# 

**Ajax学习笔记**

**Java相关课程系列笔记之十一**

**笔记内容说明**

Ajax（程祖红老师主讲，占笔记内容100%）；

**目 录**

[一、 Ajax概述](#_Toc878)

[1.1什么是Ajax](#_Toc759)

[1.2 Ajax对象：如何获得Ajax对象](#_Toc22146)

[1.3 Ajax对象的属性](#_Toc12060)

[1.4编程步骤](#_Toc3722)

[1.5编码问题](#_Toc19389)

[1.6 Ajax的优点](#_Toc31776)

[1.7缓存问题（IE浏览器）](#_Toc946)

[1.8案例：简易注册（使用Ajax进行相关验证，get请求）](#_Toc28861)

[1.9案例：修改1.8案例，使用post请求](#_Toc29835)

[1.10案例：使用Ajax实现下拉列表](#_Toc17854)

[二、 JSON](#_Toc17403)

[2.1什么是JSON](#_Toc7350)

[2.2数据交换](#_Toc23910)

[2.3轻量级](#_Toc11042)

[2.4 JSON语法（www.json.org）](#_Toc313)

[2.5如何使用JSON来编写Ajax应用程序](#_Toc13568)

[2.6案例：股票的实时行情](#_Toc23480)

[2.7案例：显示热卖的前3个商品](#_Toc13527)

[2.8同步请求](#_Toc18263)

[2.9案例：修改1.8案例step1中的JS代码（使用同步请求）](#_Toc25898)

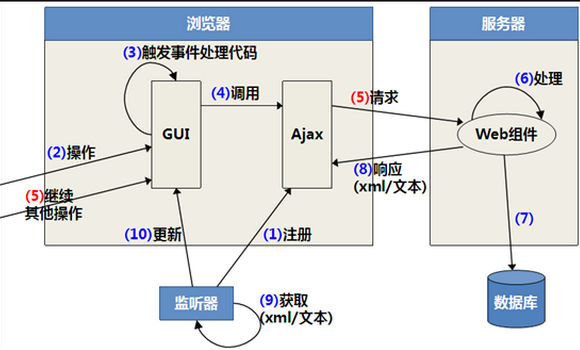
# Ajax概述

## 1.1什么是Ajax

Asynchronous Javascript And Xml 异步的JavaScript和Xml，是一种用来改善用户体验的技术，其实质是利用浏览器内置的一个特殊对象：XMLHttpRequest，一般称之为Ajax对象。Ajax对象在向服务器发送请求时，浏览器并不会销毁当前页面，用户仍然可以对当前页面作其他的操作。异步地向服务器发送请求，服务器送回部分数据，不是一个完整的新的页面，而是文本或者Xml文档。在浏览器端，可以利用这些数据部分更新当前页面。整个过程，页面无刷新，不打断用户的操作。

之前，都是先销毁原来的页面，然后发送请求，等待服务器发送响应，再生成新页面。

Ajax的工作流程：



## 1.2 Ajax对象：如何获得Ajax对象

由于XMLHttpRequest（Ajax对象）没有标准化，所以要**区分浏览器**。

function getXhr(){//注意：后面的案例**都将用到**此函数

var xhr=null;

if(window.XMLHttpRequest){

xhr=new XMLHttpRequest();//非IE浏览器

}else{

xhr=new ActiveXObject('Microsoft.XMLHttp');//IE浏览器

}

return xhr;

}

* 注意事项：后面的案例**也会用到**以下函数

function $(id){//依据id返回dom节点

return document.getElementById(id);

}

function $F(id){//返回id对应的值

return $(id).value;

}

## 1.3 Ajax对象的属性

1）onreadystatechange：绑定一个事件处理函数（监听器），该函数用来处理readystatechange事件。Ajax对象的readyState属性发生改变，比如从0到1，则会产生onreadystatechange事件。

