[第10章 Ajax编程技术 1](#_Toc509475064)

[10.1 Ajax技术简介 1](#_Toc509475065)

[10.2 XMLHttpRequest对象 2](#_Toc509475066)

[10.3 Ajax应用案例 4](#_Toc509475067)

[10.3.1 案例——异步表单验证 5](#_Toc509475068)

[10.3.2 案例——实现级联列表 8](#_Toc509475069)

[10.4 使用JSON进行数据传输 10](#_Toc509475070)

[10.4.1 JSON简介 10](#_Toc509475071)

[10.4.2 在JavaScript中使用JSON 12](#_Toc509475072)

[10.4.3 Java对象与JSON的转化 12](#_Toc509475073)

[10.4.5 案例——基于Ajax+JSON的表格数据浏览 15](#_Toc509475074)

[10.5 jQuery框架中的Ajax方法 17](#_Toc509475075)

[10.5.1 jQuery框架中常用Ajax方法 17](#_Toc509475076)

[10.5.2案例——基于Ajax+jQuery的表格记录添加 21](#_Toc509475077)

[本章小结 22](#_Toc509475078)

[习题 23](#_Toc509475079)

# 第10章 Ajax编程技术

Ajax（Asynchronous JavaScript and XML）是运用JavaScript和可扩展语言（XML）实现浏览器与服务器通讯的一种技术。本章介绍实现Ajax编程的两种方式：原始方式及利用jQuery框架提供的方法。

## 10.1 Ajax技术简介

Ajax是实现浏览器与服务器异步交互的技术，对用户请求的响应不需要刷新整个页面，只需要刷新局部页面即可。

Ajax技术是一系列技术的集合，主要涉及的技术有：

* 使用XHTML(HTML)和CSS构建标准化的展示层。
* 使用DOM进行动态显示和交互。
* 使用XML和XSLT进行数据交换和操纵。
* 使用XMLHttpRequest异步获取数据。
* 使用JavaScript将所有元素绑定在一起。

对于这些技术，大部分在前面章节中已介绍，下面主要介绍XMLHttpRequest对象及其使用。

## 10.2 XMLHttpRequest对象

XMLHttpRequest对象是Ajax的核心技术之一，在Ajax中，通过这个对象实现与服务器端通信。这个对象由JavaScript来创建，在不同的浏览器中有不同的创建方法，但创建成功后，其使用方法是相同的。

下面重点介绍该对象的创建及常用属性和方法。

1．XMLHttpRequest对象的创建

不同的浏览器创建XMLHttpRequest对象使用的语句是不同的。为了在不同的浏览器下都能成功创建XMLHttpRequest对象，需要针对不同的浏览器创建。

XMLHttpRequest对象是通过JavaScript创建的，代码如下：

var xmlHttpRequest=null; //声明XMLHttpRequest对象

if(window.XMLHttpRequest){ //针对Mozilla,Safari,Opera,IE7等浏览器创建

xmlHttpRequest = new XMLHttpRequest();

}

else if(window.ActiveXObject){

try{

xmlHttpRequest = new ActiveXObject("Msxml2.XMLHTTP"); //针对IE较新版本创建

}catch(e){

try {

xmlHttpRequest = new ActiveXObject("Microsoft.XMLHTTP");//IE较老版本

}catch(e){}

}

}

2．XMLHttpRequest的方法和属性

在XMLHttpRequest对象创建后，就可以对该对象进行各种不同的操作，从而完成和服务器的通信。XMLHttpRequest对象的常用方法和属性有：

（1）open(string request-type,string url,boolean asynch,string name,string password)方法

该方法用于建立到服务器的连接，其参数的含义：

* request-type：发送请求的类型。该参数的取值为：get或post或head方法。要特别注意参数汉字乱码的问题（处理方法在前面章节中已介绍）。
* url：要连接的服务器的URL。
* asynch：若使用异步连接则为true，否则为false。该参数是可选的，默认为true。
* username：若需要身份验证，则在此指定用户名。
* password：若需要身份验证，在此指定口令。

通常使用其中的前三个参数。

（2）send(String content)方法

该方法向服务器发送请求。参数content，表示发送的内容。

（3）setRequestHeader(string label,string value)

该方法在发送请求前，先设置请求头。

例如，若在open方法中使用的request-type的值是“post”，则需要设置请求头：

xmlHttpRequest.setRequestHeader("Content-type","application/x-www-form-urlencoded");

