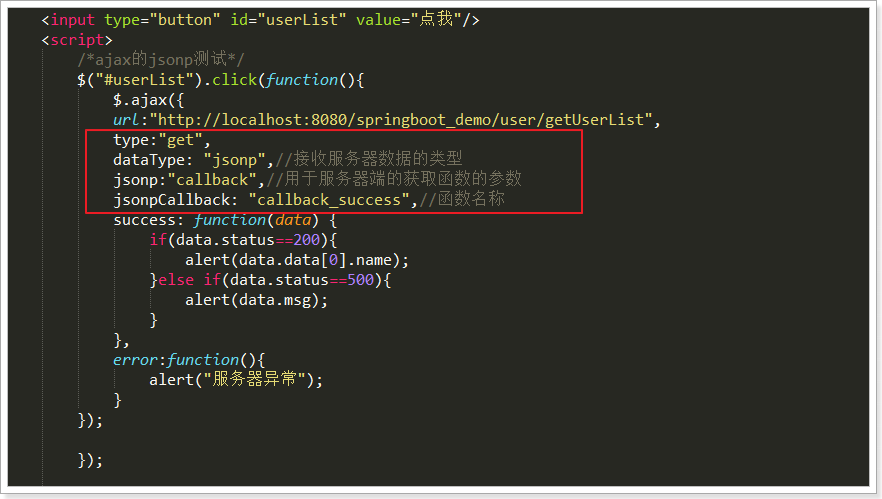
# 什么是跨域？

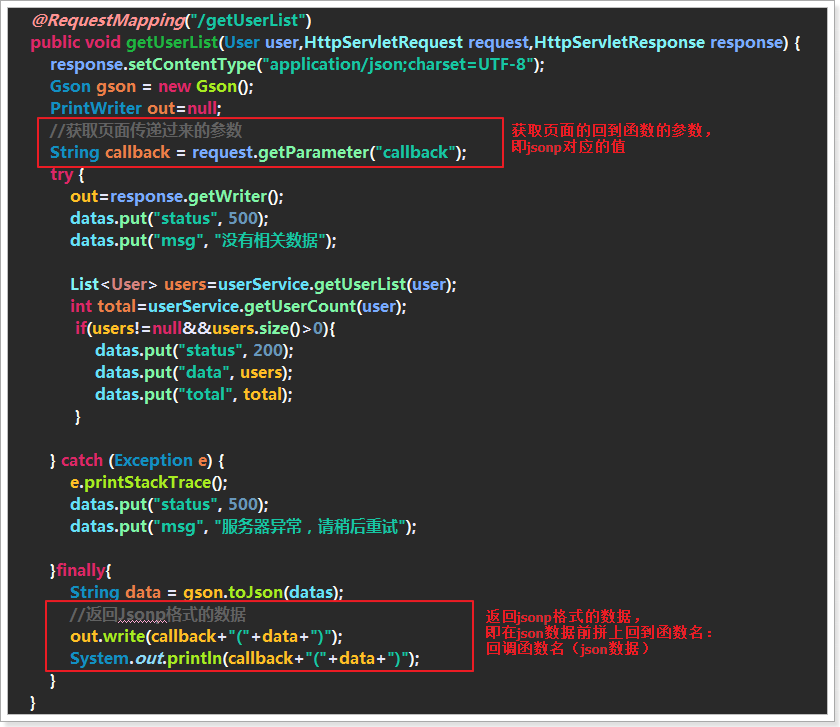
由于浏览器同源策略，凡是发送请求url的协议、域名、端口三者之间任意一个与当前页面地址不同即为跨域。存在跨域的情况：

* 网络协议不同，如http协议访问https协议。
* 端口不同，如80端口访问8080端口。
* 域名不同，如qianduanblog.com访问baidu.com。
* 子域名不同，如abc.qianduanblog.com访问def.qianduanblog.com。
* 域名和域名对应ip,如www.a.com访问20.205.28.90.

