开启aphca，mysel 浏览器进入 localhost:端口号 有内容则启动成功

php文件位置：

xmapp下的htdocs下建立项目文件夹001，001文件夹建立001.php

运行：localhost:端口号/001/001.php 代码修改后要刷新页面。

基本语法：

<?php

//php代码，以分号结尾（不分号结尾只有小部分情况不报错，大部分是报错的）

?>

<?php?>可以单独写，也可以写在html中，此时?>不可省

若php文件只有php代码，?>可省

（1）注释，输出

\*注释

# 单行 /\*多汗\*/

\*输出语句：

echo xxx; echo xxx,xxx; #输出语句

var\_dump(xxx) #输出数据类型和值

print\_r #输出复合数据类型

dd() #键值对形式输出，常用来输出对象，但是会覆盖当前所有的内容包括模板，若有多#条dd语句，则只有第一条生效。

（2）标识符，变量，常量，数据类型，运算符

\*标识符：与c一样

\*变量

$name 定义/使用

$name=5 $name=$b

$name=&$a 地址赋值

unset($a) 销毁变量

\*常量：不用$

无法销毁，初始化后不能赋值/修改

define(‘name’,value); 可以加第三个参数，不加为false，true/false区分/不区分大小写

const a=3; const定义常量

\*数据类型：

gettype($a) #返回数据类型

标量数据类型：int，double，string，bool

复合数据类型：array，object

特殊数据类型：resource，null

数组：

下标数组：

声明：$a=array(1,’c’,3); 或 $a=[‘a’,2,3];

长度：count($a);

*添加元素：$a[]=5; $a[count($a)]=6 尾部添加*

关联数组：

$a=[“a”=>5,”b”=>a] 用 $a[“b”]访问 相当于键值对，key可以是数字/字符串

添加值： $a[value]=key; 尾部添加

下标数组和关联数组可以是一个数组，各用下标/key访问

多维数组：

$a=array(array(),array())定义，$a[3][4]访问

数组的独特遍历方法：

foreach($arr as $key){} $key是值

foreach($arr as $key=>$value){} $key是下标，$value是值

数组赋值给数组，是深拷贝，而数组=&数组是浅拷贝

判断两个数组是否相等，是判断值

string：

“”内可以用{$a}嵌入变量，可使用转义字符，而’’不行

“”中 $a 也可以嵌入，但最好用{$a}

<br> 换行 ‘’ “”都可使用 \n 空格

\*数据类型转换：

number.format($a,2) 保留两位小数

number.format($a) 数字转成整型，并用千分号分隔（每三位逗号隔开）

number.format($a) 纯数字字符串（字符串不是纯数字不行）转成整型，并用千分号分隔（每三位逗号隔开）

$str+0 字符层转整型

(int)$a intval($a) (float)$a (double)$a

\*非数字开头字符串转数字为0，数字开头为数字

\*空数组/空字符串为0，非空数组为1

(bool)$a (boolean)$a 非空非0位true，空0 null为false

strval($a) 数值转字符串原样，布尔转字符串true/false长度1/空串，数组转字符串为array

null转字符串为空串。

implode(‘,’,$a) 逗号分隔数组转字符层

(array)$a 数值/字符串转为仅有该元素的数组，true长度1数组，false/null空数组。

\*运算符

. .= 连接数值/字符层

=== 与js一样

不像js有短路

$a<=>$b $a大于/等于/小于$b，为1/0/-1

xor 异或

其他与c一样

（3）流程控制

elseif else If 中间的空格可以没有，其他与c一样

（4）函数

function funcname($a){}

funcname(5);

形参可以用$$a=3设置默认值

global

函数外部定义的变量都是局部变量，函数本身是无法调用的，需要在函数内global $a; 将外部的局部变量变为全局变量，函数才能使用（流程控制中可以使用外部定义的局部变量）