（4）readyState属性

提供当前HTML的就绪状态，用于确定该请求是否已经开始、是否得到了响应或者请求/响应模型是否已经完成。它还可以帮助确定读取服务器提供的响应文本或数据是否安全。在Ajax应用程序中五种就绪状态：

* 0：请求没有发出（在调用open()之前）
* 1：请求已经建立但还没有发出（调用send()之前）
* 2：请求已经发出正在处理之中（这里通常可以从响应得到内容头部）
* 3：请求已经处理，响应中有部分数据可用，但是服务器还没有完成响应
* 4：响应已完成，可以访问服务器响应并使用它。

对于Ajax编程，需要直接处理的唯一状态就是就绪状态4，它表示服务器响应已经完成，可以安全地使用响应数据了。

（5）status属性

服务器响应的状态代码。服务器响应完成后(readyState=4)，从完成的响应信息中可获得状态代码。

例如：输入了错误的URL请求将得到404错误码，它表示该页面不存在；403和401错误码表示所访问的数据受到保护或者禁止访问；200状态码表示一切顺利。

因此，如果就绪状态是4而且状态码是200，就可以处理服务器的数据了，而且这些数据应该就是要求的数据（而不是错误或者其他有问题的信息）。

（6）onreadystatechange属性

用于指定XMLHttpRequest对象的状态改变函数（类似于按钮对象的onclick属性），当XMLHttpRequest对象状态(readyState的值)改变时，该函数将被触发，该函数也称回调函数。

假设回调函数为callback，则它的代码通常为：

function callback(){

if (xmlHttpRequest.readyState == 4){

if (xmlHttpRequest.status == 200){

事件响应代码

}

}

}

（7）responseText属性和responseXML属性

XMLHttpRequest对成功返回的信息有两种处理方式：

responseText：服务器返回的请求响应文本；将传回的信息当字符串使用。

responseXML：服务器端返回的XML类型的响应；将传回的信息当XML文档使用，可以用DOM处理。这种情形下，通过如下代码设置：

response.setContentType("text/xml;charset=UTF-8");

3．XMLHttpRequest对象的运行周期

整个Ajax技术紧紧围绕在XMLHttpRequest对象周围，XMLHttpRequest对象的运行周期就是Ajax应用的运行。

（1）Ajax应用总是从创建XMLHttpRequest对象开始，XMLHttpRequest对象允许通过客户端脚本来发送HTTP请求，这种请求可以是GET方式的，也可以是POST方式的。

（2）XMLHttpRequest发送完后，服务器的响应何时到达？应该何时处理服务器的响应呢？这需要借助于JavaScript的事件机制。XMLHttpRequest对象也是一个普通的JavaScript对象，就如一个普通按钮或一个普通文本框一样，可以触发事件；而XMLHttpRequest触发的事件就是onreadystatechange，当XMLHttpRequest对象的状态改变时，将触发它。为XMLHttpRequest对象的onreadystatechange事件指定事件处理函数，该函数将在XMLHttpRequest状态改变时被执行，这个事件处理函数也叫回调函数。

（3）XMLHttpRequest状态改变，且readyState=4&&status=200时，表明服务器响应已经完成且是正确的响应状态，此时可以开始处理服务器响应。

（4）进入事件处理函数后，XMLHttpRequest依然不可或缺，事件处理函数借助于XMLHttpRequest的responseText或responseXML属性获取服务器的响应，至此XMLHttpRequest运行周期结束。

（5）JavaScript通过DOM操作将响应动态加载到XHMTL页面中。

## 10.3 Ajax应用案例

从上一节我们可以归纳Ajax应用编程步骤如下：

（1）创建XMLHttpRequest对象。

（2）从Web表单中获取需要发送的数据。

（3）设置要连接的URL。

（4）建立到服务器的连接。

（5）设置服务器在完成后要运行的回调函数。

（6）发送请求。

（7）在回调函数中，获取服务器的响应数据(text或xml)并显示。

可以将创建XMLHttpRequest对象和发送请求封装为两个js函数放在ajax.js中在页面中导入使用。代码如下：

var httpRequest=null;

function createXHR(){

if(window.XMLHttpRequest){ //Mozilla,Safari,Opera,IE7等

httpRequest = new XMLHttpRequest();

}else if(window.ActiveXObject){

try{httpRequest = new ActiveXObject("Msxml2.XMLHTTP"); //IE较新版本

}catch(e){

try {httpRequest = new ActiveXObject("Microsoft.XMLHTTP");//IE较老版本

}catch(e){httpRequest = null;}

}

}

if(!httpRequest){

alert("fail to create httpRequest");