# 跨域的解决方案?

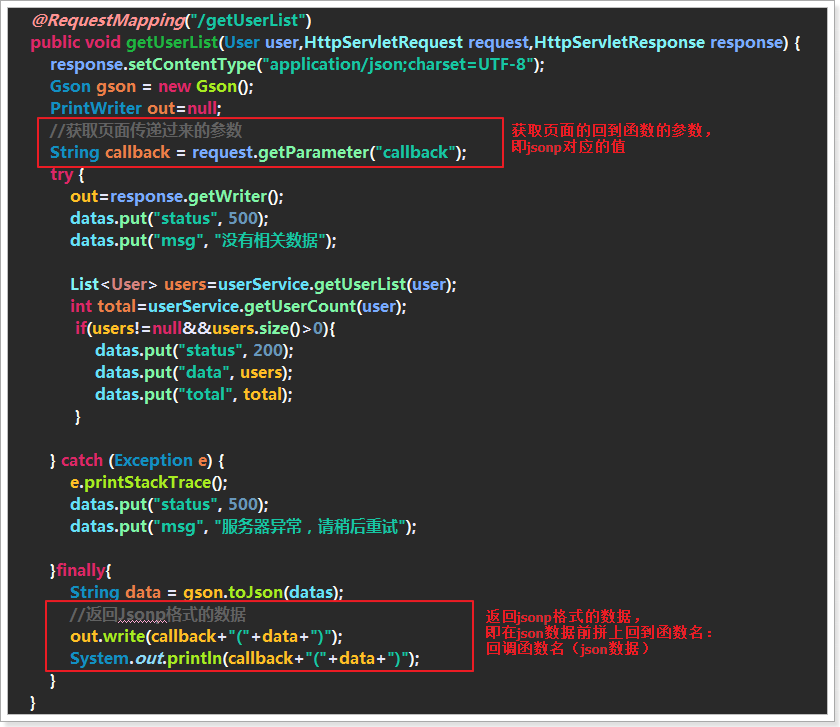
1. 通过jsonp跨域  
   2、 跨域资源共享（CORS）  
   3、 nginx代理跨域  
   4、 nodejs中间件代理跨域  
   9、 WebSocket协议跨域

方式一：使用ajax的jsonp  
前端代码  
[](https://images2015.cnblogs.com/blog/1135647/201706/1135647-20170607172018200-1525203141.png)

