Ajax

主讲:石小俊

1.介绍Ajax

Asynchronous JavaScript And Xml

异步的js与xml

1-1 什么是异步？

为了改善传统web应用中

请求-->等待-->响应的这种模式的弊端

使用js调用浏览器所提供的一个对象XMLHttpRequest(Ajax引擎)

可以向服务器异步的发送请求

在异步请求发送的同时，不会影响到当前正在执行的请求

不会丢失页面的任何数据

可以根据异步请求所返回的响应信息

通过js局部的改变页面的信息

1-2 Ajax引擎获取方式

- 通用方式
  - var xhr = new XMLHttpRequest();
- IE8之前版本

```
if(window.ActiveXObject){
    var xhr = new ActiveXObject("Microsoft.XMLHTTP");
}
```

1-3 传统web与Ajax请求的区别

区别	传统web	Ajax
发送请求的方式	表单提交时发送	Ajax引擎异步发送
服务器响应	整个页面	只是一段数据
客户端处理方式	只有响应完成之后才能进行其他操作	不需要等待响应，也能执行其他操作

2.XMLHttpRequest对象介绍

2-1 常用属性

- onreadystatechange:指定回调函数
  - 当异步请求返回响应之后，所执行的操作
- readyState:XMLHttpRequest的状态信息,值有5个
  - 0：对象已经创建，但是尚未初始化
  - 1：请求已经提出，但是尚未发送
  - 2：请求已经发送，正在发送中
  - 3：请求正在处理，尚未完成相应
  - 4：请求已经完成，可以访问服务器响应回来的数据
- status：Http的状态
  - 200：成功响应
  - 302：重定向
  - 403：权限不足
  - 404：资源不存在
  - 405：请求方式不对
  - 500：服务器内部异常
  - 状态码的值小于400，都是成功，只是成功之后处理方式不同
  - 状态码的值大于等于400，都是失败，只是失败的原因不同
- responseText
  - 获取响应回来的文本内容
- responseXML
  - 获取响应回来的xml文档信息

2-2 常用方法

- open(method,url,async):建立与服务器的连接
  - method:请求提交方式，默认为get
  - url:请求地址
  - async:是否异步，默认为true
- send(content)：发送请求
  - content:请求中的参数

    - 如果请求中没有参数，则为null
- setRequestHeader(header,value):设置请求头信息

3.Ajax返回普通字符串

```
<%--
  Created by IntelliJ IDEA.
  User: User
  Date: 2018/10/24
  Time: 11:27
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>用户注册</title>
    <script type="text/javascript">
        function checkUsername(){
            var xhr = new XMLHttpRequest();
            var username = document.getElementById("username").value;
            xhr.onreadystatechange=function(){
                if(xhr.status == 200 && xhr.readyState == 4){
                    var result = xhr.responseText;
                    document.getElementById("s1").innerHTML=result;
                }
            };
            <%--xhr.open("get","${pageContext.request.contextPath}/checkUsername?
username="+username,true);--%>
            <%--xhr.send(null);--%>

            //post请求
            xhr.open("post","${pageContext.request.contextPath}/checkUsername",true);
            //设置请求头信息
            xhr.setRequestHeader("Content-type","application/x-www-form-urlencoded");
            xhr.send("username="+username);

        }
    </script>

</head>
<body>
    <h1>用户注册</h1>
    <hr/>

    <form action="" method="post">
        用户名:<input type="text" id="username" onblur="checkUsername()"/><span id="s1"></span>
<br/>
        密   码:<input type="password"/><br/>
        电   话:<input type="text" id="phone"/><br/>
```

```
        地   址:<input type="text" id="address"/><br/>
        <input type="submit" value="注册">
    </form>

</body>
</html>
```