$GLOBALS是自带的保存所有变量的数组。

static

函数内 static $a=1; 定义静态变量，每次调用导致此静态变量的修改会保留到以后的调用（循环也可用）。

传值传引用与c一样，不同的是 形参用 &$a 实参直接用用 $a

内置函数：

\*数组

print\_array\_values($a) 遍历关联数组所有值

print\_array\_keys($a) 遍历关联数组所有键

array\_push($a,”aaa”) push并返回push后的长度 array\_unshirt() 最前面添加

array\_pop($a) pop并返回删除的值 array\_shirt() 最前面删除并返回长度

array\_implode(“ ”, $a) 空格隔开转字符串

sort($a) 升序排序

fmod()取浮点数余数 %只会取整数余数

\*字符串

str\_word\_count($s) 返回字符串的单词数

strrev($s) 翻转字符串

strstr($s,’@’) 截取字符串中从@开始到最后的内容

stripos($s,’a’) 区分大小写，查找a第一次出现的下标

substr($s,2,3) 从下标2开始，截取3个字符

str\_replace(“123”,”abc”, $s) 将123替换成abc 区分大小写 srr\_ireplace()不区分

strtoupper($s) 去不转大写 ucfirst($s) 首字母大写

mb\_strlen($s) 字符个数 strlen($s) 字节个数

explode(‘” ”,$s) 将字符串按空格分隔，后转数组

str\_split($s,3) 每3个字符分隔，转数组

\*数学

abs(-2) max(4,5,6) min(1,2,3) floor(3.4)向下取整 cell(3.4)向上取整 round(3.6)四舍五入

pow(3,2)3的平方 sqrt(4)根号4 rend(1,10)1到10左闭右闭的随机数 mt\_rend(1,10)速度更快

\*日期

time()秒时间戳 microtime(true)毫秒时间戳 microtime()秒和毫秒的时间戳

date\_default\_timezone\_get() 获取当前时区，默认为UTC

date\_default\_timezone\_set(‘Asiz/Shanghai’) 设置时区为上海

date(‘’,时间戳)格式化时间戳，字符串内自定义格式 Y m d n H I s a为上下午 L1是0否闰年

mktime(时,分,秒,月,日,年) 转成时间戳 strtotime(“年-月-日 时:分:秒”) strtotime(“now”)当前

checkdate(月,日,年) 检查日期是否存在

(时间戳-时间戳)/(24\*60\*60) 日期差

（5）面向对象：

与java差不多

\*访问修饰符没有缺省，只有public，private，protected，属性必加，方法可不加默认public

\*实例化：$a=new aaa()

\*调用属性方法：用->调用

\*构造器： function \_\_construct(){}

\*析构器： function \_\_destruct(){} 自动调用且无法自己调用

\*类内的方法外常量只能用consr定义，类内用self::常量名访问，类外用对象名::常量名访问，类名::常量名类内类外都行

\*类内的方法内可以用define()定义常量，方法内不用::访问。当然方法外无法调。

\*方法内用$this->xxx 调用类内属性

\*static调用与常量一样，static属性不需要实例化就可调用

\*静态方法内用selt::xxx调用，静态/非静态方法只能调静态/都可以属性

\*继承extends

2、高级

（1）文件操作：

3、laravel

（1）安装

\*php包管理工具：composer

注意安装时要制定php.exe

Composer –v 检查是否安装成功

\*laravel安装

composer global require "laravel/installer"

两种创建项目方式：

laravel new xxx

composer create-project --prefer-dist laravel/laravel xxx "5.6.\*"

运行,浏览器：

localhost:端口号/项目目录/public/url 数据库也同时开启

或cmd进入项目目录：

php artisan serve 但是数据库需要另外开启

浏览器输入 localhost:8000/url

入口文件：public/index.php

目录结构：

public 入口，资源

（2）语法

①路由，在routes/web.php中设置

php artisan route:list #展示所有路由的列表

基本路由：

Route::get(‘url’,function(){}); ‘url’可不加/

就可以通过 localhost:端口号/项目目录/public/url 访问

回调函数中跳转方式：

\*直接写代码，如echo

\*return 123 会把123显示在页面上

\*return view(‘xxx’); 会展示 resources/views/xxx.blade.php

\*return redirect(‘url’); 跳转到url，需要先配置该目的url的路由

\*return back(); 返回

也可以通过控制器设置路由：

在cmd’中项目目录下，命令行创建文件：

php artisan make:controller xxx

创建在app/Http/controllers/aaa.php，也可以直接创建该文件

在aaa.php里面的类中，增加：

public function xxx(){}

web.php中：

Route::get(‘url’,’aaa@xxx’);

控制器注意：

\*每个控制器/中间件都需要命名空间namespace 指向当前所在的路径

\*若出现类名不存在，则在app/provides/RouteServiceProvider中， protected $namespace = 'App\\Http\\Controllers'; 取消注释，没有就加上这行。