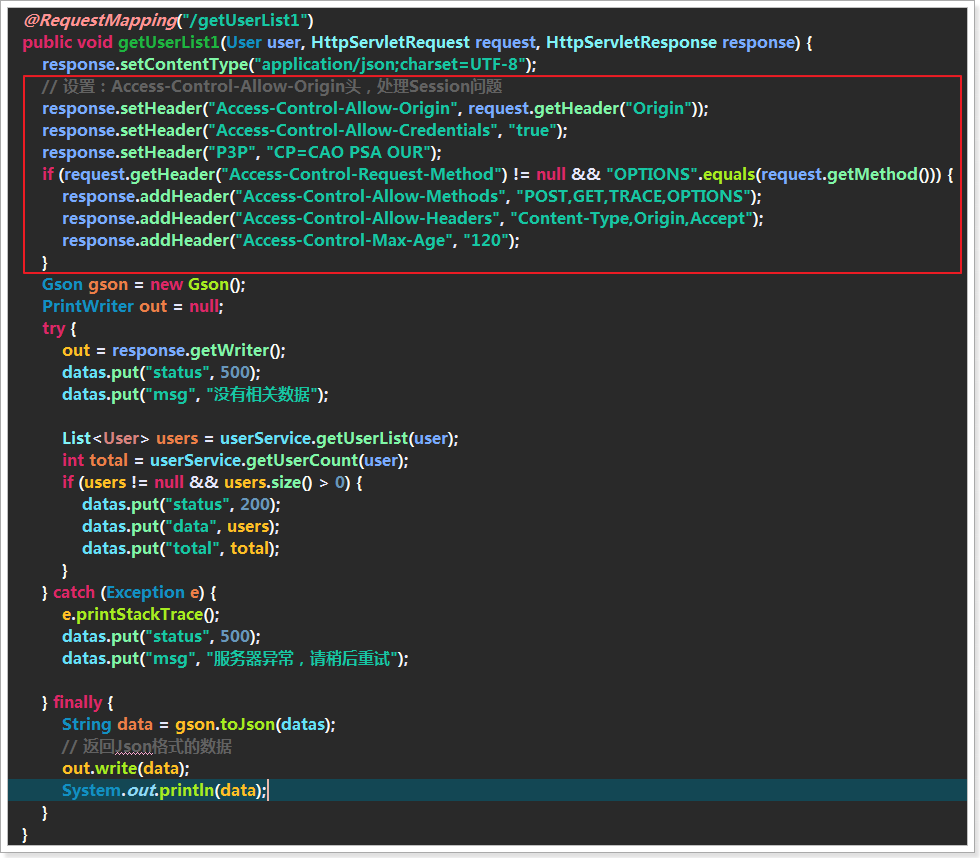
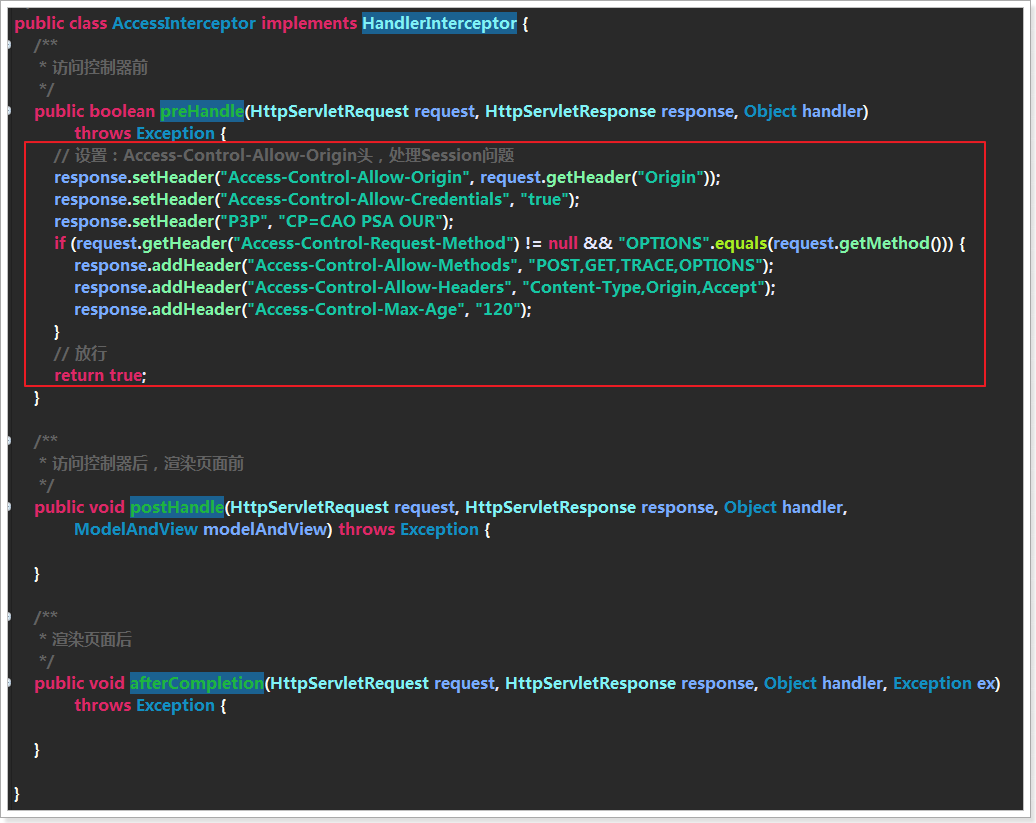
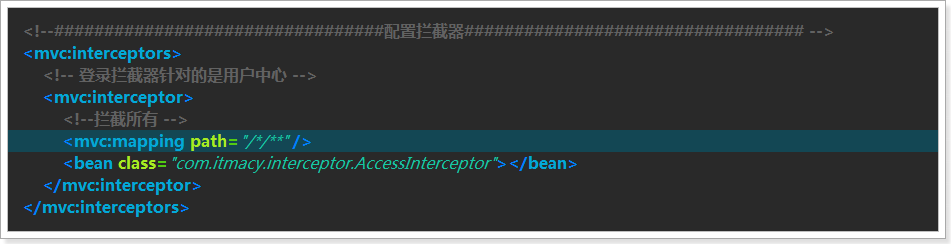
服务器代码  
[](https://images2015.cnblogs.com/blog/1135647/201706/1135647-20170607172045528-363980482.png)  
 使用该方式的缺点：请求方式只能是get请求

