38.理解中间件

学习要点：

1.路由中间件

本节课我们来开始学习中间件的使用方法。

一. 路由中间件

1. 什么是中间件？中间件就是当程序接收 HTTP 请求时，拦截后进行过滤和处理；
2. 比如当用户登录时，可以通过中间件进行验证比对，错误后让其跳转到登录页面；
3. 框架系统自带了一些中间件，比如之前 CSRF 令牌保护，就是中间件实现的；
4. 如果创建一个自定义的中间件？可以通过一句命令创建一个 check 中间件；

```
php artisan make:middleware Check
```

5. 实现一个简单的登录身份验证的效果，首先创建一个 Login 控制器；

```
class LoginController
{
    public function index()
    {
        echo '管理员，您好！';
    }

    public function login()
    {
        echo '登录失败！';
    }
}

Route::get('/admin','LoginController@index');
Route::get('/login','LoginController@login');
```

6. 我们要求，当 Http 接受 id=1 的情况下，它就是管理员，跳转 index 否则 login；
7. 此时，我们打开一开始创建好的 Check 中间件文件，了解一下结构；

```
//固定方法，固定格式
public function handle($request, Closure $next)
{
    //这里编写验证跳转方法
    if ($request->get('id') != 1) {
        return redirect(url('/login'));
    }
    //固定返回格式，让其继续往下执行
    return $next($request);
}
```

8. 下面，我们要对中间件进行注册，才可以使用，这个中间件适合在路由上使用；
9. 所以，我们打开 Http/Kernel.php 文件，在路由配置中间件的区域进行注册；

```
protected $routeMiddleware = [
    'check'=>\App\Http\Middleware\Check::class,
]
```

10. 然后，我们在登录的路由上执行中间件即可完成登录验证；

```
Route::get('/admin','LoginController@index')->middleware('check');
```

11. 这种中间件，属于前置中间件，也就是先拦截 Http 请求，再执行主体代码；
12. 另一种，就是后置中间件，也就是先执行主体代码，再拦截处理；

```
//固定方法，固定格式
public function handle($request, Closure $next)
{
    //先执行主体代码
    $response = $next($request);

    //再进行拦截 Http 请求处理
    echo '我是后置中间件';

    //固定格式返回
    return $response;
}
```