}

}

function sendRequest(url,params,method,handler){

createXHR();

if(!httpRequest) return false;

httpRequest.onreadystatechange = handler;

if(method == "GET"){

httpRequest.open(method,url+ '?' + params,true);

httpRequest.send(null);

}

if(method == "POST"){

httpRequest.open(method,url,true);

httpRequest.setRequestHeader("Content-type","application/x-www-form-urlencoded");

httpRequest.send(params);

}

}

### 10.3.1 案例——异步表单验证

在传统的Web应用中，用户的身份验证是通过向服务器提交表单，服务器对表单中用户信息进行验证，然后再返回验证的结果，在这样的处理方式中，用户端必须等待到服务器返回处理结果才能进行别的操作，而且在这个过程中，会刷新整个页面。

在Ajax的处理方式中，可以把用户的信息通过XMLHttpRequest对象异步发送给服务器，在服务器端完成对用户身份验证后，把处理结果通过XMLHttpRequest对象返回用户，从而以异步的方式，在不刷新整个页面的情况下完成对用户身份的验证。

【例10-1】利用Ajax技术实现登录数据的检验，假设正确的用户名和密码是：张三和123，要求在输入完用户名后立即判断用户名是否存在。

【设计关键】表单数据验证的问题用其它技术已经实现过，用Ajax实现的不同点是：发送请求的方式、响应及获取响应的方式。

（1）设计页面Form.html，该页面是提交表单页面，并在该页面中利用JavaScript设计有关的函数：

function formCheck()：发送请求，并调用showResult()将获取的响应显示。

function showResult()：获取响应并显示。

（2）设计检验表单数据的Servlet：FormCheck.java（在web.xml中配置的映射地址是formcheck）。

【实现】

（1）表单页面Form.html，其代码如下：

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

<head>

<title>表单验证</title>

<meta http-equiv="content-type" content="text/html; charset=UTF-8">

<script type="text/javascript" src="js/ajax.js"></script>

<script type="text/javascript">

function formcheck(){

var url="formcheck";

var userid=document.getElementById("userid").value;

var userpwd=document.getElementById("userpwd").value;

var params="userid="+userid+"&userpwd="+userpwd;

sendRequest(url,params,'POST',showresult);

}

function showresult(){

if (httpRequest.readyState == 4) {

if (httpRequest.status == 200) {

var info=httpRequest.responseText;

document.getElementById("result").innerHTML=info;;

}

}

}

</script>

</head>

<body>

请输入用户名：<input type="text" name="userid" id=”userid” onblur="formcheck()"/><br/>

请输入密码　：<input type="password" name="userpwd" id=”userpwd”/><br/>

<input type="button" value="登录" onclick="formcheck()"/>

<div id="result"></div>

</body>

</html>

（2）检验表单数据的Servlet：FormCheck.java，其关键代码：

package servlets;

//省略了导入包

public class FormCheck extends HttpServlet {

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

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

PrintWriter out = response.getWriter();

request.setCharacterEncoding("UTF-8");

String userid=request.getParameter("userid");

if(!"张三".equals(userid)){

out.print("用户名不存在");

}

else{

String userpwd=request.getParameter("userpwd");

if(!"".equals(userpwd)){

if("123".equals(userpwd))

out.print("欢迎您");

else

out.print("密码错误");

}

}

}

}

（3）Servlet在web.xml中的配置信息：

<servlet>

<servlet-name>formcheck</servlet-name>

<servlet-class>servlets.FormCheck </servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>formcheck</servlet-name>

<url-pattern>/formcheck</url-pattern>

</servlet-mapping>

运行结果如图8-9所示。

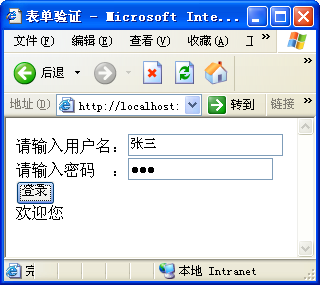
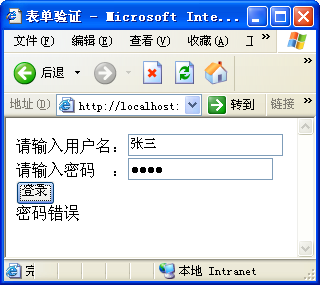
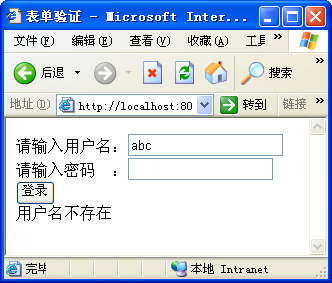


