

狂神说SpringMVC07：Ajax研究

秦疆 狂神说 2020-04-05

狂神说SpringMVC系列连载课程，通俗易懂，基于Spring5版本（视频同步），欢迎各位狂粉转发关注学习。未经授权，禁止转载

Ajax研究

简介

- **AJAX = Asynchronous JavaScript and XML**（异步的 **JavaScript** 和 **XML**）。
- AJAX 是一种在无需重新加载整个网页的情况下，能够更新部分网页的技术。
- **Ajax** 不是一种新的编程语言，而是一种用于创建更好更快以及交互性更强的**Web**应用程序的技术。
- 在 2005 年，Google 通过其 Google Suggest 使 AJAX 变得流行起来。Google Suggest能够自动帮你完成搜索单词。
- Google Suggest 使用 AJAX 创造出动态性极强的 web 界面：当您在谷歌的搜索框输入关键字时，JavaScript 会把这些字符发送到服务器，然后服务器会返回一个搜索建议的列表。
- 就和国内百度的搜索框一样！
- 传统的网页(即不用ajax技术的网页)，想要更新内容或者提交一个表单，都需要重新加载整个网页。
- 使用ajax技术的网页，通过在后台服务器进行少量的数据交换，就可以实现异步局部更新。
- 使用Ajax，用户可以创建接近本地桌面应用的直接、高可用、更丰富、更动态的Web用户界面。

伪造Ajax

我们可以使用前端的一个标签来伪造一个ajax的样子。iframe标签

- 1、新建一个module：sspringmvc-06-ajax，导入web支持！
- 2、编写一个 ajax-frame.html 使用 iframe 测试，感受下效果

```
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="UTF-8">
  <title>kuangshen</title>
</head>
```

```

<body>

<script type="text/javascript">
    window.onload = function(){
        var myDate = new Date();
        document.getElementById('currentTime').innerText = myDate.getTime();
    };

    function LoadPage(){
        var targetUrl = document.getElementById('url').value;
        console.log(targetUrl);
        document.getElementById("iframePosition").src = targetUrl;
    }

</script>

<div>
    <p>请输入要加载的地址: <span id="currentTime"></span></p>
    <p>
        <input id="url" type="text" value="https://www.baidu.com/" />
        <input type="button" value="提交" onclick="LoadPage()" />
    </p>
</div>

<div>
    <h3>加载页面位置: </h3>
    <iframe id="iframePosition" style="width: 100%;height: 500px;"></iframe>
</div>

</body>
</html>

```

3、使用IDEA开浏览器测试一下！

利用AJAX可以做：

- 注册时，输入用户名自动检测用户是否已经存在。
- 登陆时，提示用户名密码错误
- 删除数据行时，将行ID发送到后台，后台在数据库中删除，数据库删除成功后，在页面DOM中将数据行也删除。
-等等

jQuery.ajax

纯JS原生实现Ajax我们不去讲解这里，直接使用jquery提供的，方便学习和使用，避免重复造轮子，有兴趣的同学可以去了解下JS原生XMLHttpRequest！

Ajax的核心是XMLHttpRequest对象(XHR)。XHR为向服务器发送请求和解析服务器响应提供了接口。能够以异步方式从服务器获取新数据。

jQuery 提供多个与 AJAX 有关的方法。

通过 jQuery AJAX 方法，您能够使用 HTTP Get 和 HTTP Post 从远程服务器上请求文本、HTML、XML 或 JSON – 同时您能够把这些外部数据直接载入网页的被选元素中。

jQuery 不是生产者，而是大自然搬运工。

jQuery Ajax本质就是 XMLHttpRequest，对他进行了封装，方便调用！

```
jQuery.ajax(...)  
    部分参数:  
        url: 请求地址  
        type: 请求方式, GET、POST (1.9.0之后用method)  
        headers: 请求头  
        data: 要发送的数据  
        contentType: 即将发送信息至服务器的内容编码类型 (默认: "application/x-www-form-urlencoded;  
        charset=UTF-8")  
        async: 是否异步  
        timeout: 设置请求超时时间 (毫秒)  
        beforeSend: 发送请求前执行的函数 (全局)  
        complete: 完成之后执行的回调函数 (全局)  
        success: 成功之后执行的回调函数 (全局)  
        error: 失败之后执行的回调函数 (全局)  
        accepts: 通过请求头发送给服务器, 告诉服务器当前客户端可接受的数据类型  
        dataType: 将服务器端返回的数据转换成指定类型  
            "xml": 将服务器端返回的内容转换成xml格式  
            "text": 将服务器端返回的内容转换成普通文本格式  
            "html": 将服务器端返回的内容转换成普通文本格式, 在插入DOM中时, 如果包含JavaScript标签, 则会  
            尝试去执行。  
            "script": 尝试将返回值当作JavaScript去执行, 然后再将服务器端返回的内容转换成普通文本格式  
            "json": 将服务器端返回的内容转换成相应的JavaScript对象  
            "jsonp": JSONP 格式使用 JSONP 形式调用函数时, 如 "myurl?callback=?" jQuery 将自动替换 ?  
            为正确的函数名, 以执行回调函数
```

