itmacy

昵称：itmacy
园龄：1年7个月
粉丝：8
关注：31
+加关注

< 2018年11月 >

日	一	二	三	四	五	六
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	1
2	3	4	5	6	7	8

跨域问题：解决跨域的三种方案

当前端页面与后台运行在不同的服务器时，就必定会出现跨域这一问题，本篇简单介绍解决跨域的三种方案，部分代码截图如下，仅供参考：

方式一：使用ajax的jsonp

前端代码

我的评论
我的参与
最新评论
我的标签

我的标签

i18n(1)
idea(1)
properties中文乱码(1)
springboot(1)

随笔分类

hibernate
javaee
javase
maven(3)
spring
springboot(2)

```
<input type="button" id="userList" value="点我"/>
<script>
    /*ajax的jsonp测试*/
    $("#userList").click(function(){
        $.ajax({
        url:"http://localhost:8080/springboot_demo/user/getUserList",
        type:"get",
        dataType: "jsonp",//接收服务器数据的类型
        jsonp:"callback",//用于服务器端的获取函数的参数
        jsonpCallback: "callback_success",//函数名称
        success: function(data) {
            if(data.status==200){
                alert(data.data[0].name);
            }else if(data.status==500){
                alert(data.msg);
            }
        },
        error:function(){
            alert("服务器异常");
        }
    });

    });
```

服务器代码

struts2

前后端交互(1)

随笔档案

2017年11月 (1)

2017年6月 (3)

2017年5月 (2)

2017年3月 (1)

最新评论

1. Re:跨域问题：解决跨域的三种方案

@田想兵谢谢提醒。。。...

--itmacy

2. Re:跨域问题：解决跨域的三种方案

加一个，可以用ngix代理

--田想兵

3. Re:跨域问题：解决跨域的三种方案

mark

```java
@RequestMapping("/getUserList")
public void getUserList(User user,HttpServletRequest request,HttpServletResponse response) {
    response.setContentType("application/json;charset=UTF-8");
    Gson gson = new Gson();
    PrintWriter out=null;
    //获取页面传递过来的参数
    String callback = request.getParameter("callback");
    try {
        out=response.getWriter();
        datas.put("status", 500);
        datas.put("msg", "没有相关数据");

        List<User> users=userService.getUserList(user);
        int total=userService.getUserCount(user);
        if(users!=null&&users.size()>0){
            datas.put("status", 200);
            datas.put("data", users);
            datas.put("total", total);
        }

    } catch (Exception e) {
        e.printStackTrace();
        datas.put("status", 500);
        datas.put("msg", "服务器异常，请稍后重试");

    }finally{
        String data = gson.toJson(datas);
        //返回Jsonp格式的数据
        out.write(callback+"("+data+")");
        System.out.println(callback+"("+data+")");
    }
}
```

获取页面的回到函数的参数，即jsonp对应的值

返回jsonp格式的数据，即在json数据前拼上回到函数名：回调函数名（json数据）

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

--我在马路边

前端代码

```
<head>
    <meta charset="UTF-8">
    <title></title>                                    引入js文件
    <script type="text/javascript" src="js/jquery-1.12.4.min.js"></script>
    <script type="text/javascript" src="js/jquery.jsonp.js"></script>
</head>

<body>
    <input type="button" id="userList" value="点我" />
    <script>
        /*jQuery的jsonp插件测试*/
        $("#userList").click(function() {
            $.jsonp({
                url: "http://localhost:8080/springboot_demo/user/getUserList",
                callbackParameter:"callback",//设置传到后台方法的参数
                callback: "callback_success",//设置函数名
                success: function(data) {
                    if(data.status == 200) {
                        alert(data.data[0].name);
                    } else if(data.status == 500) {
                        alert(data.msg);
                    }
                },
                error: function(xOptions, textStatus) {
                    alert("服务器异常");
                }
            });
        });
    </script>
</body>
```

服务器代码

```
@RequestMapping("/getUserList")
public void getUserList(User user,HttpServletRequest request,HttpServletResponse response) {
    response.setContentType("application/json;charset=UTF-8");
    Gson gson = new Gson();
    PrintWriter out=null;
    //获取页面传递过来的参数
    String callback = request.getParameter("callback");
    try {
        out=response.getWriter();
        datas.put("status", 500);
        datas.put("msg", "没有相关数据");

        List<User> users=userService.getUserList(user);
        int total=userService.getUserCount(user);
         if(users!=null&&users.size()>0){
            datas.put("status", 200);
            datas.put("data", users);
            datas.put("total", total);
        }

    } catch (Exception e) {
        e.printStackTrace();
        datas.put("status", 500);
        datas.put("msg", "服务器异常，请稍后重试");

    }finally{
        String data = gson.toJson(datas);
        //返回Jsonp格式的数据
        out.write(callback+"("+data+")");
        System.out.println(callback+"("+data+")");
    }
}
```