2）responseText：获得服务器返回的文本数据。

3）responseXML：获得服务器返回的Xml文档。

4）status：获得状态码。

5）readyState：返回Ajax对象与服务器通讯的状态，返回值是一个number类型的值。一共有5个值，分别是：

①0：（未初始化）对象已建立，但是尚未初始化（尚未调用open方法）。

②1：（初始化）对象已建立，尚未调用send方法。

③2：（发送数据）send方法已调用。

④3：（数据传送中）已接收部分数据。

⑤4：（响应结束）Ajax对象已经获得了服务器返回的所有的数据。

## 1.4编程步骤

1）发送get请求：

step1：获得Ajax对象，比如：var xhr=getXhr();//调用之前定义的函数

step2：使用Ajax对象发送get请求

①调用xhr.open('get',check\_username.do?username=chang&age=23,true);方法：建立与服务器之间的连接，三个参数依次为：请求方式、请求资源路径、请求是同步还是异步。

true：表示异步请求（Ajax对象发送请求时，用户可以对当前页面作其他的操作，不会销毁页面）。

false：表示同步请求（Ajax对象发送请求时，浏览器会锁定当前页面，用户只能等待，不会销毁页面）。

②xhr.onreadystatechange=func1();：绑定一个事件处理函数（监听器）

③xhr.send(null);：发送请求参数，因为参数已经写在了请求资源路径中，所以这里为null。

step3：编写服务器端的处理程序，跟以前相比，有一点点改变，就是一般不需要返回一个完整的页面，只需要返回部分的数据。

step4：编写事件处理函数

function f1(){

if(xhr.readyState==4){

var txt=xhr.responseText;

dom操作……

}

}

2）发送post请求：

step1：获得Ajax对象，比如：var xhr=getXhr();//调用之前定义的函数

step2：使用Ajax对象发送post请求

①xhr('post','check\_username.do',true);：建立连接

②xhr.setRequestHeader('content-type','application/x-www-form-urlencoded');：发送一个content-type消息头

③xhr.onreadystatechange=func1();：绑定一个事件处理函数（监听器）

④xhr.send('username=chang');：发送请求参数

* 注意事项：
* 与get请求不同，请求参数要放到send方法里面。
* 因为按照HTTP协议的要求，发送post请求时，应该发送一个content-type消息头，而Ajax对象在默认情况下，不会发送这个消息头，所以，需要调用setRequestHeader方法来添加。

step3：编写服务器端的处理程序，跟以前相比，有一点点改变，就是一般不需要返回一个完整的页面，只需要返回部分的数据。

step4：编写事件处理函数

## 1.5编码问题

1）发get请求

①乱码产生的原因：

IE浏览器内置的Ajax对象会使用“GBK”或“GB2312”对中文参数进行编码，而其他浏览器（Chrom、Firefox）内置的Ajax对象会使用“utf-8”对中文参数进行编码。服务器端，默认会使用“ISO-8859-1”去解码。因为编码与解码所使用的字符集（编码格式）不一致，所以，会出现乱码问题。

②解决：

step1：设置服务器使用指定的字符集去解码。比如，可以修改Tomcat的server.xml配置（conf文件夹中），添加URIEncoding="utf-8"（告诉服务器，对于所有的get请求，默认使用“utf-8”去解码），修改之后重新启动服务器。

<Connector port="8080" maxHttpHeaderSize="8192"

maxThreads="150" minSpareThreads="25" maxSpareThreads="75"

enableLookups="false" redirectPort="8443" acceptCount="100"

connectionTimeout="20000" disableUploadTimeout="true" URIEncoding="utf-8" />

step2：使用encodeURI()函数（JS中内置函数）对请求地址进行编码。encodeURI()函数会使用“utf-8”进行编码

xhr.open('get','check\_username.do?username='+$F('username'),true);

var uir='check\_username.do?username='+$F('username');

xhr.open('get',encodeURI(uri),true);

2）发post请求

①乱码问题产生的原因：

所有浏览器（一般指三大浏览器：Chrom、Firefox、IE）内置的Ajax对象都会使用“utf-8”对中文参数进行编码，而服务器默认情况下，会使用“ISO-8859-1”去解码。

* 注意事项：Firefox特殊，本应是乱码，但能正常显示。通过截取消息头发现Firefox会在消息头中自动添加"charset=utf-8"。

②解决：

服务器端添加：request.setCharacterEncoding("utf-8");

