前后端交互

什么是 AJAX ？

AJAX = 异步 JavaScript 和 XML。

AJAX 是一种用于创建快速动态网页的技术。

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。

传统的网页（不使用 AJAX）如果需要更新内容，必需重载整个网页面。

有很多使用 AJAX 的应用程序案例：新浪微博、Google 地图、开心网等等。、

关于 AJAX 的学习网站：http://www.runoob.com/ajax/ajax-tutorial.html

一、jquery ajax

1、采用$.post 方法

index.jsp 页面

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<%@ page contentType="text/html; charset=UTF-8"%>
<html>
<head>
<title></title>
<script src="js/jquery-1.12.2.js"></script>
<script language="JavaScript">
  function checkUserid() {
    $.post('Ajax/CheckServlet',//url
    {
      userid : $("#userid").val(),
      sex : "男"
    }, function(data) {
      var obj = eval('(' + data + ')');
      alert(obj.success);
    });
  }
</script>
</head>
<body>
  用户 ID:
  <input type="text" id="userid" name="userid"> <span id="msg"></span>
  <br> <button onclick="checkUserid()">传输</button>
</body>
</html>
```

CheckServlet.java 代码如下

```
package com.ajax;

```

```
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 CheckServlet extends HttpServlet {

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

  public void doPost(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
    /*设置字符集为'UTF-8'*/
    request.setCharacterEncoding("UTF-8");
    response.setCharacterEncoding("UTF-8");
    String userid = request.getParameter("userid"); // 接收 userid
    String sex = request.getParameter("sex");//接收性别
    System.out.println(userid);
    System.out.println(sex);

    //写返回的 JSON
    PrintWriter pw = response.getWriter();
    String json = "{'success':'成功','false':'失败'}";
    pw.print(json);
    pw.flush();
    pw.close();

  }
}
```

由于这里采用的是 servlet 的方式，所以要配置 web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaeehttp://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
  id="WebApp_ID" version="3.1">
  <display-name>Ajax</display-name>

  <servlet>
    <servlet-name>CheckServlet</servlet-name>
    <servlet-class>com.ajax.CheckServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>CheckServlet</servlet-name>
    <url-pattern>/Ajax/CheckServlet</url-pattern>
  </servlet-mapping>
</web-app>
```

在页面输入一个 ID，可以在后台接收到并且打印出来，后台通过 PrintWriter 进行回写 JSON 返回前端，前端通过 eval 将 JSON 变换为 Object 对象，通过 obj.name 获取 JSON 值

2、采用$.get 方法

只需要将 jsp 页面里面的 post 改为 get 即可

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<%@ page contentType="text/html; charset=UTF-8"%>
<html>
<head>
<title></title>
<script src="js/jquery-1.12.2.js"></script>
<script language="JavaScript">
  function checkUserid() {
    $.get(
      'Ajax/CheckServlet',//url
      {
        userid:$("#userid").val(),
        sex:"男"
      },
      function(data){
        var obj = eval('('+data+')');
        alert(obj.success);
      }
    );
  }
</script>
</head>
<body>

  用户 ID:
  <input type="text" id="userid" name="userid"> <span id="msg"></span>
    <br>
      <button onclick="checkUserid()">传输</button>
</body>
</html>
```

结果与$.post 一样

3、通过$.ajax 方法

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<%@ page contentType="text/html; charset=UTF-8"%>
<html>
<head>
<title></title>
<script src="js/jquery-1.12.2.js"></script>
<script language="JavaScript">
  function checkUserid() {
    $.ajax({
      type : 'post',
      data : {
        userid : $("#userid").val(),
        sex : "男"
      },
```

```
15.       url : "Ajax/CheckServlet",
16.       success : function(data) {  //data为后端post函数传递来的数据，花括号里面这里写结果操作代码
17.         var obj = eval('(' + data + ')');
18.       alert(obj.success);
19.       },
20.       error : function() {
21.         alert("数据传输失败!");
22.       },
23.       complete : function() {
24.       }
25.     });
26.   }
27. </script>
28. </head>
29. <body>
30.
31.   用户ID:
32.   <input type="text" id="userid" name="userid"> <span id="msg"></span>
33.     <br>
34.       <button onclick="checkUserid()">传输</button>
35. </body>
36. </html>
```

$.ajax 方法也是可以分为 post 和 get 方法的，通过修改 type 来修改发送的方式
结果与方法 1 是相同的

二、原生 js ajax

loginAjax.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<script type="text/javascript">
   var xmlHttp;
   function createXMLHttp(){
      if (window.XMLHttpRequest) {
         xmlHttp = new XMLHttpRequest();
      }else {
         xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
      }
```

```
    }
    function showMsg(){
        createXMLHttp();
        xmlHttp.open("POST","LoginCheckServlet?
username="+document.getElementById('input').value);  红色部分为后端提供给前端的接口
LoginCheckServlet 和前端传给后端的参数
        xmlHttp.onreadystatechange = showMsgCallback;
        xmlHttp.send(null);
    }
    function showMsgCallback(){
        if (xmlHttp.readyState == 4) {
            if (xmlHttp.status == 200) {
                var text = xmlHttp.responseText; /*前端获取从后台传来的json数据*/
                alert(text);
                document.getElementById("msg").innerHTML = text;
            }
        }
    }
</script>
username:<input id="input" type="text" name="username">

userPassword:<input type="text" name="password">

<input  type="button" name="msg" onclick="showMsg()" value="登录"><span
id="msg"></span>
</body>
</html>
```

LoginCheckServlet 部分代码：

```
protected void doGet(HttpServletRequest request,
HttpServletResponse  response) throws ServletException, IOException {
        request.setCharacterEncoding("utf-8");
```

```
    response.setContentType("text/html;charset=utf-8");
```

```
    String username=request.getParameter("username");
    String password=request.getParameter("password");
```

```
    PrintWriter  pw=response.getWriter();
    JSONObject js = new JSONObject();     使用 JSONObject 类需要导入 几个 jar
包 （见本机地址：F:\Java 学习）
```

```
    if(username.equals("admin")){
        js.put("loginSuccess", username);
        pw.print(js.toString());
    }else{
        js.put("loginFail","登录失败");
        pw.print(js.toString());
    }
  }
```

番外篇：前后端交互问题

前后端交互问题分为两种：

1、后端接受请求 处理业务逻辑并且负责页面的跳转和数据的显示

这种情况，后端为大，前端仅仅把参数传过来，后端就进行一系列的操作（参数解析，业务逻辑处理，查询数据库，包装前端所需数据，重定向或者返回带有数据的页面），可能就会造成后端代码臃肿，页面太多

2、前端展示数据，jquery+ajax 请求后端数据库资源（后端主要是接受传参，业务处理后返回数据），前端拿到数据后负责页面的局部更新和数据展示一前一后各有利弊

这种情况，前端为大，前端一方面负责展示数据（html 页面），另一方面前端还负责 DOM 操作发送参数给后端，接受返回数据并展示（jquery+ajax），后端仅仅做的就是接受参数，业务逻辑处理，查询包装数据，返回给前端就可以了（一般返回 json）