图10-1 例10-1运行结果

### 10.3.2 案例——实现级联列表

在We应用的开发中，经常会遇到联动动态列表的需求，有其是在查询条件的选择中，所有的下拉列表中的选项都是从数据库中动态取出的，当选择第一个下拉列表的时候，后面的下拉列表要以这个选择为条件从数据库中取出满足条件的内容，从而调整显示选项的内容。下面给出采用Ajax实现级联列表的方法。

【例10-2】实现级联列表。当在第一个列表框中输入省名时，在第二个列表框中显示该省包含的城市。

【设计关键】采用Ajax实现级联列表是最合理的技术，第一个列表项变化时发送请求，让服务器查找并返回第二个列表框要显示的值，在客户端将信息更新到第二个列表框中。为此，需要在网页select.html中设计函数：function refresh()和function show()。

【实现】

（1）级联列表页面select.html

<html>

<head>

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

<script type="text/javascript" src="js/ajax.js"></script>

<script type="text/javascript">

function refresh(){

var p=document.getElementById("prov").value;

var city=document.getElementById("city");

if(p==""){

city.options.length=0;

city.options.add(new Option("--请选择城市--"))

}

else{

var url="list";

var params="proc="+p ;

sendRequest(url,params,'POST',show);

}

}

function show(){

var city=document.getElementById("city");

if (httpRequest.readyState == 4) {

if (httpRequest.status == 200) {

var citylist=httpRequest.responseText.split(",");

var citynum=citylist.length;

city.options.length=0;

for(i=0;i<citynum;i++)

city.options.add(new Option(citylist[i]))

}

}

}

</script>

</head>

<body>

<select name="prov" id="prov" onchange="refresh();">

<option value="">--请选择省份--</option>

<option value="山东">山东</option>

<option value="江苏">江苏</option>

<option value="广东">广东</option>

</select>

<select name="city" id="city">

<option>--请选择城市--</option>

</select>

</body>

</html>

（2）查找指定省份的城市列表的Servlet：List.java（只列出主要代码）

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

Map<String,String> pm=new HashMap<String,String>();

pm.put("山东", "济南,青岛,泰安,潍坊,烟台,聊城,枣庄,菏泽,莱芜,临沂");

pm.put("江苏", "南京,苏州,无锡,徐州,南通,连云港,镇江,常州,淮安,扬州");

pm.put("广东", "广州,深圳,珠海,汕头,佛山,东莞,湛江,江门,中山,惠州");

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

request.setCharacterEncoding("UTF-8");

PrintWriter out = response.getWriter();

String s1=request.getParameter("proc");

out.print(pm.get(s1));

}

运行结果如图10-2所示。

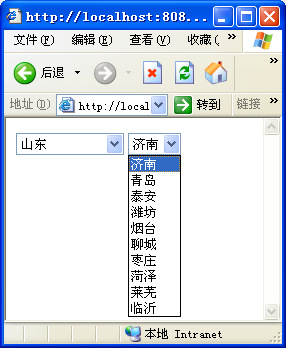


图10-2 例10-2运行结果

## 10.4 使用JSON进行数据传输

当客户端和服务器传输的信息（参数信息和响应信息）比较简单时，可以采取一般的文本格式，10.3节的两个实例都是这样，客户端向服务器传递的参数是键值对字符串，服务器返回的也是普通的字符串（例10-2稍微复杂点，是带定界符“，”的字符串）。当传输的数据量很大且结构复杂时，可以考虑使用结构化的字符串进行数据传输，如XML或JSON等，在Ajax领域后者较为流行。

### 10.4.1 JSON简介

JSON (JavaScript Object Notation) 是一种轻量级的数据交换格式，易于人阅读和编写，同时也易于机器解析和生成，而且它是基于JavaScript 的。 JSON采用完全独立于语言的文本格式，但是也使用了类似于C语言家族的习惯（包括C, C++, C#, Java, JavaScript 等）。这些特性使JSON成为理想的数据交换语言。

1. JSON基本语法

Json包括两种基本结构：

（1）对象：对象表示为“{}”扩起来的内容，格式为 {key1:value1,key2:value2,...}的键值对集合。通过对象.key 获取属性值，属性值类型可以是 数字、字符串、数组、对象等。

