52. 伪静态.参数绑定.请求缓存

学习要点：

1.伪静态
2.参数绑定
3.请求缓存

本节课我们来学习一下伪静态的设置，操作方法的参数绑定，以及响应输出缓存。

一. 伪静态

1. 可以通过 route.php 修改伪静态的后缀，比如修改成 shtml、xml 等；

```
'url_html_suffix'       => 'html',
```

2. 如果地址栏用后缀访问成功后，可以使用 Request::ext()方法得到当前伪静态；

```
return Request::ext();
```

3. 配置文件伪静态后缀，可以支持多个，用竖线隔开；

```
'url_html_suffix'       => 'shtml|xml|pdf',
```

4. 直接将伪静态配置文件设置为 false，则关闭伪静态功能；

```
'url_html_suffix'       => false,
```

二. 参数绑定

1. 参数绑定功能：即 URL 地址栏的数据传参，我们一直在使用的功能；

```
public function get($id)
{
    return 'get:'.$id;
}
```

2. 操作方法 URL：/get，而带上 id 参数后，则为：/get/id/5；
3. 如果缺少了/5 或者缺少了/id/5，则都会报错方法参数错误：id；
4. 那么解决方案，就是给$id = 0 一个默认值，防止 URL 参数错误；
5. 如果设置了两个参数，那么参数传递的执行顺序可以设置，比如；

```
public function get($id, $name)
{
    return 'get:'.$id.' , '.$name;
}
```

6. 不管是：/id/5/name/lee，还是：/name/lee/id/5，都不会产生错误；

三．请求缓存

1. 请求缓存仅对 GET 请求有效，可以在全局和局部设置缓存；
2. 如果要设置全局请求缓存，在中间件文件 middleware.php 中设置；

```
'think\middleware\CheckRequestCache',
```

3. 然后在 route.php 中设置缓存的声明周期即可；

```
'request_cache_expire'   => 3600,
```

4. 当第二次访问时，会自动获取请求缓存的数据响应输出，并发送 304 状态码；
5. 如果要对路由设置一条缓存，直接使用 cache(3600)方法；

```
Route::get('get/:id', 'Rely/get')->cache(3600);
```