JavaScript： 浅讲ajax

1.ajax入门案例

1.1 搭建Web环境

ajax对于各位来说，应该都不陌生，正因为ajax的产生，导致前台页面和服务器之间的数据传输变得非常容易，同时还可以实现页面的局部刷新。通过在后台与服务器进行少量数据交换，AJAX可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

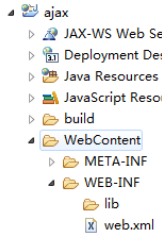
对于JavaWeb项目而言，ajax主要用于浏览器和服务器之间数据的传输。

如果是单单地堆砌知识点，会显得比较无聊，那么根据惯例，我先不继续介绍ajax，而是来写一个案例吧。

打开eclipse，新建一个web项目。

如果对JavaWeb项目还不太清楚的，可以去参考我之前的文章。我

目录结构：




```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>

</web-app>
```

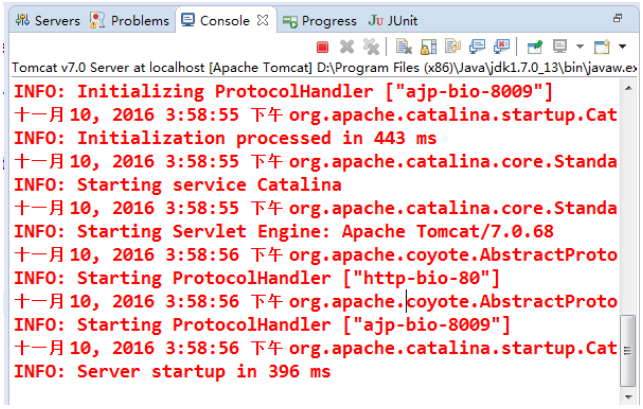
这样就好了，web项目搭建完毕。

暂时不要往下写，先确保我们到目前为止的工作是没问题的。

验证方法就是在WebContent目录下，新建一个空的jsp页面，里面随便写的什么。


```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body style="padding:100px">
<h1>Hello World</h1>
</body>
</html>
```

启动tomcat，把这个项目跑起来。



没报错。

打开浏览器，输入访问地址，我这里的tomcat端口号是80 ,默认可以不写。

http://localhost/ajax/index.jsp

Hello World

来了，没问题。

好的，这说明我们的web项目搭建没有问题。

1.2 编写服务器程序Servlet

个人感悟，精粹整理

web环境已经搭好，接下来，让我们来编写一个简单的Servlet程序，tomcat是一个服务器，现在它里面有一个名字叫做ajax的web项目，那么这些Servlet就好比是web项目里面的一个个小功能。

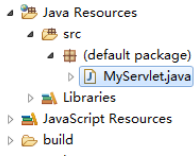
你的电脑里面有QQ，Word，杀毒软件等程序。一个web项目，也就是一个应用程序。本质上和你电脑上的QQ概念是一样一样的。

你打开QQ，可以聊天，语音，视频。这些小功能，类比到JavaWeb项目，就是一个个Servlet。

很多人都知道框架，比如大名鼎鼎的SpringMVC，里面有一个个的Controller，其实这些Controller到底是什么玩意，不要怕，他们其实就是对Servlet做了一个封装，本质上还是一样一样的。

我们写一个Servlet，都需要去web.xml里面注册一下，否则就用不了。你安装一个QQ，注册表里面是不是肯定也需要注册一下啊，这不还是一样一样的吗？

好了，闲话不多说。我们来写一个小功能，也就是一个Servlet。



继承HttpServlet，同时改写doGet方法

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class MyServlet extends HttpServlet{
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        System.out.println("lll");
    }
}
```

里面我们先什么也不写。

接下来，我们要在web.xml里面把这个Servlet注册一下。

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <servlet>
        <!-- 这里是servlet的名字 -->
        <servlet-name>MyServlet</servlet-name>
        <servlet-class>MyServlet</servlet-class>
        <!-- 这里写servlet类在的包路径 -->
    </servlet>

    <servlet-mapping>
        <!-- 这里是地址映射 -->
        <servlet-name>MyServlet</servlet-name>

        <!-- 这里写servlet映射地址 -->
        <url-pattern>/MyServlet</url-pattern>

    </servlet-mapping>

</web-app>
```