方式二：使用jQuery的jsonp插件

插件下载网址：[https://github.com/jaubourg/jquery-jsonp](https://github.com/jaubourg/jquery-jsonp" \o "jsonp插件下载" \t "https://www.cnblogs.com/itmacy/p/_blank)  
前端代码  
[](https://images2015.cnblogs.com/blog/1135647/201706/1135647-20170607172231606-1668496813.png)

服务器代码  
[](https://images2015.cnblogs.com/blog/1135647/201706/1135647-20170607172301622-622679626.png)  
 使用该方式的特点：与方式一相比，请求方式不只局限于get请求，还可以是post请求，但从服务器获取的数据依然是jsonp格式

方式三：使用cors  
前端代码  
[](https://images2015.cnblogs.com/blog/1135647/201706/1135647-20170607172341575-2077889877.png)

服务器代码  
[](https://images2015.cnblogs.com/blog/1135647/201706/1135647-20170607172406418-1975877347.png)  
使用该方式的特点：与前两种方式相比，前端代码和未处理跨域前一样，即普通的ajax请求，但服务器代码添加了一段解决跨域的代码  
    // 设置：Access-Control-Allow-Origin头，处理Session问题  
        response.setHeader("Access-Control-Allow-Origin", request.getHeader("Origin"));  
        response.setHeader("Access-Control-Allow-Credentials", "true");  
        response.setHeader("P3P", "CP=CAO PSA OUR");  
        if (request.getHeader("Access-Control-Request-Method") != null && "OPTIONS".equals(request.getMethod())) {  
            response.addHeader("Access-Control-Allow-Methods", "POST,GET,TRACE,OPTIONS");  
            response.addHeader("Access-Control-Allow-Headers", "Content-Type,Origin,Accept");  
            response.addHeader("Access-Control-Max-Age", "120");  
        }  
cors高级使用：在springmvc中配置拦截器  
 创建跨域拦截器实现HandlerInterceptor接口，并实现其方法，在请求处理前设置头信息，并放行.  
[](https://images2015.cnblogs.com/blog/1135647/201706/1135647-20170607172500653-793475022.png)  
在springmvc的配置文件中配置拦截器，注意拦截的是所有的文件  
[](https://images2015.cnblogs.com/blog/1135647/201706/1135647-20170607172515887-1312141011.png)

### Springboot跨域注解 @CrossOrigin

在controller控制类上方加注解;



**spring注解@CrossOrigin不起作用的原因**

1、是springMVC的版本要在4.2或以上版本才支持@CrossOrigin.

2、非@CrossOrigin没有解决跨域请求问题，而是不正确的请求导致无法得到预期的响应，导致浏览器端提示跨域问题。

3、在Controller注解上方添加@CrossOrigin注解后，仍然出现跨域问题，解决方案之一就是：

在@RequestMapping注解中没有指定Get、Post方式，具体指定后，问题解决。

## **方式四: 使用nginx代理**

**原先：**

调试页面是：<http://192.168.1.100:8080/>

请求的接口是：<http://ni.hao.sao/api/get/info>

**步骤一：**

请求的接口是：[http://192.168.1.100:8080/api/get/info](http://192.168.1.100:8080/api/get/info" \t "https://www.cnblogs.com/lovesong/p/_blank)

PS：这样就解决了跨域问题。

**步骤二：**

安装好Nginx后，去到/usr/local/etc/nginx/目录（这是Mac的），修改nginx.conf文件。

**步骤三：**

把默认的server配置注释掉。

在下面增加：

|  |
| --- |
| server{  listen 8888;  server\_name 192.168.1.100;    location /{  proxy\_pass http://192.168.1.100:8080;  }    location /api{  proxy\_pass http://ni.hao.sao/api;  }  } |

保存后，启动Nginx。

PS：并不需要太了解Nginx的配置，很简单的。

**步骤四：**

访问：<http://192.168.1.100:8888/>

搞定。

PS：注意访问的端口是‘8888’，有其他域的地址继续加location就行了。