（2）数组：数组表示为 “[]”扩起来的内容，格式为 ["java","javascript","vb",...]，通过数组名.[index] 获取数组元素，元素类型可以是数字、字符串、数组、对象等。

经过对象、数组两种结构的嵌套可以组成更复杂的数据结构。JSON数据举例如下：

* 简单的名称/值对：{ "firstName": "Brett" }
* 多个名称/值对：

{ "firstName": "Brett", "lastName":"McLaughlin", "email": "brett@newInstance.com" }

* 对象数组：

[

{ "firstName": "Brett", "lastName":"McLaughlin", "email": "brett@newInstance.com" },

{ "firstName": "Jason", "lastName":"Hunter", "email": "jason@servlets.com" },

{ "firstName": "Elliotte", "lastName":"Harold", "email": "elharo@macfaq.com" }

]

* 对象属性值是数组

{ “employees": [

{ "firstName": "Brett", "lastName":"McLaughlin", "email": "brett@newInstance.com" },

{ "firstName": "Jason", "lastName":"Hunter", "email": "jason@servlets.com" },

{ "firstName": "Elliotte", "lastName":"Harold", "email": "elharo@macfaq.com" }

] }

2.JSON与XML的比较

XML同样可以表示结构化的数据，例如上面最后一个JSON串对应的XML表示形式为：

<employees>

< employee >

<firstName>Brett</firstName>

<lastName>McLaughlin</lastName>

<email>brett@newInstance.com</email>

</ employee >

< employee >

<firstName>Jason</firstName>

<lastName>Hunter</lastName>

<email>jason@servlets.com</email>

</ employee >

< employee >

<firstName>Elliotte</firstName>

<lastName>Harold</lastName>

<email>elharo@macfaq.com</email>

</employee>

</ employees >

可以从以下几方面比较两者：

* 可读性：JSON和XML的可读性可谓不相上下，XML略占上风。
* 可扩展性：XML天生有很好的扩展性，JSON当然也有，没有什么是XML能扩展，JSON不能的。
* 编码难度：XML有丰富的编码工具，比如Dom4j、JDom等，JSON也有json.org提供的工具，但是JSON的编码明显比XML容易许多，即使不借助工具也能写出JSON的代码，可是要写好XML就不太容易了。
* 解码难度：XML的解析得考虑子节点父节点关系，让人头昏眼花，而JSON的解析难度几乎为零。
* 流行度：XML已经被业界广泛的使用，而JSON才刚刚开始，但在Ajax领域，JSON凭借自身的优势有可能最终取代XML。

### 10.4.2 在JavaScript中使用JSON

1.创建和访问JSON对象

JSON是JavaScript原生格式，这意味着在JavaScript中处理JSON数据不需要任何特殊的 API 或工具包。例如：

（1）创建一个JSON对象

var company =

{“employees": [

{ "firstName": "Brett", "lastName":"McLaughlin", "email": "brett@newInstance.com" },

{ "firstName": "Jason", "lastName":"Hunter", "email": "jason@servlets.com" },

{ "firstName": "Elliotte", "lastName":"Harold", "email": "elharo@macfaq.com" }

] }；

（2）获取第一个雇员的firstName信息：

company.employees[0].fristName

（3）修改第一个雇员的firstName信息数据：

company.employees[0].fristName=“Vincent”

2.将JSON字符串转换为JSON对象

当服务器端返回JSON字符串时，客户端需要将JSON字符串转换为对象再访问，常用解析方法有两种：

（1）使用JSON.parse（JsonStr）方法（推荐）

（2）使用eval（“（”+JsonStr+“）”）方法

例如：

<script>

var jsonStr='{"name":"runoob1", "alexa":10000, "site":"www.runoob.com" }';

//方法一：

var obj = eval("("+jsonStr+")");

//方法二

var obj = JSON.parse(jsonStr);

</script>

两种方法推荐使用第一种，因为eval方法不会去检查给的字符串是否符合json的格式，同时如果给的字符串中存在js代码，eval也会一并执行，存在安全风险。

### 10.4.3 Java对象与JSON的转化

服务器端响应的结果通常是一个Java对象，在Ajax编程模式下需要将它转化成字符串发送给客户端，手动转化是很繁琐的，可以利用开源类库完成两者的转化，包括JackSon、FastJson、Google-Gson、JSon-lib类库等，这里简单介绍JSon-lib类库的使用，它的下载地址为：https://sourceforge.net/projects/json-lib/，还需要commons-beanutils-1.8.0.jar、commons-collections-3.2.1.jar、commons-lang-2.5.jar、commons-logging-1.1.1.jar、ezmorph-1.0.6.jar五个包的支持，可以自行从网上下载。主要用到JSONObject（JSON对象）和JSONArray（JSON数组）两个类。