## 1.6 Ajax的优点

1）页面无刷新，不打断用户的操作。

2）按需要获取数据，客户端（浏览器）与服务器端之间的数据传输量大大减少。

3）是一种标准化的技术，不需要下载任何插件。

## 1.7缓存问题（IE浏览器）

1）当发送get请求时

IE浏览器（其他浏览器没这个问题）内置的Ajax对象会检查请求地址是否访问过，如果访问过，则不再向服务器发送请求。

2）解决方式

方式一：在请求地址后面添加一个随机数，用于欺骗IE，例如：

xhr.open('get','getNumber.do?'+Math.random(),true);

方式二：使用post方式发请求。

3）案例：在IE浏览器中测试缓存问题

step1：getNumber.jsp页面

<a href="javascript:;" onclick="getNumber();">点这儿，在链接底下显示一个随机数</a>

<div id="d1"></div>

step2：JS代码getNumber函数

function getNumber(){

var xhr=getXhr();

xhr.open('get','getNumber.do?'+Math.random(),true);//若没随机数，则点链接不会变化

xhr.onreadystatechange=function(){

if(xhr.readyState==4){

var txt=xhr.responseText;

$('d1').innerHTML=txt; } };

xhr.send(null); }

step3：Servlet中service方法中的if判断

if(action.equals("/getNumber")){

Random r=new Random();

int number=r.nextInt(10000);

System.out.println(number);

out.println(number); }

## 1.8案例：简易注册（使用Ajax进行相关验证，get请求）

step1：编写myScript.js并放在js文件夹中

此处省略三个函数，详看1.2

function check\_username(){//检查用户名，get请求

var xhr=getXhr();//step1获得Ajax对象

//step2使用Ajax对象发送请求

xhr.open('get','check\_username.do?username='+$F('username'),true);

//绑定一个事件处理函数，里面的代码在状态改变时执行，且状态为4时执行if语句

xhr.onreadystatechange=function(){

if(xhr.readyState==4){

if(xhr.status==200){//服务器返回了正确的结果

//只有readyState为4时，Ajax对象才获得服务器返回的所有数据

var txt=xhr.responseText;

$('username\_msg').innerHTML=txt;//利用服务器返回的数据更新页面

}else{//服务器运行出错

$('username\_msg').innerHTML='验证出错'; } }

};

$('username\_msg').innerHTML='正在验证...';//模拟用户量较大的情况，显示正在验证

xhr.send(null); }

function check\_number(){//检查验证码

var xhr=getXhr();

xhr.open('get','check\_number.do?number='+$F('number'),true);

xhr.onreadystatechange=function(){

if(xhr.readyState==4){

var txt=xhr.responseText;

$('number\_msg').innerHTML=txt; } };

xhr.send(null); }

step2：编写regist.jsp页面，get请求

<form action="regist.do" method="get">

<fieldset>

<legend>注册</legend>

用户名：<input type="text" name="username" id="username"

onblur="check\_username();"/>

<span class="tips" id="username\_msg"></span><br/>

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

验证码：<input type="text" name="number" id="number"

onblur="check\_number();" />

<img src="checkcode" id="img1" onclick="this.src='checkcode?'+Math.random();" />

<a href="javascript:;" onclick="$('img1').src='checkcode?'+Math.random();">

看不清换一个</a>

<span class="tips" id="number\_msg"></span><br/>

<input type="submit" value="提交" />

</fieldset>

</form>

* 注意事项：href="javascript:;"相当于href="#"，写成href="javascript:;"一般要和onclick事件一起使用，表示a元素不再指向一个地址，而是点击后触发一个事件。

step3：ActionServlet中service方法

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

PrintWriter out = response.getWriter();

String uri = request.getRequestURI();

String action = uri.substring(uri.lastIndexOf("/"), uri.lastIndexOf("."));

if(action.equals("/check\_username")) {

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

// try {//模拟耗时操作

// Thread.sleep(6000); } catch (InterruptedException e) { e.printStackTrace(); }

if(1==1){//模拟一个系统异常

throw new ServletException("some error"); }

if(username.equals("常")){

out.println("已被占用"); }else{ out.println("可以使用"); }

}else if(action.equals("/check\_number")) {

String number1=request.getParameter("number");

HttpSession session=request.getSession();

String number2=(String)session.getAttribute("number");

if(number1.equalsIgnoreCase(number2)){ out.println("验证码正确");