\*namespace后面跟该控制器的所在路径

\*控制器文件名与里面的类名要一样。

\*导入的类名不要与控制器的类名冲突。

多请求路由：

Route::match([‘get’,’post’],’url’,function(){}); #数组包含请求类型

Route::any(‘url’,function(){}); #所有类型的请求的适用

②路由参数：

\*设置必选参数

Route::get(‘url/{id}’,function($id){});

\*设置可选参数，此时函数参数必须要有默认值

Route::get(‘url/{id?}’,function($id=123){});

\*用正则表达式限定参数规则

Route::get(‘url/{id}’,function($id){})->where(‘id’,’[0-9]+’); #规定id必须有一个数字

\*传递多个参数

Route::get(‘url/{id}/{name}’,function($id,$name){})

\*多个参数的规则限定

->where([‘id’=>’…’,’name’=>’…’])

③命名路由

Route::get(‘url’,[‘as’ => ‘rName’,function(){} ]);

rName不能再url上使用，一般用在a标签的跳转：

\*<a href=”{{ url(‘url’) }}”></a> 使用url跳转

\*<a href=”{{ route(‘rName’) }}”></a> 使用路由名跳转

④资源路由：

Route::resource(‘url’,’xxx’);

\*xxx后不跟@方法名

\*控制器内的类内入口方法名必须是index

\*控制器内的类的其他方法也不能自定义，必须是create，update，destory，show，store

⑤路由群组

当路由指向的控制器有多个层级：

Route::get(‘a/aa’,’a\aa@fff’);

控制器aa的命名空间namespace要指向当前路径：

namespace App\Http\Controllers\a

并加上一行代码：

use App\Http\Controllers\Controller

\*未使用路由群组：

Route::get(‘a/aa’,’a\aa@fff’); 当有多个这样的路由时，代码就有点冗余。

\*使用后：

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

Route::get(‘aa’,’aa@fff’);

});

（2）模板，在resources/views/ 下，xxx.blade.php

引入自己的css和js，从public目录开始找：

<link rel="stylesheet" href=" {{ URL::asset('css/1.css') }} ">

<script type="text/javascript" src="{{ URL::asset('js/1.js') }} "></script>

引入CDN的css和js，去掉http:即可

<link rel="stylesheet" href=" {{ URL::asset('//域名/css/css.css') }} ">

<script type="text/javascript" src=" {{ URL::asset('//域名/js/js.js')}}"></script>

①基本使用：

模板可以写html，并用 @xxx 控制内容，@xxx()里的参数可以变量也可以是值

@yield(‘yieldName’,默认值) 在不指定内容时，使用默认值

@section(‘secName’)

….html….

用来嵌入内容，secName为该section的名字，@show为该section结束的标志

@show

②继承，替换/扩展内容：

子模板中：

@extends(‘路径/父模板名’) 父模板名不加后缀.blade.php 若父模板在当前目录则直接写父模板名。

会继承父模板的所有内容。

在父模板基础上给子模板替换/扩展内容：

父模板中，@yield只可替换不可扩展，@section可替换可扩展

\*@section和@yield的替换格式：

@section(‘secName或yieldName’)

…替换的html..

@stop 或 @endsection

当替换的内容只有文本时，可以写成 @section(‘secName或yieldName’,’文本’)，不用加@stop

\*@section扩展内容的格式：

@section(‘secName’)

@parent

…新增的html…

@stop 或 @endsection

\*@show和@stop区别：

@shiow解析并显示该@section内容，只在首次定义@section时使用

@stop只解析内容，在替换/扩展内容时使用

③return view()传递参数，view接收

在Route::get()里的function 或 控制器中的index() 中：

$xxx=123;

$aaa=[5,6,7];

return view(‘viewName’,[

‘x’=>$xxx,

‘a’=>$aaa

]);

view中：

{{ $x }}

{{ $a[0] }}

{{ var-dump($a) }}

<ul>

@for($i;$i<count($a);$i++)

<li>{{ $a[$i] }}</li>

@endfor

</ul>

④流程控制与其他：

[除了@for...@endfor](mailto:除了@for...@endfor) [还有@foreach...@endforeach](mailto:还有@foreach...@endforeach)

@foreach($arr as $i) $i为元素

@foreach($arr as $k => $i) $k为索引，$i为元素

循环体内有内置的$loop

$loop->index 索引

$loop->parent 父循环的$loop $loop->depth 当前循环嵌套的深度(1开始)

$lppp->iteration $loop->first $loop->last 当前循环次数(1开始)，第一次循环1，最后一次循环

$loop->count $loop->remaining 循环总数，剩余循环数

@if…@elseif…@else…@endif

@include(‘视图’)，可以有第二个参数，为数组，传参给视图

注意不要进入了死循环

（3）表单：

<form action=”/url” method=”get”>

{{ csrf\_field() }} #post专用且必用

….