1. Java对象、集合/数组转换为JSON字符串

主要方法：

JSONObject.fromObject();//对象=>字符串

JSONArray.fromObject(); //集合/数组=>字符串

2．JSON字符串转换为Java对象、集合

主要方法：

JSONObject.toBean();//json对象=>java对象

JSONArray.toCollection();//json集合=》java集合

【例10-3】Java对象与JSON转化示例。

测试类：

public class JsonlibTest {

public static void main(String[] args){

Student student1=new Student("20181234","张三","男");

System.out.print("将student1对象转换成json字符串:");

JSONObject jso1=JSONObject.fromObject(student1);

System.out.println(jso1.toString());

System.out.print("将studentList1对象转换成json字符串:");

Student student2=new Student("20185678","李四","女");

List<Student> studentList1=new ArrayList<Student>();

studentList1.add(student1);studentList1.add(student2);

JSONArray jsa1=JSONArray.fromObject(studentList1);

System.out.println(jsa1.toString());

System.out.print("将Map转换为json字符串:");

Map<String,String> map1=new HashMap<String,String>();

map1.put("sno", "20181111");

map1.put("sname","王五");

map1.put("sex","男");

JSONObject jso2=JSONObject.fromObject(map1);

System.out.println(jso2.toString());

System.out.print("将json字符串转换成student3对象:");

String jsonStr1="{'sex':'男','sname':'张三','sno':'20181234'}";

JSONObject jso3=JSONObject.fromObject(jsonStr1);

Student student3=(Student)JSONObject.toBean(jso3,Student.class);

System.out.println(student3);

System.out.print("将json字符串转换成studentList2对象:");

String jsonStr2="[{'sex':'男','sname':'张三','sno':'20181234'},{'sex':'女','sname':'李四','sno':'20185678'}]";

JSONArray jsa2=JSONArray.fromObject(jsonStr2);

List<Student> studentList2=(List<Student>)JSONArray.toCollection(jsa2,Student.class);

for(Student student:studentList2){

System.out.print(student+";");

}

}

}

Student类

public class Student{

String sno,sname,sex;

public Student() {

super();

}

public Student(String sno, String sname, String sex) {

super();

this.sno = sno;

this.sname = sname;

this.sex = sex;

}

public String toString(){

return "学号："+sno+",姓名："+sname+",性别："+sex;

}

public String getSno() {

return sno;

}

public void setSno(String sno) {

this.sno = sno;

}

public String getSname() {

return sname;

}

public void setSname(String sname) {

this.sname = sname;

}

public String getSex() {

return sex;

}

public void setSex(String sex) {

this.sex = sex;

}

}

程序运行结果如图10-3所示。

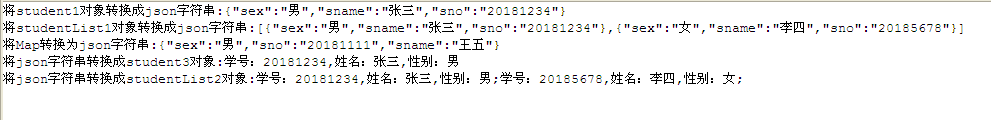


图10-3 例10-3运行结果

### 10.4.5 案例——基于Ajax+JSON的表格数据浏览

【例10-4】利用Ajax方式将服务器端返回的JSON格式的表格数据显示到页面上，运行结果如图10-4所示。

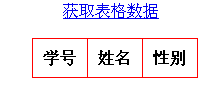


图10-4基于Ajax+JSon的表格数据浏览

【分析】为了只阐述清楚思路，减小代码篇幅，我们使用了前面的ajax.js里的函数和上个实例的实体类，并且省略了数据库操作代码，直接创建了List对象作为返回的表格数据，页面和Servlet代码如下：

页面代码：

<%@ page pageEncoding="UTF-8"%>

<html>

<head>

<script type="text/javascript" src="js/ajax.js"></script>

<script>

function query(){

sendRequest("query",null,'POST',show);

