

跨域

相关概念

相关概念											
<p>同源策略是浏览器的一种安全策略，所谓同源是指域名，协议，端口完全相同，只有同源的地址才可以相互通过AJAX的方式请求。</p> <p>同源或者不同源说的是两个地址之间的关系，不同源地址之间请求我们称之为跨域请求</p> <p>同源策略限制以下几种行为：</p> <ul style="list-style-type: none">1.) Cookie、LocalStorage 和 IndexDB 无法读取2.) DOM 和 Js对象无法获得3.) AJAX 请求不能发送 <p>什么是同源？</p> <p>常见跨域场景</p> <table><thead><tr><th>URL</th><th>说明</th><th>是否允许通信</th></tr></thead><tbody><tr><td>http://www.domain.com/a.js</td><td></td><td></td></tr><tr><td>http://www.domain.com/b.js</td><td>同一域名，不同文件或路径</td><td>允许</td></tr></tbody></table>			URL	说明	是否允许通信	http://www.domain.com/a.js			http://www.domain.com/b.js	同一域名，不同文件或路径	允许
URL	说明	是否允许通信									
http://www.domain.com/a.js											
http://www.domain.com/b.js	同一域名，不同文件或路径	允许									

相关概念		
http://www.domain1.com/a.js		
http://www.domain2.com/b.js	不同域名	不允许
跨域解决方案		
1、通过jsonp跨域		
2、document.domain + iframe跨域		
3、location.hash + iframe		
4、window.name + iframe跨域		
5、postMessage跨域		
6、跨域资源共享（CORS）		
7、nginx代理跨域		
8、nodejs中间件代理跨域		
9、WebSocket协议跨域		

JSONP

JSONP

跨域的安全限制都只对浏览器端来说的，服务器端是不存在跨域安全限制的。

浏览器的同源策略限制从一个源加载的文档或脚本与来自另一个源的资源进行交互。

如果协议，端口和主机对于两个页面是相同的，则两个页面具有相同的源，否则就是不同源的。

如果要在js里发起跨域请求，则要进行一些特殊处理了。或者，你可以把请求发到自己的服务端，再通过后台代码发起请求，再将数据返回前端。

jsonp:

jsonp 全称是JSON with Padding,是为了解决跨域请求资源而产生的解决方案,是一种依靠开发人员创造出的一种非官方跨域数据交互协议。

一个是描述信息的格式，一个是信息传递双方约定的方法。

jsonp的产生:

JSONP

一个是描述信息的格式，一个是信息传递双方约定的方法。

jsonp的产生:

1. AJAX直接请求普通文件存在跨域无权限访问的问题,不管是静态页面也好.
2. 不过我们在调用js文件的时候又不受跨域影响,比如引入jquery框架的,或者是调用相片的时候
3. 凡是拥有src这个属性的标签都可以跨域例如<script><iframe>
4. 如果想通过纯web端跨域访问数据只有一种可能,那就是把远程服务器上的数据装进js格式的文件里.
5. 而json又是一个轻量级的数据格式,还被js原生支持

JSONP-A中index.jsp

```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<html>
<head>
<title>A跨域请求</title>
<script src="js/jquery-1.12.4.js"></script>
<script type="text/javascript">
    $(function () {
        $("#btn").click(function () {
            //发送一个get请求
            $.get(
                "http://localhost:8888/JSONP-B/goodsServlet",
                function (data) {
                    $("#show").append(data);
                },
                "json"
            )
        })
    })
})
```

```

    })
  </script>
</head>
<body>
  <input type="button" id="btn" value="A发送跨域请求">
  <div id="show">

  </div>
  
</body>
</html>

```

JSONP

```

$("#btn").click(function () {
  $.ajax({
    url: 'http://localhost:8080/B/student',
    type: 'GET',
    dataType: "jsonp", //指定服务器返回的数据类型
    jsonpCallback: "callback", //指定回调函数名称
    success: function (data) {
      var result = JSON.stringify(data); //json对象转成字符串
      $("#text").val(result);
    }
  });
});

```

JSONP-A中的index.jsp

```

<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<html>
<head>
  <title>A跨域请求</title>
  <script src="js/jquery-1.12.4.js"></script>
  <script type="text/javascript">
    $(function () {
      $("#btn").click(function () {
        $.get({
          url:"http://localhost:8888/JSONP-B/jsonServlet",
          dataType:"jsonp", //指定入服务器返回的数据类型
          jsonpCallback : "callback", //指定回调函数名称
          success:function (data) {
            var result = JSON.stringify(data);
            $("#show").text(result);
          }
        });
      });
    });
  </script>