4.json

4-1 是什么

JavaScript Object Notation

是一种轻量级的数据交换格式

采用的是与程序语言无关的文本格式

易于编写与阅读

易于解析

4-2 语法

相当于一组键值对的集合，类似于map

{属性名:属性值,属性名:属性值....}

4-3 为什么用

在json之前所使用的数据交换格式为:xml

- xml缺点
  - 解析过于麻烦
  - 可读性较差
  - 占用资源较多
- json优点
  - 易于解析
  - 可读性较高
  - 占用资源较少

4-4 怎么用

- 基础语法

- {属性名:属性值,属性名:属性值....}
- 所涉及的所有的属性值都是一个普通数据

```
var user = {"username":"admin","password":"123456","phone":"13812345678","address":"江苏-南京","2test":"testaaa"};
console.log(user.username);
console.log(user["password"]);
console.log(user['phone']);
console.log(user['address']);
//通过.获取属性值的时候，属性名不能以数字开头
//假如当前的属性偏偏只能是数字
//则不要使用.的方式来实现
// console.log(user.2test);
//使用[]来获取
console.log(user['2test']);
```

- 属性中包含json对象

```
var name = {"firstName":"zhang","lastName":"san"};
// var user = {"id":1,"name":name};
var user = {"id":2,"name":{"firstName":"li","lastName":"si"}};
console.log(user.name.firstName);
console.log(user.name['firstName']);
console.log(user["name"].lastName);
console.log(user["name"]["lastName"]);
```

- json对象集合

```
var users = [{
    "id":1,
    "username":"admin"
},{
    "id":2,
    "username":"zhangsan"
},{
    "id":3,
    "username":"lisi"
}];
console.log(users[1].username);
```