}else{ out.println("验证码错误"); }

}else if(action.equals("/regist")){

//加上验证代码，比如检查用户名是否正确，验证码是否正确，此处略

System.out.println("注册成功"); }

step4：CheckcodeServlet借用之前Servlet笔记中13.20案例，随机生成验证码

## 1.9案例：修改1.8案例，使用post请求

step1：修改regist.jsp页面请求方式为post

step2：添加JS验证代码check\_username\_post

function check\_username\_post(){

var xhr=getXhr();

xhr.open('post','check\_username.do',true);

//添加一个消息头

xhr.setRequestHeader('content-type','application/x-www-form-urlencoded');

xhr.onreadystatechange=function(){

if(xhr.readyState==4){

var txt=xhr.responseText;

$('username\_msg').innerHTML=txt; } };

xhr.send('username='+$F('username')); }

## 1.10案例：使用Ajax实现下拉列表

step1：ActionServlet中service方法中的if判断

if(action.equals("/select")){

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

if(name.equals("qq")){ out.println("性价比高");

}else if(name.equals("bmw")){ out.println("驾驶性能出众");

}else{ out.println("好车,也贵"); } }

step2：select.jsp页面

<select id="s1" onchange="change(this.value);">

<option value="qq">QQ</option> <option value="bmw">宝马</option>

<option value="ff">法拉利</option></select>

<div id="d1"></div>

step3：JavaScript代码

function change(value){

var xhr=getXhr(); xhr.open('get','select.do?name='+value,true);

xhr.onreadystatechange=function(){

if(xhr.readyState==4){ var txt=xhr.responseText; $('d1').innerHTML=txt; }

};

xhr.send(null); }

# JSON

## 2.1什么是JSON

JavaScript Object Natation，是一种轻量级的数据交换技术规范（因为借鉴了JavaScript对象创建的一种语法结构，故命名为JSON，详情见JavaScript第8章）。

## 2.2数据交换

将数据转换成一种中间的，与平台无关的数据格式（比如Xml或者JSON字符串）发送给另外一方来处理。

## 2.3轻量级

JSON相对于Xml，所需的数据大小要小的多，并且解析的速度更快。因此Xml现在用的少了（Ajax中的x即指用Xml交换数据）。

## 2.4 JSON语法（www.json.org）

可以使用{} [] : , “”这些字符来描述json规则

以{}开始时，是在用json描述一个对象

以[]开始时，是在用json描述一个数组

虽然javascript语言也用这两种符号来表达对象和数组，但只能说它与json的语法规则比较类似，实质上两者是一个不同的东西。

需要注意的是，属性名一定要用双引号隔开引起来，属性值如果是String类型也必须用双引号隔起来。但在javascript语言中，对象就没有这么严格的规定属性名可以不引或单引 或双引。属性值如果是String类型可以单引或双引

虽然json是网络中传输的字符串的格式，但是当符合json格式的字符串到达客户端或服务器时，客户端或服务器端要针对这段符合json格式的字符串进行操作，所以都该完成共json字符串到集体语言对象的转换。

如果当前json字符串到达的是客户端，那么就要完成从json字符串到javascript对象的一个转换。

stringfy()方法：javascript对象—>json字符串

parse()方法：json字符串—>javascript对象

javascript 对象🡨🡪json字符串

java对象—>json字符串

java集合—>json 字符串

json字符串-🡪java对象

json字符串→java对象

使用浏览器内的原生的JSON对象的stringfy()方法和parse方法

* 注意事项：
* 属性名要使用引号括起来。
* 属性值如果是字符串，要使用引号括起来。
* 属性值可以是string、number、boolean、null、object。

例如：function f1(){//表示一个对象

var obj={'name':'chang','age':22}; alert(obj.name); }

function f2(){//表示一个对象

var obj={'name':'bo','address':{ 'city':'beijing', 'room':'1101' } };

alert(obj.address.room); }

* 注意事项：JavaScript中创建对象的三种方式（也可看JavaScript笔记中的第8章）：

方式一：使用Json语法来创建

var p = {'name':'zs','age':22};

方式二：使用Object来创建

var obj = new Object(); obj.name = 'zs'; obj.age = 22;

方式三：利用JavaScript函数来创建（对象模版）

function Person(name,age){

this.name = name; this.age = age; }

var person1 = new Person("zs",22)