OK，Servlet注册完毕，我们先来访问一下这个功能。

重启tomcat。

访问：http://localhost/ajax/MyServlet

```
十一月 10, 2016 4:29:52 下午 org.apache.coyote.AbstractProto
INFO: Starting ProtocolHandler ["http-bio-80"]
十一月 10, 2016 4:29:52 下午 org.apache.coyote.AbstractProto
INFO: Starting ProtocolHandler ["ajp-bio-8009"]
十一月 10, 2016 4:29:52 下午 org.apache.catalina.startup.Cat
INFO: Server startup in 349 ms
111
```

1.3 前台页面设计

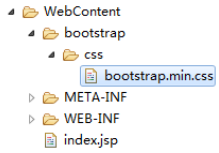
服务器小程序已经差不多了，现在我们为了和服务器进行交互，就需要编写前台页面。这个页面也就是给用户看的。换言之，用户只能通过前台页面来访问我们的Servlet。

我们来写一个小案例，在页面上发送一句话到服务器，然后服务器给出一个回应就行了。

就是这么一个简单的案例，主要用来熟悉一下流程。

为了简单起见，我就不自己调css样式了，直接用bootstrap吧。

引入bootstrap的核心css文件。



然后，修改index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<link rel="stylesheet" href="${pageContext.request.contextPath}/bootstrap/css/bootstrap.min.css">
<title>Insert title here</title>
<style type="text/css">
    .container {
        margin-top:100px;
    }
</style>
<script>

    window.onload = function(){
        var btn = document.getElementById("submit");
        btn.onclick = function(){
            alert();
        }
    };

</script>
</head>
<body>
    <div class="container">

        <div class="row">
          <div class="col-lg-8">
            <div class="input-group">
              <input type="text" class="form-control">
              <span class="input-group-btn">
                <button id="submit" class="btn btn-default" type="button">提交</button>
              </span>
            </div>
          </div>

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

页面效果：

1.4 基于get方式的数据请求

当我们点击提交按钮，就alert（）一下，如果成功的话，那么说明点击事件没有问题。然后，继续往下写代码。

如果是以往，我们都是通过form表单来进行提交的，可是这样的话，就会有一个问题，就是页面会刷新，而且代码也相对比较难懂。

自从ajax出来了之后，这种情况得到了巨大的改善，局部刷新技术在当时来看，还是非常不错的。

我先把实现代码给出：

```
btn.onclick = function(){
    var  xhr = window.XMLHttpRequest?new XMLHttpRequest():new ActiveXObject("Microsoft.XMLHTTP");
    xhr.open("get","MyServlet?message="+document.getElementsByTagName("input")[0].value,true);
    xhr.send();
    xhr.onreadystatechange = function(){
        if(xhr.readyState==4 && xhr.status==200){
            alert(xhr.responseText);
        }
    };
}
```

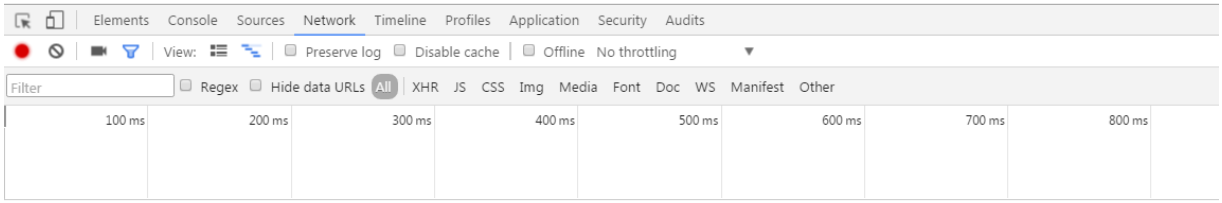
同时修改一下MyServlet

```
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
```

```
    String msg = req.getParameter("message");
    System.out.println(msg);
}
```

重启 tomcat，访问index.jsp页面。

现在的浏览器一般都有调试功能，按一下F12，调试界面就出来了。然后，找到一个network，以谷歌浏览器为例



Recording network activity...

