Javascript的跨域请求策略

什么是跨域请求?

A网站想要获取B网站服务器上的资源，网站通过AJAX发送请求的时候，本地服务器地址与请求地址、协议类型（http）、IP地址（域名）、端口，三者有其一不同都称之为跨域请求资源。

http://www.baidu.com:80/

协议	http
域名	www.baidu.com
端口	80

为什么不能跨域请求?

浏览器有一个同源策略，用来保护用户的安全。　　所谓同源策略，它是浏览器的一种最核心最基本的安全策略。它对来至不同源的文档或这脚本对当前文档的读写操作做了限制。如果没有这个策略的话，a网站就可以操作b网站的页面，这样将会导致b网站的页面发生混乱，甚至信息被获取，包括服务器端发来的session。

URL	结果	原因
http://a.b.com/image.png	成功	
http://a.b.com/res/a.png	成功	
https://a.b.com/res/b.png	失败	协议不同
http://a.b.com:8080/res/b.png	失败	端口不同
http://g.b.com/res/c.png	失败	主机名不同

在这里需要注意的是，文档中的所有带“src”属性的标签都可以跨域加载资源，而不受同源策略的限制。如script、img、iframe、link等标签

如何实现跨域请求

> 原理分析：浏览器是可以使用script标签引入不同域下的JS文件,利用这个特性来实现跨域请求.

> 流程步骤：

1.在a.com页面中,添加一个script标签,src属性为b.com网站的url,并且传入一个callback参数

2.b.com网站接受到请求后,生成一段可执行的JS代码,调用a.com网站的写好的JS函数。

代码实现

- 第一步创建**2**个不同域的项目
  项目一：http://localhost:9090/pro1/
  项目二：http://localhost:8080/pro2/
- 在项目一种编写接受请求的**Servlet**

```
public class DemoDemo1 extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        this.doPost(request, response);
    }

    /**
     * 返回JSON数据
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //创建JSONObject对象
        JSONObject jsonObject = new JSONObject();

        //添加JSON数据
        jsonObject.put("username", "zs");
        jsonObject.put("password", 123);

        //使用Response将JSON数据输出到浏览器
        response.getWriter().print(jsonObject);
    }

}
```

- 配置**web.xml**

```
<servlet>
  <servlet-name>DemoDemo1</servlet-name>
  <servlet-class>cn.itcast.web.servlet.DemoDemo1</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>DemoDemo1</servlet-name>
  <url-pattern>/DemoDemo1</url-pattern>
</servlet-mapping>
```

- 在项目二中编写跨域的请求

```
<!-- 引入JQuery文件 -->
<script type="text/javascript" src="/pro2/js/jquery-1.8.3.js"></script>
<script type="text/javascript">
    /**使用Jquery向项目一发起Ajax请求*/
    $.get("http://localhost:8080/pro1/DemoDemo1",function(data){
        console.log(data);
    });
</script>
```

> 项目二浏览器控制台输出的结果
>
> Failed to load http://localhost:8080/pro1/DemoDemo1: No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'http://localhost:9090' is therefore not allowed access. (index):1

控制台结果显示该访问是跨域请求,浏览器不允许直接访问.

- 修改项目二的访问策略

```
<!-- 引入JQuery文件 -->
<script type="text/javascript" src="/pro2/js/jquery-1.8.3.js"></script>
<script type="text/javascript">
    /*动态创建一个Script标签来实现跨域请求 */
    //创建一个script标签
    var script = document.createElement("script");
    //添加script的src属性,src属性值为项目一的Servlet访问路径,追加一个callback参数
    script.setAttribute("src","http://localhost:8080/pro1/DemoDemo1?callback=show");
    //将script标签添加到HTML文档中
    document.head.appendChild(script);

    //编写与callback参数值同名的函数
    function show(data){
        console.log(data);
    }
</script>
```

- 修改项目一的**Servlet**代码,接受**callback**参数

```
public class DemoDemo1 extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        this.doPost(request, response);
    }

    /**
     * 返回JSON数据
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //创建JSONObject对象
        JSONObject jsonObject = new JSONObject();

        //添加JSON数据
        jsonObject.put("username", "zs");
        jsonObject.put("password", 123);

        //获得callback参数
        String callback = request.getParameter("callback");

        //使用Response将输出一个JS语句表达式
        response.getWriter().print(callback+"("+jsonObject+")");

    }

}
```

> 项目二浏览器控制台输出的结果
>
> ▶ {username: "zs", password: 123}

总结:

以上的解决跨域请求的方式也被称为JSONP(JSON with Padding)方式。
JQuery提供了一种简化版的JSONP的实现方式，服务器端代码不变，修改前端代码即可。
使用Jquery异步请求API中的getJson方法进行实现

```
<!-- 引入JQuery文件 -->
<script type="text/javascript" src="/pro2/js/jquery-1.8.3.js"></script>
<script type="text/javascript">
    //调用JQuery中的getJSON方式实现跨域请求
    $.getJSON("http://localhost:8080/pro1/DemoDemo1?callback=?",function(data){
        console.log(data);
    });
</script>
```

JQuery的JSONP实现方式也是在请求路径后面跟了一个callback参数，该参数名不一定为callback，只要和服务器端获得的参数名一样即可，callback的传递的参数值为"?"，没有创建callback回调函数名称，这是由于当请求--->响应回来的时候Jquery会自动调用function(data)匿名函数的原故。