```

```

        })
    })
</script>
</head>
<body>
    A跨域请求
    <input type="button" id="btn" value="A发送跨域请求">
    <div id="show">

    </div>
    
</body>
</html>

```

JSONP-B中的jsonServlet类中

```

@WebServlet( "/jsonServlet")
public class jsonServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        doGet(request, response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        //前端穿过来的回调函数的名称
        String callback = request.getParameter("callback");
        response.setContentType("text/json");
        response.setCharacterEncoding("utf-8" );
        PrintWriter out = response.getWriter();
        goodsService goodsService = new goodsServiceImpl();
        //获取商品列表
        List<Goods> goodsList = goodsService.findAllGoods();
        //将其转化为JSON字符串
        String strJson = JSON.toJSONString(goodsList);
        //回调函数名称包裹返回函数，返回函数作为回调参数传到前端
        strJson =callback + "("+strJson+")";
        //将其输出到jsp页面
        out.print(strJson);
        //关闭流
        out.close();
    }
}

```

JSONP

```
setInterval(function() {
    //用回调函数名称包裹返回数据，这样，返回数据就作为回调函数的参数传回去了
    jsonStr = callback + "(" + jsonStr + ")";
    out = response.getWriter();
    out.write(jsonStr);
    out.close();
}
```

```
{
  "employees": [
    { "firstName": "Bill", "lastName": "Gates" },
    { "firstName": "George", "lastName": "Bush" },
    { "firstName": "Thomas", "lastName": "Carter" } ]
}
```

跨域获取数据

问题:

1. JSONP 需要服务端配合，服务端按照客户端的要求返回一段 JavaScript 调用客户端的函数
2. 只能发送 GET 请求

CORS

CORS

CORS是一个W3C标准，全称是“跨域资源共享”（Cross-origin resource sharing）。它允许浏览器向跨源服务器，发出XMLHttpRequest请求，从而克服了AJAX只能同源使用的限制。

整个CORS通信过程，都是浏览器自动完成，不需要用户参与。对于开发者来说，CORS通信与同源的AJAX通信没有差别，代码完全一样。浏览器一旦发现AJAX请求跨源，就会自动添加一些附加的头信息，有时还会多出一次附加的请求，但用户不会有感觉。

这种方案无需客户端作出任何变化（客户端不用改代码），只是在被请求的服务端响应的时候添加一个 Access-Control-Allow-Origin 的响应头，表示这个资源是否允许指定域请求。

```
response.setHeader("Access-Control-Allow-Origin", "*");
```

JSONP-A中index.jsp中

```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<html>
<head>
<title>A跨域请求</title>
<script src="js/jquery-1.12.4.js"></script>
<script type="text/javascript">
    $(function () {
        $("#btn").click(function () {
            $.get({
                url: "http://localhost:8888/JSONP-B/jsonServlet",
                /*dataType: "jsonp", //指定入服务器返回的数据类型
                jsonpCallback: "callback", //指定回调函数名称*/
                success: function (data) {
                    var result = JSON.stringify(data);
                    $("#show").text(result);
                }
            })
        })
    })
</script>
```

```

</head>
<body>
    A跨域请求
    <input type="button" id="btn" value="A发送跨域请求">
    <div id="show">

    </div>
    
</body>
</html>

```

JSONP-B中jsonServlet类中

```

@WebServlet( "/jsonServlet")
public class jsonServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        doGet(request, response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        //前端穿过来的回调函数的名称
        // String callback = request.getParameter("callback");
        response.setContentType("text/json");
        response.setCharacterEncoding("utf-8" );

        response.setHeader("Access-Control-Allow-Origin","*");

        PrintWriter out = response.getWriter();
        goodsService goodsService = new goodsServiceImpl();
        //获取商品列表
        List<Goods> goodsList = goodsService.findAllGoods();
        //将其转化为JSON字符串
        String strJson = JSON.toJSONString(goodsList);
        //回调函数名称包裹返回函数，返回函数作为回调参数传到前端
        //strJson =callback + "("+strJson+")";
        //将其输出到jsp页面
        out.print(strJson);
        //关闭流
        out.close();
    }
}

```