Perform a request or hit **F5** to record the reload.

network视图会把所有的数据交互显示出来，包括引入的 js， css文件，还有各种请求和回应，都会在这里显示出来。

比如，现在我刷新一下页面

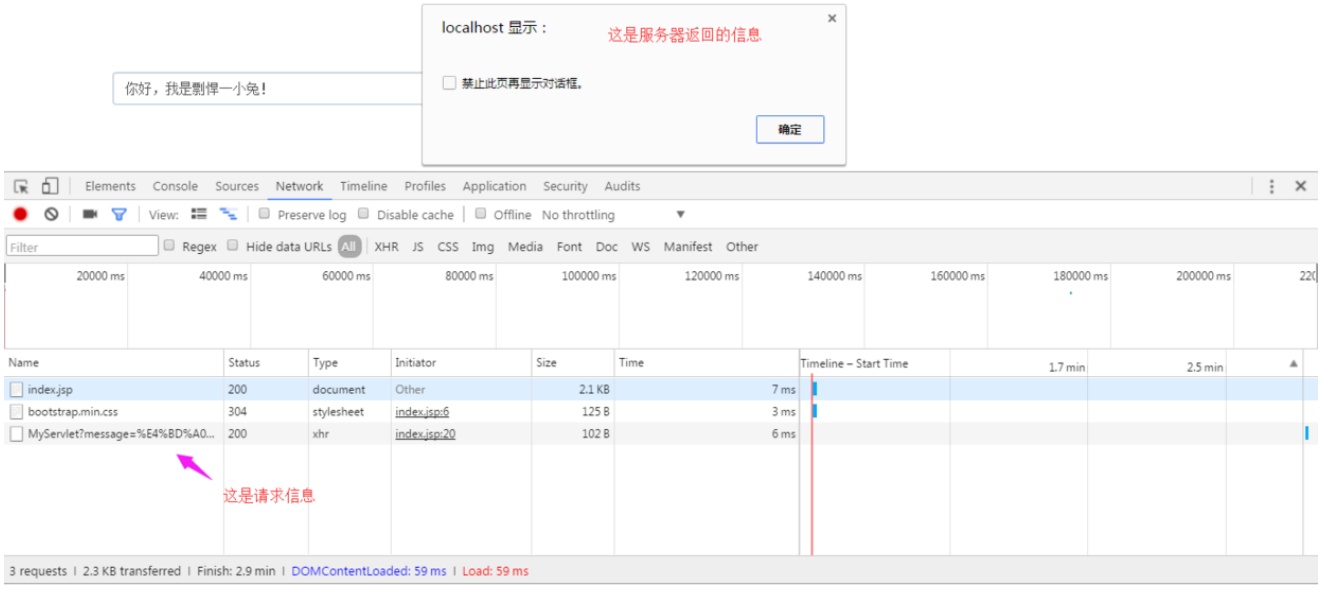
Name	Status	Type	Initiator	Size	Time
index.jsp	200	document	Other	1.7 KB	9 ms
bootstrap.min.css	304	stylesheet	index.jsp:6	125 B	125 ms

我这么一刷新，首先服务器接收到的是这么一个 URL： http://localhost/ajax/index.jsp

这就是一个请求，服务器收到这个请求后，返回给我 index.jsp页面和bootstrap.min.css这个文件。

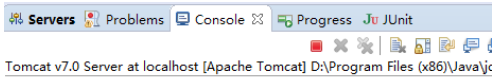
因为我在index.jsp的确引入过bootstrap.min.css，所以他也就一起加载进来了。