我们来个简单的测试，使用最原始的HttpServletResponse处理，.最简单，最通用

1、配置web.xml 和 springmvc的配置文件，复制上面案例的即可 【记得静态资源过滤和注解驱动配置上】

```
<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
    xmlns:context="http://www.springframework.org/schema/context"  
    xmlns:mvc="http://www.springframework.org/schema/mvc"  
    xsi:schemaLocation="http://www.springframework.org/schema/beans  
        http://www.springframework.org/schema/beans/spring-beans.xsd  
        http://www.springframework.org/schema/context  
        https://www.springframework.org/schema/context/spring-context.xsd  
        http://www.springframework.org/schema/mvc  
        https://www.springframework.org/schema/mvc/spring-mvc.xsd">
```

```

<!-- 自动扫描指定的包，下面所有注解类交给IOC容器管理 -->
<context:component-scan base-package="com.kuang.controller"/>
<mvc:default-servlet-handler />
<mvc:annotation-driven />

<!-- 视图解析器 -->
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver"
      id="internalResourceViewResolver">
    <!-- 前缀 -->
    <property name="prefix" value="/WEB-INF/jsp/" />
    <!-- 后缀 -->
    <property name="suffix" value=".jsp" />
</bean>

</beans>

```

2、编写一个AjaxController

```

@Controller
public class AjaxController {

    @RequestMapping("/a1")
    public void ajax1(String name , HttpServletResponse response) throws IOException {
        if ("admin".equals(name)) {
            response.getWriter().print("true");
        }else{
            response.getWriter().print("false");
        }
    }

}

```

3、导入jquery， 可以使用在线的CDN， 也可以下载导入

```

<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
<script src="${pageContext.request.contextPath}/statics/js/jquery-3.1.1.min.js"></script>

```

4、编写index.jsp测试

```

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>$Title$</title>
<!--<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>-->
<script src="${pageContext.request.contextPath}/statics/js/jquery-3.1.1.min.js"></script>
<script>
    function a1() {
        $.post({
            url:"${pageContext.request.contextPath}/a1",
            data:{'name':$("#txtName").val()},
            success:function (data,status) {
                alert(data);
            }
        });
    }

```

```

        alert(status);
    }
    });
}
</script>
</head>
<body>

<%--onblur: 失去焦点触发事件--%>
用户名:<input type="text" id="txtName" onblur="a1()" />

</body>
</html>

```

5、启动tomcat测试！打开浏览器的控制台，当我们鼠标离开输入框的时候，可以看到发出了一个ajax的请求！是后台返回给我们的结果！测试成功！

Springmvc实现

实体类user

```

@Data
@AllArgsConstructor
@NoArgsConstructor
public class User {

    private String name;
    private int age;
    private String sex;

}

```

我们来获取一个集合对象，展示到前端页面

```

@RequestMapping("/a2")
public List<User> ajax2() {
    List<User> list = new ArrayList<User>();
    list.add(new User("秦疆1号",3,"男"));
    list.add(new User("秦疆2号",3,"男"));
    list.add(new User("秦疆3号",3,"男"));
    return list; //由于@RestController注解，将list转成json格式返回
}

```

前端页面

```

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>

```

```

<input type="button" id="btn" value="获取数据"/>
<table width="80%" align="center">
    <tr>
        <td>姓名</td>
        <td>年龄</td>
        <td>性别</td>
    </tr>
    <tbody id="content">
    </tbody>
</table>

<script src="${pageContext.request.contextPath}/statics/js/jquery-3.1.1.min.js"></script>
<script>

    $(function () {
        $("#btn").click(function () {
            $.post("${pageContext.request.contextPath}/a2",function (data) {
                console.log(data)
                var html="";
                for (var i = 0; i <data.length ; i++) {
                    html+= "<tr>" +
                        "<td>" + data[i].name + "</td>" +
                        "<td>" + data[i].age + "</td>" +
                        "<td>" + data[i].sex + "</td>" +
                        "</tr>"
                }
                $("#content").html(html);
            });
        })
    })
</script>
</body>
</html>