}

function show(){

if (httpRequest.readyState == 4) {

if (httpRequest.status == 200) {

var tBody=document.getElementById("tBody");

tBody.innerHTML="";

var studentList=JSON.parse(httpRequest.responseText);

for(var index in studentList){

var newTr=tBody.insertRow();

var newTd1=newTr.insertCell();newTd1.innerHTML=studentList[index].sno;

var newTd2=newTr.insertCell();newTd2.innerHTML=studentList[index].sname;

var newTd3=newTr.insertCell();newTd3.innerHTML=studentList[index].sex;

}

}

}

}

</script>

<style>

table,th,td{

border:1px solid red;

padding:10px;

}

table{

border-collapse:collapse;

margin:0 auto;

}

body{

text-align:center;

}

</style>

</head>

<body>

<a href="javascript:query()">获取表格数据</a>

<br/><br/>

<table id="studentTable">

<tr><th>学号</th><th>姓名</th><th>性别</th></tr>

<tbody id="tBody">

</tbody>

</table>

</body>

</html>

Servlet的doPost（）方法代码如下：

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

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

PrintWriter out = response.getWriter();

Student student1=new Student("20181234","张三","男");

Student student2=new Student("20185678","李四","女");

List<Student> studentList1=new ArrayList<Student>();

studentList1.add(student1);studentList1.add(student2);

JSONArray jsa=JSONArray.fromObject(studentList1);

out.print(jsa.toString());

}

## 10.5 jQuery框架中的Ajax方法

在jQuery框架中提供了对Ajax编程的支持，可以让我们简化代码，且能实现对各种浏览器的兼容。

### 10.5.1 jQuery框架中常用Ajax方法

常用方法介绍如下：

1．load（）方法

（1）功能：从服务器加载数据到当前页面的元素中。

（2）格式：$(selector).load(URL,data,callback);

URL：必选项，设置加载的 URL。

data ：可选项，请求字符串，键/值对集合（&隔开的字符串或JSON形式）。

Callback：可选项 ，设置load() 方法完成后所执行的函数。

（3）示例：加载servlet（地址为/load）返回的内容到div中。

页面代码如下：

<%@ page pageEncoding="UTF-8"%>

<html>

<head>

<script src="js/jquery-1.11.1.js"></script>

<script>

function jiazai(){

//当传递参数时，将执行servlet的doPost（）方法

$("#div1").load("load",{“p”:"jsp程序设计"});

}

</script>

</head>

<body>

<input type="button" value="加载数据" onclick="jiazai()">

<div id="div1"></div>

</body>

</html>

Servlet的doPost（）方法为

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

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

PrintWriter out = response.getWriter();

out.println("传递过来的参数为:"+request.getParameter("p”));

out.flush();

out.close();

}

2．$.get() 方法

（1）功能：以GET方式从服务器请求数据。

（2）格式：$.get(URL,data,callback);

URL：必选项，设置请求的 URL。

data ：可选项，请求字符串，键/值对集合（&隔开的字符串或JSON形式）。

Callback：可选项 ，设置请求成功后所执行的函数。

（3）示例：请求servlet（地址为/get），将返回的内容放到div中。

页面代码如下：

<%@ page pageEncoding="UTF-8"%>

<html>

<head>

<script src="js/jquery-1.11.1.js"></script>

<script>

function getF(){

//执行servlet的get方法

$.get("get",{"p":"abc"},function(data,status){

$("#div1").html("数据为："+data+"，状态为"+status);

});

}

</script>

</head>

<body>

<input type="button" value="get方式请求数据" onclick="getF()">

<div id="div1"></div>

</body>

</html>

Servlet的doGet方法如下：

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

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

PrintWriter out = response.getWriter();

out.print("发送的参数为："+request.getParameter("p"));

out.flush();

out.close();

}

3．$.post() 方法

（1）功能：以POST方式从服务器请求数据。

（2）格式：$.post(URL,data,callback);

URL：必选项，设置请求的 URL。

data ：可选项，请求字符串，键/值对集合（&隔开的字符串或JSON形式）。

Callback：可选项 ，设置请求成功后所执行的函数。

（3）示例：同get类似，读者可自行补上。

4．$.ajax() 方法

（1）功能：jQuery的底层Ajax实现，简单易用的高层实现见 $.get, $.post 等。

（2）格式：$.ajax({name:value, name:value, ... });

参数只有一个，是多个键值对的集合，常用键包括url，data，type，success，error等。

（3）示例：使用$.ajax（）完成$.post（）的功能。

页面代码如下：

<%@ page pageEncoding="UTF-8"%>

<html>

<head>

<script src="js/jquery-1.11.1.js"></script>

<script>

function ajaxF(){

//执行servlet的post方法

$.ajax({

url:"post",

type:"post",

data:{"p":"jsp程序设计"},

success:function(data,status){

$("#div1").html("数据为："+data+"，状态为"+status);

