1. routes/web.php中Route::get(‘/{id}’,function(){})->name(‘login’)->where([‘id’=>’[0-9]+’])，可以进行路由的参数限制，可以多个参数限制；
2. middleware中间件的作用是在作用在路由的应用层，也就是说用户进行http对其进行请求时，middleware可以进行做出筛选或者限制。用 php artisan make:middleware UserMiddleware,中间件都放在app/http/middle下，然后在对应生产的UserMiddleware文件里的handle()中进行条件设置，比如

if($request->id <= 200){

return redirect(‘home’);

}

return $next($request);

然后应用在路由：

Route::middleware(‘User’)->group(function(){

Route::get(‘/{id}’,控制器@action);

Route::get(‘/users/{id}’,XXController@action);

})

这里也是进行了中间件的属性共用，也能进行条件筛选；

中间件也有一个前缀后缀，也就是在$next($request)的前面或者后面去运行。

中间件有一个terminate方法，也就是服务器反应到浏览器后所执行的动作:

Public function terminate($request,$response){ }

1. route可以进行路由的隐形查询，指的的是for instance::参数Route::get(‘/{id}’,function(User $user){ return view(URL,compact(‘user’))}因为后面的function有User模型的实例，然后Laravel的隐性route就会对应的去表查找id这个模型实例返回。
2. CSRF令牌验证：
3. 有post,put,delete,patch表单提交的时候需要进行验证，那我们就要在表单那里添加{{ csrf\_field() }}方法即可，不支持delete,put,patch，那我们就用{{ method\_field() }}来进行隐藏提交。
4. 还有对请求头的x-csrf-token的验证，也就是要在header那里有 <meta name=’csrf-token’ content={{ csrf\_token() }} ,for instance::

$.ajaxSetup({

headers:{

‘X-CSRF-TOKEN’:$(‘meta[name=”csrf-token”]’).attr(‘content’)

});

不过默认的bootstrap.js默认已经是带有这些的，除非你不用这里的jq，就得手动去配置了。

1. 如果要进行CSRF令牌的白名单设置的话，可以把路由放到routes/web.php文件下或者在VerifycsrfToken中间件的$except属性来排除对这类路由的CSRF保护。
2. 可以在控制器那里设置一个public function \_\_invoke($id)来单个控制器操作方法，路由则写成route(‘/{id}’,’XXController’)，不用写方法即可。
3. middleware在中间件在route和控制器都是可以使用，但一般建议在控制器那里使用$this->middleware(‘auth’,[‘except’=>[‘create’,’show’]]),,也可以在路由那里直接声明Route::resource(‘photos’,’PhotoController’)->middleware([ ‘only’=>[‘show’,’create’] ])这里进行限制哪些路由可以访问。

7.本地化资源URI：

AppServiceProvider的boot中使用Route::resourceVerbs([ ‘create’=>’crear’, ‘edit’=>’editer’ ])方法，然后Route::resource(‘users’,’UserController’)中的/users/create, /users/edit对应的路由会变成/users/crear ,/users/editer.

8.route缓存：路由必包是没有缓存的，控制器是有的，而且有缓存的时候，反应速度会提高一百倍；；程序开发有三个步骤：开发环境->测试环境->生产环境，，使用php artisan route:cache来生产缓存，但只在生产环境开启(因为你每次修改路由都得重新生成下缓存才行)，使用php artisan route:clear 来清除缓存。

9.HTTP请求：Request $request ,:

$request->path()返回路径，除了域名外后的路径。

$request->is()是判断请求的http和规则是否一致，比如$request->is(‘admin/\*’)，\*号代表通配符。

$request->url() 或 $request->fullUrl()方法是获取完整的URL，只是fullUrl()连带获取参数。

$request->method() http的请求方式。

$request->isMethod(‘post’) 判断请求方式跟指定的规则一样。

$request->all()获取所有输入的数据，以array的形式返回。

$request->input(‘name’,’Jackey’);可以使用这样来获取用户输入的单个值，如果没有则默认返回第二个参数(可选)，若提交的表单存在数组，则用$request->input(‘products.0.name’)。