获取页面的回到函数的参数，即jsonp对应的值

返回jsonp格式的数据，即在json数据前拼上回到函数名：回调函数名（json数据）

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，但从服务器从获取的数据依然是jsonp格式

方式三：使用cors

前端代码

```
<input type="button" id="userList" value="点我" />
<script>
    /*cors测试*/
    $("#userList").click(function() {
        alert("ok");
        $.ajax({
            url: "http://localhost:8080/springboot_demo/user/getUserList1",
            type: "GET",
            dataType: "json",
            success: function(data) {
                if(data.status == 200) {
                    alert("名字："+data.data[0].name);
                    console.log("名字："+data.data[0].name);
                } else if(data.status == 500) {
                    alert(data.msg);
                }
            },
            error: function() {
                alert("服务器异常");
            }
        });

    });
</script>
```

服务器代码

```
@RequestMapping("/getUserList1")
public void getUserList1(User user, HttpServletRequest request, HttpServletResponse response) {
    response.setContentType("application/json;charset=UTF-8");
    // 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");
    }
    Gson gson = new Gson();
    PrintWriter out = null;
    try {
        out = response.getWriter();
        datas.put("status", 500);
        datas.put("msg", "没有相关数据");

        List<User> users = userService.getUserList(user);
        int total = userService.getUserCount(user);
        if (users != null && users.size() > 0) {
            datas.put("status", 200);
            datas.put("data", users);
            datas.put("total", total);
        }
    } catch (Exception e) {
        e.printStackTrace();
        datas.put("status", 500);
        datas.put("msg", "服务器异常，请稍后重试");

    } finally {
        String data = gson.toJson(datas);
        // 返回Json格式的数据
        out.write(data);
        System.out.println(data);
    }
}
```

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

```java
public class AccessInterceptor implements HandlerInterceptor {
    /**
     * 访问控制器前
     */
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)
            throws Exception {
        // 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");
        }
        // 放行
        return true;
    }

    /**
     * 访问控制器后，渲染页面前
     */
    public void postHandle(HttpServletRequest request, HttpServletResponse response, Object handler,
            ModelAndView modelAndView) throws Exception {

    }

    /**
     * 渲染页面后
     */
    public void afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex)
            throws Exception {

    }

}
```

在springmvc的配置文件中配置拦截器，注意拦截的是所有的文件

```
<!--###################################配置拦截器#################################### -->
<mvc:interceptors>
  <!-- 登录拦截器针对的是用户中心 -->
  <mvc:interceptor>
    <!--拦截所有 -->
    <mvc:mapping path="/*/**"/>
    <bean class="com.itmacy.interceptor.AccessInterceptor"></bean>
  </mvc:interceptor>
</mvc:interceptors>
```

分类: 前后端交互

itmacy
关注 - 31
粉丝 - 8
+加关注

2 0

« 上一篇：maven问题：如何启动maven项目
» 下一篇：springboot问题：解决异常Unable to start embedded container;

posted @ 2017-06-07 17:41 itmacy 阅读(39314) 评论(3) 编辑 收藏

评论列表

#1楼 2017-06-07 17:52 我在马路边

mark

支持(0) 反对(0)

#2楼 2017-06-07 18:20 田想兵

加一个，可以用ngix代理

支持(0) 反对(0)

#3楼[楼主] 2017-06-07 18:57 itmacy

@ 田想兵
谢谢提醒。。。

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

相关博文：
· jquery ajax中使用jsonp的限制
· 前端跨域问题，以及ajax，jsonp，json的区别

- ajax原理，跨域，有哪些解决方法
- cookie跨域问题汇总
- 跨域及前后端分离

最新新闻:

- 库克：苹果正在开发自动驾驶汽车软件系统
- 2018，创业黄金时代结束的一年
- 音集协下架六千首歌曲背后：被质疑分配机制不公
- 新一代iPhone需求低迷 苹果被迫恢复生产iPhone X
- 微软亚洲研究院院长洪小文：科技创新要回归人本位

» 更多新闻...