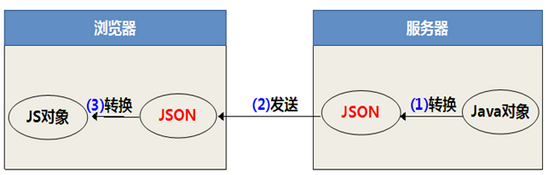
2）如何表示一个对象组成的数组，[ {},{},{},……]

例如：function f3(){//表示一个对象组成的数组

var arr=[{'name':'chang','age':22},{'name':'bo','age':23}];

alert(arr[1].name); }

## 2.5如何使用JSON来编写Ajax应用程序



step1：Java对象转换成JSON字符串（就是普通字符串用了JSON语法而已的字符串）

一般使用JSON官方提供的API（json-lib）来实现转换（6个包：1主5副）。也可用谷歌提供的API。

例如：股票实体类Stock，有name、code、price三个属性和相应的get/set方法

情形一：Java对象转换成一个JSON字符串，使用JSONObject.fromObject()

/\*\* 效果：{'name':'chang','code':'10086','price':12.8} \*/

public static void test1(){

Stock s=new Stock(); s.setName("chang"); s.setCode("10086"); s.setPrice(12.8);

JSONObject obj=JSONObject.fromObject(s);//得到一个json对象

String jsonStr = obj.toString(); System.out.println(jsonStr); }

情形二：Java对象组成的集合转换成一个JSON字符串，使用JSONArray.fromObject()

/\*\* 效果：[{'name':'chang','code':'10086','price':12.8},

{'name':'chang','code':'100861','price':12.81},{'name':'chang','code':'1008611','price':12.82}] \*/

public static void test2() {

List<Stock> stocks=new ArrayList<Stock>();

Random r=new Random(); DecimalFormat df=new DecimalFormat("#.##");

for(int i=0;i<3;i++){ Stock s=new Stock(); s.setName("chang"+r.nextInt(10));

s.setCode("60001"+r.nextInt(10)); double price=r.nextInt(100)+r.nextDouble();

s.setPrice(Double.parseDouble(df.format(price))); stocks.add(s); }

JSONArray array=JSONArray.fromObject(stocks);

String jsonStr=array.toString(); System.out.println(jsonStr); }

情形三：Java对象组成的数组转换成一个JSON字符串，使用JSONArray.fromObject()

public static void test3() {

Stock[] stocks = new Stock[3]; Random r=new Random();

DecimalFormat df=new DecimalFormat("#.##");

for(int i=0;i<3;i++){ Stock s=new Stock(); s.setName("chang"+r.nextInt(10));

s.setCode("60001"+r.nextInt(10)); double price=r.nextInt(100)+r.nextDouble();

s.setPrice(Double.parseDouble(df.format(price))); stocks[i]=s; }

JSONArray array=JSONArray.fromObject(stocks);

String jsonStr=array.toString(); System.out.println(jsonStr); }

step2：JSON字符串转换成JavaScript对象

可以使用一些工具提供的方法，比如prototype提供了evalJSON()函数，prototype是一个js文件，里面提供了很多常用的函数，比如：

1）$(id)：document.getElementBuId(id);

2）$F(id)：$(id.value);

3）$(id1,id2,id3...)：分别依据id1,id2...查找对应的节点，然后放到一个数据里面返回。

4）strip()：除掉字符串两端的空格。trim()函数也有，但有的浏览器不支持。

5）evalJSON()：将JSON字符串转换成对应的JavaScript对象或者JavaScript对象组成的数组。

例如：//将一个JSON字符串转成一个JavaScript对象

function f4(){ var str="{'name':'chang','age':24}"; //alert(typeof str);

//使用prototype框架提供的evalJSON函数，将字符串转成一个JavaScript对象

var obj=str.evalJSON();//别忘记引入.js文件

alert(obj.name); }

//将JSON字符串转换成一个JacaScript对象组成的数组

function f5(){ var str="[{'name':'chang','age':22},{'name':'bo','age':23}]";

var arr=str.evalJSON(); alert(arr[1].name); }

## 2.6案例：股票的实时行情

step1：股票实体类Stock，有name、code、price三个属性和相应的get/set方法

step2：ActionServlet中service方法中的if判断

if(action.equals("/quoto")){

/\*\* 模拟生成八只股票的价格信息，然后将这些信息转换成JSON字符串，并发送到客户端 \*/

List<Stock> stocks=new ArrayList<Stock>();