}

});

}

</script>

</head>

<body>

<input type="button" value="ajax方式请求数据" onclick="ajaxF()">

<div id="div1"></div>

</body>

</html>

Servlet的doPost方法如下：

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

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

PrintWriter out = response.getWriter();

out.print("传递过来的参数值为："+request.getParameter("p"));

out.flush();

out.close();

}

5．serialize()和serializeArray()方法

（1）功能：两个方法功能类似，都用来序列化一组表单元素，前者返回一组键值对字符串，后者返回JSON对象。

（2）示例：利用serialize()和serializeArray()方法生成表单提交参数字符串。

页面代码如下：

<%@ page pageEncoding="UTF-8"%>

<html>

<head>

<script src="js/jquery-1.11.1.js"></script>

<script>

function f1(){

$("#result1").text($("#form1").serialize());

}

function f2(){

$("#result2").text(JSON.stringify($("#form1").serializeArray()));

}

</script>

</head>

<body>

<form id="form1">

学号: <input type="text" name="sno" /><br>

姓名: <input type="text" name="sname" /><br>

性别: 男<input type="radio" name="sex" value="男" checked />

女<input type="radio" name="sex" value="女" />

<br>

</form>

<input type="button" value="serialize序列化表单值" onclick="f1()">

<input type="button" value="serializeArray序列化表单值" onclick="f2()">

<div id="result1"></div>

<div id="result2"></div>

</body>

</html>

说明：JSON.stringify（）方法的作用是将JSON对象转换为字符串。

结果如图10-5所示。

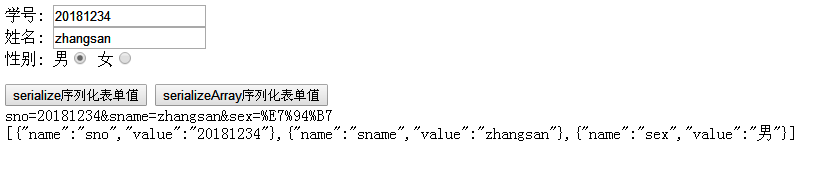


图10-5表单元素序列化

### 10.5.2案例——基于Ajax+jQuery的表格记录添加

【例10-5】利用Ajax+jQuery完成学生记录的添加。运行结果如图10-6所示。

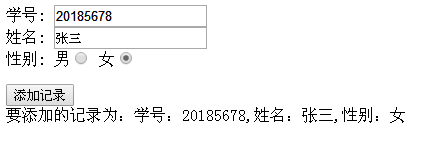


图10-6基于Ajax+jQuery的表格记录添加

【分析】为了只阐述清楚思路，减小代码篇幅，在此省略了数据库操作代码，页面和Servlet代码如下。

页面代码：

<%@ page pageEncoding="UTF-8"%>

<html>

<head>

<script src="js/jquery-1.11.1.js"></script>

<script>

function add(){

$.post("add",$("#form1").serialize(),function(data){

$("#result").text(data);

}

);

}

</script>

</head>

<body>

<form id="form1">

学号: <input type="text" name="sno" /><br>

姓名: <input type="text" name="sname" /><br>

性别: 男<input type="radio" name="sex" value="男" checked />

女<input type="radio" name="sex" value="女" />

<br>

</form>

<input type="button" value="添加记录" onclick="add()">

<div id="result"></div>

</body>

</html>

Servlet的doPost（）方法：

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

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

PrintWriter out = response.getWriter();

String sno=request.getParameter("sno");

String sname=request.getParameter("sname");

String sex=request.getParameter("sex");

out.print("要添加的记录为：学号："+sno+",姓名："+sname+",性别："+sex);

out.flush();

out.close();

}

## 本章小结

本章介绍了Ajax编程技术，包括Ajax与传统编程模式的比较、Ajax的核心对象XMLHttpRequest、Ajax编程的实现步骤、使用JSON进行数据传输以及jQuery框架中对Ajax的支持等内容。

## 习题

1. 比较Ajax和前面学习的编程模式有何不同？

2．设计一个注册表单，当用户填写注册id时，页面即时给出该id是否可用，利用Ajax技术实现。

3. 去网上下载我国省市区数据库，实现省、市、区三级级联列表。

4．基于Ajax功能实现表格的添加、修改、删除、查询功能。

5．利用Ajax简单实现搜索引擎的“Search Suggest”功能。