<input type=”submit”>

</form>

点击submit后，会replace跳转到对应的url，请求类型为get（不写method默认为get）

注意：必须具有name属性的表单，才会将表单的value发送给后端

POST请求必用 {{ csrf\_field() }} ,否则会出现419的错误。

作用是生成一个 <input type=”hidden” name=”Token” value=”随机序列”>，形成跨站请求伪造，防止网站被恶意攻击。

还需要配置相应的路由和控制器

路由：Route::get(‘/url’,’sss@index’);

控制器：

use Illuminate/Http/Request #引入request类

index()方法：public function index(Request $request){};

$request->fullUrl() #当前完整url

$request->url() #不带query参数的url

#request->path() #路由的url

$request->method() #请求类型

$request->isMethod(‘xxx’) #请求类型是/不是xxx，true/false，xxx不区分大小写

$request->all() #获得所有有name属性的表单发送的value，是一个数组

$request->input(‘inputName’) #获取对应name的inout发送的value，只能获取单个input

$request->only(‘inputName1’,’2’…) #获取对应name的inout发送的value，参数可以是多个，也可以是一个数组，返回的是一个数组。

$request->except(‘inputName1’,’2’…) #获取除了对应name，其他所有的inout发送的value，参数可以是多个，也可以是一个数组，返回的是一个数组。

$request->has(‘inputName’) #该input存在/不存在，true/false，区分大小写

（4）验证码/登录：

①验证码：

1、自己定义

比较繁琐

\*App\下建立 Tool\ 下建立 ValidateCode 下建立 ValidateCode.php ，内容：

use App\Tool\Validate

class ValidateCode {….}; 内容是创建出验证码图片，并给一个get方法获取验证码图片。

\*创建一个控制器

use App\Tool\Validate\ValidateCode #导入

inxex中new该类，return get方法

\*配置路由

\*img标签使用：src=”url(‘/url’)”

2、安装验证码的库直接使用

②登录：

1、通过路由中间件限制非登录状态的访问（判断session是否存在）

路由中间件：对进入路由前的一些限定访问操作做封装

存放在App\Http\Middelware中 php artisan make:middleware xxx 创建

创建后，中间件中有一个xxx类，xxx类中有一个handle方法：

public function handle(Request $request, Closure $next)

{

if($request->session()->has('userName'))

return $next($request); #若sesson中有用户名，则允许访问

else return redirect('/fa'); #否则不允许访问，回到登录界面

}

然后到 App\Http\Kernel.php 中对该中间件进行注册：

找到 protected $routeMiddleware = [] 这个数组，增加一个：

'lo' => \App\Http\Middleware\lo::class,

然后到web.php中：

通过路由群组链接中间件：

Route::group(['middleware'=>'lo'],function(){

//注意该路由群组里不用包含登录界面路由，而应该放的是被限制访问的路由

})

2、进行登录验证，验证若成功，将用户名保存到session，以便中间件判断

在登录跳转的控制器中：

use DB;

index()中：

$username=$request.input(‘username’);

$password=…

$验证码=…

If(//验证码判断是否正确){

#需要提前建立好数据库，根据表单提交的数据在数据库中查询是否存在，查到$res为数

#组，查不到则是空数组

$res=DB::select(‘select \* from 表 where 账户字段=? And 密码字段=? ,[$username,$password]’);

If($res){

session([‘key’ => $username]); #将用户名保存到session

return 登录成功并跳转; #若$res非空，则验证成功。

}

else return back();

}

else echo ‘验证码错误’;

3、退出登录，清除session即可

在第二步相同的控制器的相同类中，增加方法：

public function xxx(Request $request){

$request->session()->flash(); #清除session所有内容

return 跳转回登录界面;

# $request->session()->forget(‘key’) 清除session部分内容

}

通过a标签的href=”{{ url(‘clearurl’) }}”

Route::get(‘/clearurl’,控制器@xxx);

③增删改查内容，适合用资源路由