5.AJax返回json对象

```
package servlet;

import entity.Product;
import net.sf.json.JSONObject;
```

```
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

/**
 * Author:shixiaojun@itany.com
 * Date:2018/10/24 16:10
 * Description:
 * version:1.0
 */
@WebServlet("*.product")
public class ProductServlet extends HttpServlet {

    @Override
    protected void service(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType("text/html;charset=utf-8");
        String path = request.getServletPath();
        if("/findAll.product".equals(path)){
            findAll(request,response);
        }
        if("/findById.product".equals(path)){
            findById(request,response);
        }
    }

    protected void findById(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        PrintWriter out = response.getWriter();
        String id = request.getParameter("id");
        Connection conn = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        String url = "jdbc:mysql://127.0.0.1:3306/shop?useUnicode=true&characterEncoding=utf8";
        Product product = null;
        try {
            Class.forName("com.mysql.jdbc.Driver");
            conn = DriverManager.getConnection(url,"root","");
            String sql = new StringBuffer()
                    .append(" select id,name,price ")
                    .append(" from t_product")
                    .append(" where id = ? ")
                    .toString();
```

```
            ps = conn.prepareStatement(sql);
            ps.setInt(1,Integer.parseInt(id));
            rs = ps.executeQuery();
            while(rs.next()){
                product = new Product();
                product.setId(rs.getInt("id"));
                product.setName(rs.getString("name"));
                product.setPrice(rs.getDouble("price"));
            }
//            out.print("
{'id':"+product.getId()+",'name':'"+product.getName()+"','price':"+product.getPrice()+"}");
            //fromObject(Object)将一个java对象转换成json格式的对象
            //toString():将一个json格式的对象转换成json格式的字符串
            String jsonProduct = JSONObject.fromObject(product).toString();
            out.print(jsonProduct);
        } catch (Exception e){
            e.printStackTrace();
        }
    }
    protected void findAll(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        Connection conn = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        String url = "jdbc:mysql://127.0.0.1:3306/shop?useUnicode=true&characterEncoding=utf8";
        List<Product> products = new ArrayList<Product>();
        try {
            Class.forName("com.mysql.jdbc.Driver");
            conn = DriverManager.getConnection(url,"root","");
            String sql = new StringBuffer()
                    .append(" select id,name,price ")
                    .append(" from t_product")
                    .toString();
            ps = conn.prepareStatement(sql);
            rs = ps.executeQuery();
            while(rs.next()){
                Product product = new Product();
                product.setId(rs.getInt("id"));
                product.setName(rs.getString("name"));
                product.setPrice(rs.getDouble("price"));
                products.add(product);
            }
            request.setAttribute("products",products);
            request.getRequestDispatcher("/list.jsp").forward(request,response);
        } catch (Exception e){
            e.printStackTrace();
        }
    }
}
```

```
<%--
```

```
  Created by IntelliJ IDEA.
  User: User
  Date: 2018/10/24
  Time: 14:02
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>Ajax返回json对象</title>
    <script type="text/javascript">
        function show(id,e){
            document.getElementById("d").style.display="block";

            var xhr = new XMLHttpRequest();
            xhr.onreadystatechange=function(){
                if(xhr.status == 200 && xhr.readyState == 4){
                    var result = xhr.responseText;
                    // console.log(result);
                    var product = eval("("+result+")");
                    document.getElementById("d").style.left=e.clientX;
                    document.getElementById("d").style.top=e.clientY;
                    document.getElementById("s1").innerHTML=product.id;
                    document.getElementById("s2").innerHTML=product.name;
                    document.getElementById("s3").innerHTML=product.price;
                }
            };
            xhr.open("get","${pageContext.request.contextPath}/findById.product?id="+id,true);
            xhr.send(null);
        }
        function hide(){
            document.getElementById("d").style.display="none";
        }
    </script>
</head>
<body>
    <!-- 练习要求
        列表中的所有数据全部取自于数据库中
        当鼠标光标悬停在某一个商品上的时候
        在div中显示该商品的详细信息
        并将div在鼠标光标所在位置显示
        当鼠标光标不在任何商品上的时候
        div不显示
    -->
    <ul>
        <c:forEach items="${products}" var="product">
            <li onmouseover="show(${product.id},event)" onmouseout="hide()">${product.name}</li>
        </c:forEach>
    </ul>

    <hr/>
```

```
    <div id="d" style="background-color: #abc7c4;width: 20%;padding: 10px;display:
none;position: absolute;">
        编号:<span id="s1"></span><br/>
        商品名称:<span id="s2"></span><br/>
        商品单价:<span id="s3"></span><br/>
    </div>

</body>
</html>
```

6.Ajax返回json对象集合

```
package servlet;

import entity.Data;
import entity.Product;
import net.sf.json.JSONArray;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

/**
 * Author:shixiaojun@itany.com
 * Date:2018/10/25 10:55
 * Description:
 * version:1.0
 */
@WebServlet("/query")
public class QueryServlet extends HttpServlet {

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

        PrintWriter out = resp.getWriter();

        String key = req.getParameter("key");
```

```
        Connection conn = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        String url = "jdbc:mysql://127.0.0.1:3306/shop?useUnicode=true&characterEncoding=utf8";
        List<Data> datas = new ArrayList<Data>();
        try {
            Class.forName("com.mysql.jdbc.Driver");
            conn = DriverManager.getConnection(url,"root","");
            String sql = new StringBuffer()
                    .append(" select id,spell,message ")
                    .append(" from t_data ")
                    .append(" where spell like ? ")
                    .append(" or message like ? ")
                    .toString();
            ps = conn.prepareStatement(sql);
            ps.setString(1,"%"+key+"%");
            ps.setString(2,"%"+key+"%");
            rs = ps.executeQuery();
            while(rs.next()){
                Data data = new Data();
                data.setId(rs.getInt("id"));
                data.setSpell(rs.getString("spell"));
                data.setMessage(rs.getString("message"));
                datas.add(data);
            }

            String datasJson = JSONArray.fromObject(datas).toString();
            out.print(datasJson);

        } catch (Exception e){
            e.printStackTrace();
        }
    }
}
```

```
<%--
  Created by IntelliJ IDEA.
  User: User
  Date: 2018/10/25
  Time: 9:08
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>关键字搜索</title>
    <style type="text/css">
        li{
            list-style: none;
        }
        div{
```

```
            border:1px solid gray;
            width: 300px;
            padding: 5px;
            border-top: 0px;
            max-height: 100px;
            overflow: auto;
            display: none;
        }
    </style>
    <script type="text/javascript">
        function query(){
            var xhr = new XMLHttpRequest();
            var key = document.getElementById("key").value;
            var d = document.getElementById("d");
            if(key == ""){
                d.style.display="none";
                return;
            }
            d.innerHTML="";
            xhr.onreadystatechange=function(){
                if(xhr.status == 200 && xhr.readyState == 4){
                    var datas = xhr.responseText;
                    console.log(datas);
                    datas = eval("("+datas+")");
                    d.style.display="block";
                    if(datas.length == 0){
                        d.style.display="none";
                        return;
                    }
                    for(var i = 0; i < datas.length;i++){
                        d.innerHTML = d.innerHTML+"<li>"+datas[i].message+"</li>";
                    }
                }
            };
            xhr.open("get","${pageContext.request.contextPath}/query?key="+key,true);
            xhr.send(null);
        }
    </script>
</head>
<body>
    <input type="text" id="key" style="width: 312px;" onkeyup="query()">
    <div id="d">
    </div>
</body>
</html>
```