```

成功实现了数据回显！可以体会一下Ajax的好处！

注册提示效果

我们再测试一个小Demo，思考一下我们平时注册时候，输入框后面的实时提示怎么做到的；如何优化

我们写一个Controller

```

@RequestMapping("/a3")
public String ajax3(String name,String pwd) {
    String msg = "";
    //模拟数据库中存数据
    if (name!=null){
        if ("admin".equals(name)) {
            msg = "OK";
        }else {

```

```

        msg = "用户名输入错误";
    }
}

if (pwd!=null) {
    if ("123456".equals(pwd)) {
        msg = "OK";
    }else {
        msg = "密码输入有误";
    }
}

return msg; //由于@RestController注解，将msg转成json格式返回
}

```

前端页面 login.jsp

```

<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<html>
<head>
    <title>ajax</title>
    <script src="${pageContext.request.contextPath}/statics/js/jquery-3.1.1.min.js"></script>
    <script>

        function a1() {
            $.post({
                url: "${pageContext.request.contextPath}/a3",
                data: { 'name': $("#name").val() },
                success: function (data) {
                    if (data.toString()=='OK') {
                        $("#userInfo").css("color", "green");
                    }else {
                        $("#userInfo").css("color", "red");
                    }
                    $("#userInfo").html(data);
                }
            });
        }

        function a2() {
            $.post({
                url: "${pageContext.request.contextPath}/a3",
                data: { 'pwd': $("#pwd").val() },
                success: function (data) {
                    if (data.toString()=='OK') {
                        $("#pwdInfo").css("color", "green");
                    }else {
                        $("#pwdInfo").css("color", "red");
                    }
                    $("#pwdInfo").html(data);
                }
            });
        }

    </script>

```

```
</head>
<body>
<p>
  用户名:<input type="text" id="name" onblur="a1()" />
  <span id="userInfo"></span>
</p>
<p>
  密码:<input type="text" id="pwd" onblur="a2()" />
  <span id="pwdInfo"></span>
</p>
</body>
</html>
```

【记得处理json乱码问题】

测试一下效果，动态请求响应，局部刷新，就是如此！

用户名: OK

密码: 密码输入有误

获取baidu接口Demo

```
<!DOCTYPE HTML>

<html>

<head>

  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

  <title>JSONP百度搜索</title>

  <style>

    #q{

      width: 500px;

      height: 30px;

      border:1px solid #ddd;

      line-height: 30px;

      display: block;

      margin: 0 auto;

      padding: 0 10px;

      font-size: 14px;

    }

    #ul{

      width: 520px;

      list-style: none;

      margin: 0 auto;

      padding: 0;

      border:1px solid #ddd;

      margin-top: -1px;
```



```

        display: none;
    }
    #ul li{
        line-height: 30px;
        padding: 0 10px;
    }
    #ul li:hover{
        background-color: #f60;
        color: #fff;
    }
</style>
<script>

```

// 2.步骤二

// 定义demo函数 (分析接口、数据)

```

function demo(data){
    var U1 = document.getElementById('ul');
    var html = '';
    // 如果搜索数据存在 把内容添加进去
    if (data.s.length) {
        // 隐藏掉的ul显示出来
        U1.style.display = 'block';
        // 搜索到的数据循环追加到li里
        for(var i = 0;i<data.s.length;i++){
            html += '<li>'+data.s[i]+'</li>';
        }
        // 循环的li写入ul
        U1.innerHTML = html;
    }
}

```

// 1.步骤一

```

window.onload = function(){
    // 获取输入框和ul
    var Q = document.getElementById('q');
    var U1 = document.getElementById('ul');

```

// 事件鼠标抬起时候

```

Q.onkeyup = function(){
    // 如果输入框不等于空
    if (this.value != '') {
        // ☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆JSONPz重点

```

☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆

// 创建标签

```

var script = document.createElement('script');

```

//给定要跨域的地址 赋值给src

//这里是要请求的跨域的地址 我写的是百度搜索的跨域地址

```

script.src = 'https://sp0.baidu.com/5a1Fazu8AA54nxGko9WTAnF6hhy/su?

```

wd='+this.value+'&cb=demo';

// 将组合好的带src的script标签追加到body里

```

document.body.appendChild(script);

```

```
        }  
    }  
}  
  
</script>  
</head>  
  
<body>  
<input type="text" id="q" />  
<ul id="ul">  
  
</ul>  
</body>  
</html>
```

Ajax在我们开发中十分重要，一定要学会使用！

end

视频同步更新，这次一定！



“赠人玫瑰，手有余香”

狂神说的赞赏码

 狂神说