$request->name 来动态获取值，这里的name 是表单中的name属性的名字。

如果传入的数据是JSON数据，若请求的Content-Type标头正确设置为application/json可以用$request->input(‘user.name’)有点来获取值

10.return response()->header(‘Content-type’,$type);; return reponse()->withHeaders([])等两个来方式来进行返回的头部信息设置；在返回头部信息的时候，直接用cookie($name,$value,$minutes)链接方式来进行cookie储存，cookie的值默认是会加密和签名的，若不要则到App\Http\Middleware\EncryptCookies这里的$except=[ ‘cookie\_name’]设置下。

11.redirect()： redirect()->back()是基于web中间件 和 所有session中间件的路由，因为back()是基于缓存的；；redirect()->action(‘Controller@action’,参数) 转到这去，并传递参数过去，肯定是数组的形式啦；；redirect()->with(‘status’, ‘profile update’)这是来进行session的闪存的。

12.如果你在控制器那里返回视图则直接view()全局辅助函数，若要返回标题和自定义状态码和返回的视图做为响应的内容，则使用response()->view(‘title’, $data, 200)->header(‘Content-type’, $type);

13.对哦，response返回的 ‘数组’ 或者 ‘对象集合’ 都会自动的转化为JSON格式, response()->json([ ‘name’=>’Jackey’, ‘age’=>’222’]);这里的json()方法会自动的为头部信息添加application/json 和 利用php的json\_encode()来进行转换

14.文件下载：response()->download($pathToFile, $name, $headers)这里的第二个参数是下载时显示的名字，第二个是响应的头部信息(数组)，文件名必须是ASCII值。

15.文件响应：response()->file($path,$header)第一个参数是显示文件名，第二个是显示头部信息，这里是在浏览器显示而已，并不下载。

16.生成基础URL:

echo url(‘post’); //生成<http://example.com/post> 这样或 <https://example.com/post> ，就是http:// 或 https:// + 主机名 和url() 方法里的参数。

url()->current();返回当前的url，不带查询字符的；

url()->full();返回当前的url,带查询字符；

url()->previous(); 返回上一个请求的完整url.

17.session:

$request->session()->get(‘key’, ‘default’)获取没有键的值

$request->session()-get(‘key’)获取单个键值

$request->session()->all()获取所有的值

$request->session()->push(‘user.name’, ‘Jackey’)往user.name数组存一个数据

$request->session()->flash(key,value)进行数据闪存(意思就是短期的数据储存)

$request->session()->keep([‘username’, ‘email’])保留特定的闪存值

session([‘key’=>’value])全局辅助函数储存

$request->session()->put(‘key’, ‘value’);

$request->session()->forget(‘key’)删除一条session数据

$request->session()->flush()清除所有session数据

$request->session()->regenerate();进行Session ID 重新生成

18.Eloquent :

1) 因为Eloquent会进行表字段的自动保护，如果要插入数据，需要在模型中添加 protected $fiillable 或者protected $guarded ：

$fillable=[] 在这里面的表示可以赋值；

$guarded=[] **除了**在这里面的字段，其他的都可以赋值；

2)使用游标卡尺可以很大程度上减少内存的开销：

Foreach(User::where(‘foo’, ‘bar’)->cursor() as $flight){ }

1. 要删除模型则需要从主键中取回模型：User::find(1)->delete() 这样来删除模型实例；

或者直接使用User::destroy([1, 2, 3])这样直接删除模型实例。

1. 如果要启用软删除，则在User模型中启用 protected $dates=[‘deleted\_at’]然后delete() 或 destroy()方法都是软删除，永久的删除模型是要 $user->forceDelete()强制删除一个模型，$user->history()->forceDelete()删除所有相关的模型；；恢复删除的数据则是：User::withTrashed()->where(‘name’, ‘Jackey’)->restore(); 或者$user->history()->restore(); 获取删除的数据用： User::history()->withTrashed()->get