7.jquery-Ajax

7-1 get

处理get请求

- url:请求地址
- data:可选参数，请求参数
- callback:可选参数，指定回调函数
- type:可选参数，指定响应数据的格式类型
    - text:默认值，表示一个普通字符串
    - json:表示返回的是json对象

7-2 post

处理post请求

- url:请求地址
- data:可选参数，请求参数
- callback:可选参数，指定回调函数
- type:可选参数，指定响应数据的格式类型
    - text:默认值，表示一个普通字符串
    - json:表示返回的是json对象

7-3 getJSON

响应数据为json

- url:请求地址
- data:可选参数，请求参数
- callback:可选参数，指定回调函数

7-4 Ajax

只有一个参数options,该参数表示一组选项的集合

常用选项如下:

- type:请求方式，默认为get
- url:请求地址
- data:请求参数
- dataType:响应数据的格式，默认为text
    - text
    - json
    - jsonp
- jsonp

    - 在使用jsonp实现跨域时可以使用
    - 一般情况不用
    - 其值表示的是请求中?key=某个回调函数的名字时其中key所对应的数据
    - 默认值为callback
- success:当请求成功响应之后调用的回调函数
- error:当响应出错之后调用的回调函数

8.Ajax跨域之jsonp

8-1 同源策略

所谓的同源策略指的是协议名、域名、端口号相同

同源策略与工程无关

同一个服务器中的不同工程他们也是符合同源策略

什么是跨域

不符合同源策略则产生跨域

当协议名、域名、端口号三者有任意一个不同，则表示跨域

当协议名不一样时，不仅仅是跨域，已经属于跨协议了

8-2 url

Uniform Resource Locator

统一资源定位符

完整url地址:

协议名://域名:端口/工程名/访问命令[可选参数]

- 协议名
    - 通信协议
    - 常见协议:http、ftp、https、file
- 域名
    - 主机名
    - 开发中一般使用的是ip
- 端口
    - 8080
    - 3306
    - 1521
- 工程名
    - 部署到服务器中的工程

- 访问命令
  - 访问该工程中某个资源所使用的命令
  - 访问Servlet时使用的url-pattern
  - 访问jsp时所使用的路径+文件名
- 可选条件
  - ?id=...
  - ;jsessionid=....
  - #id

8-3 jsonp

Json with padding

json的一种使用方式

在html中，很多的标签都自带跨域功能

本质上，jsonp就是通过script表示实现的