。（命令：php artisan 需要执行的命令）【注：php必须要添加环境变量，artisan必须存在命令行当前的工作路径下】

composer.json：依赖包的配置文件

1. 启动方式：

方式一：php artisan serve 该启动方式与apache无关。（不推荐）

方式二：使用wamp或lamp。

1. 配置虚拟主机

修改apache的虚拟主机vhosts.conf

|  |
| --- |
| <VirtualHost \*:80>  #配置站点管路员的邮箱，当站点产生500错误时，会显示在页面上  ServerAdmin webmaster@yourdomain.com  #站点根目录  DocumentRoot "E:\phpStromworkspace\demo01\public"  #站点需要绑定的域名  ServerName www.hdljd.com  #服务器的别名  ServerAlias hdljd.com  #针对站点的详细配置  <Directory "E:\phpStromworkspace\demo01\public">  allow from all  AllowOverride All  Options +Indexes +Includes +FollowSymLinks +MultiViews  </Directory>  </VirtualHost> |

1. 修改hosts文件（线上叫DNS域名解析）

win+R：drivers 🡪 etc 重启apache

|  |
| --- |
| 127.0.0.1 www.hdljd.com hdljd.com |

1. 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或者功能函数来进行处理。【通俗的讲，路由就是访问地址形式】

注册路由相应多个http请求：

|  |
| --- |
| Route::match([‘get’, ‘post’], ‘/’, function() {});  Route::any(‘/foo’, function() {}) |

1. 路由（写在routes/web.app）

* 传参：

必选参数：Route::get(‘/foo/{id}’, function($id) {})

可选参数：Route::get(‘/foo/{id?}’, function($id = ‘’) {})

通过拼接？传参更佳：Route::get('/test1', function () {})

* 路由别名：

Route::any(‘/foo’, function() {}) ->name(‘名字’)

调用该路由：route(“名字”)

查看系统已经有的路由命令：php artisan route:list

* 路由群组

Route::group([‘prefix’ => ‘admin’], function() {

Route::get(‘users’, function() {})

})

1. 控制器（写在app/Http/Controllers）
2. 命名：大驼峰的控制器命名+Controller.php
3. artisan创建控制器：**php artisan make:controller [Adimin/]TestController**
4. 控制器路由：使用路由规则调用控制器下的方法，而不再走回调函数。

**Route::get('/home/test/test1', 'TestController@test1');**

1. 分目录管理

自己创建目录，执行创建控制器命令。

1. 接收用户输入

接收用户输入的类：**Illuminate\Support\Facades\Input**

Facades：门面是介于一个类的实例化与没有实例化中间的一个状态。其实是类的一个接口实现。在这个状态下可以不实例化类，但是可调用类中的方法。

|  |
| --- |
| Input::get(‘parame’, ‘defaultVal’)  Input::all() // 获取所有的用户输入  Input::get(‘’) // 获取单个用户的输入  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参数  Input::has(‘name’) // 判断某个输入的参数是否存在  上述方法既可获取get中的信息，也可获取post中的信息 |

【补充：在laravel中如果需要使用facades的话，可在config/app.php中aliases设置别名】

1. dd() 框架内置函数dump+die，后续代码将不再执行。
2. DB类操作数据库

**数据库在.env文件中配置**。

【另：config下database.php：env()函数表示先从env文件中获取，如果获取成功则使用；如果获取失败，则使用env函数的第二个参数】

* 插入：

DB::table(‘表名’) -> insert() insert()参数既可为一维数组也可为二维数组，返回true或者false。

DB::table(‘表名’) -> insertGetId() 只能插入一条数据，必须为一位数组，返回id

* 修改

update()：返回收到影响的行数。

increment()：只能修改数字字段的数值，递增

decrement()：只能修改数字字段的数值，递减

where(字段，运算符，值)：如果第二个参数为“=”，则可以省略不写，先指定条件后update。

* 查询

get()：返回一个集合对象，get查询结果的每条记录是一个对象，不是一个数组。

where() -> where() ->get() 相当于and

where() -> orWhere() ->get() 相当于or

where() -> first() 返回第一行数据，等价于limit 1。

where() -> value( ‘name’ ) 返回第一条数据某个字段的值。

where() -> select( ‘name as user\_name’ ,’email’) ->get() 获取某些字段数据

orderBy(‘age’, ‘desc’) ->get() 排序

limit(3) -> offset(2) -> get() limit表示限制输出的条数，offset表示从哪里开始，相当于limit 3,2

* 删除

delete() 删除记录，返回受到影响的行数

truncate() 删除整个数据表

* 执行任意的select语句

DB::statement(“insert into…”) 增删改，返回影响记录的条数

DB::select(“select \* from…”) 查询语句

1. 视图操作