现在，我输入一句话，点击提交，看看会发生什么？



我们把input框里面的内容提交到服务器程序 MyServlet

控制台已经接受到了，这里比较幸运，没有遇到中文乱码的问题，那么先不管乱码了。



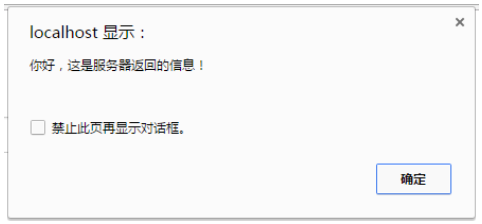
你好，我是剽悍一小兔！

因为MyServlet中没有返回什么东西，所以alert出来的是空。

好的，那我们给浏览器也返回一句话吧。

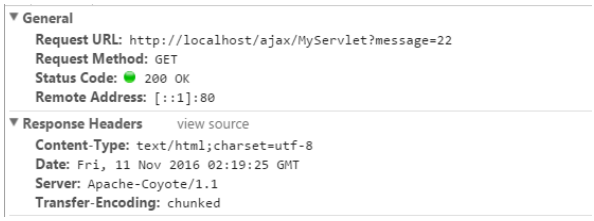
```
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    String msg = req.getParameter("message");
    System.out.println(msg);

    resp.setContentType("text/html;charset=utf-8");
    PrintWriter out=resp.getWriter();
    out.println("你好，这是服务器返回的信息！");
    out.flush();
    out.close();

}
```

再次点击提交按钮



OK了。

接下来，看一下请求的具体信息



在比对一下js代码，就一目了然了。

```
btn.onclick = function(){
    var  xhr = window.XMLHttpRequest?new XMLHttpRequest():new ActiveXObject("Microsoft.XMLHTTP");
    xhr.open("get","MyServlet?message="+document.getElementsByTagName("input")[0].value,true);
    xhr.send();
    xhr.onreadystatechange = function(){
        if(xhr.readyState==4 && xhr.status==200){
            alert(xhr.responseText);
        }
    };
}
```

readyState:

0: 请求未初始化
1: 服务器连接已建立，还没发送
2: 请求已接收
3: 请求处理中
4: 请求已完成，且响应已就绪
当 readyState 等于 4 且状态为 200 时，表示响应已就绪.

请求方式是get，并且只有当返回的状态码为200的时候，才会打印出responseText，这个就是对应的

out.println("你好，这是服务器返回的信息！");

这句话。

1.5 基于post方式的数据请求

get方法会在URL地址栏里显示你提交所带的值，post方法不会。所以，相对来说，post方法比较安全。

post方法在流程上和get方式差不多，我就直接上代码了：

```
window.onload = function(){
    var btn = document.getElementById("submit");
    btn.onclick = function(){
        var  xhr = window.XMLHttpRequest?new XMLHttpRequest():new ActiveXObject("Microsoft.XMLHTTP");
        xhr.open("post","MyServlet",true);
        xhr.setRequestHeader("Content-type","application/x-www-form-urlencoded");

        var postData = {message : document.getElementsByTagName("input")[0].value};

        var postDataStr = (function(obj){ // 转成post需要的字符串.
            var str = "";
            for(var prop in obj){
                str += prop + "=" + obj[prop] + "&"
            }
            return str;
        })(postData);
        alert(postDataStr);
        xhr.send(postDataStr);
        xhr.onreadystatechange = function(){
            if(xhr.readyState==4 && xhr.status==200){
                alert(xhr.responseText);
            }
```

```
        };
    }
};
```

MyServlet.java

```
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class MyServlet extends HttpServlet{

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        req.setCharacterEncoding("UTF-8");
        String msg = req.getParameter("message");
        System.out.println(msg);

        resp.setContentType("text/html;charset=utf-8");
        PrintWriter out=resp.getWriter();
        out.println("你好，这是服务器返回的信息！");
        out.flush();
        out.close();

    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        doGet(req, resp);
    }

}
```

个人博客地址：http://www.xiaotublog.com/index.html

首页：

详情页：

JavaScript： 零基础轻松学闭包（1）

发布时间：2016年11月09日 16点52分08秒
博客类别：零基础学习JavaScript
阅读(31) 评论(0) 喜欢(3)