Random r=new Random(); DecimalFormat df=new DecimalFormat("#.##");

for(int i=0;i<8;i++){ Stock s=new Stock(); s.setName("chang"+r.nextInt(30));

s.setCode("6000"+r.nextInt(30)); double price=r.nextInt(100)+r.nextDouble();

s.setPrice(Double.parseDouble(df.format(price))); stocks.add(s); }

JSONArray array=JSONArray.fromObject(stocks);//将Java数组转成JSON字符串

String jsonStr=array.toString(); System.out.println(jsonStr);

out.println(jsonStr);//将JSON字符串发送到客户端 }

step3：stock.jsp页面表格（表格样式忽略）

<body onload="f1();">

<div id="d1"><div id="d2">股票实时行情</div>

<div id="d3"><table cellpadding="0" cellspacing="0" width="100%">

<thead><tr><td>股票名称</td><td>股票代码</td><td>股票价格</td></tr></thead>

<tbody id="tb1"></tbody></table></div>

</div></body>

step4：JavaScript代码

function f1(){ setInterval(quoto,3000); }

function quoto(){ var xhr = getXhr(); xhr.open('get','quoto.do',true);

xhr.onreadystatechange = function(){

if(xhr.readyState == 4){ var txt = xhr.responseText;

var arr = txt.evalJSON();//将json字符串转换成javascript对象组成的数组

var html = '';//将数组中的数据取出来，添加到tbody

for(i=0;i<arr.length;i++){ html +='<tr><td>' + arr[i].name+ '</td><td>'

+ arr[i].code + '</td><td> ' + arr[i].price+ '</td></tr>'; }

$('tb1').innerHTML = html; } };

xhr.send(null); }

* 注意事项：innerHTML属性对于IE浏览器只能对td赋值！对表格table里的其他节点，如tr、tbody、thead、caption都不能赋值，只能用它去读（兼容性问题）。

## 2.7案例：显示热卖的前3个商品

step1：建表，实体类Sale有属性name和qty，以及相应get/set方法。DBUtil类此处省略

step2：SaleDAO

public List<Sale> limit(int top) throws Exception{

List<Sale> list=new ArrayList<Sale>(); Connection conn=DBUtil.getConnection();

PreparedStatement prep=conn.prepareStatement(

"select \* from chang\_sale order by qty desc limit ?");

prep.setInt(1, top); ResultSet rs=prep.executeQuery(); Sale sale=null;

while(rs.next()){ sale=new Sale(); sale.setName(rs.getString("name"));

sale.setQty(rs.getInt("qty")); list.add(sale); }

DBUtil.close(conn); return list; }

step3：ActionServlet中service方法中的if判断

if(action.equals("/limit")){

int top=Integer.parseInt(request.getParameter("qty"));

SaleDAO dao=new SaleDAO(); List<Sale> sales=new ArrayList<Sale>();

try { sales=dao.limit(top); JSONArray array=JSONArray.fromObject(sales);

String jsonStr=array.toString(); out.println(jsonStr);

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

step4：JavaScript代码（jsp页面表格与2.6案例step3类似，此处忽略）

function f1(){ setInterval(quoto,3000); }

function quoto(){ var xhr = getXhr(); var top=5;

xhr.open('get','limit.do?qty='+top,true);

xhr.onreadystatechange = function(){

if(xhr.readyState == 4){ var txt = xhr.responseText;

var arr = txt.evalJSON(); var html = '';

for(i=0;i<arr.length;i++){ html +='<tr><td>' + arr[i].name

+ '</td><td>' + arr[i].qty + '</td></tr>'; }

$('tb1').innerHTML = html; } };

xhr.send(null); }

## 2.8同步请求

1）什么是同步请求

Ajax对象在向服务器发送请求时，浏览器会锁定当前页面，用户不能够对当前页面做任何的操作。

2）如何发送同步请求

open(请求方式，请求地址，false)

3）优先使用异步，因为同步会影响性能，当服务器端处理比较慢的时候，浏览器会锁定当前页面（“假死”）。

4）只有当客户端需要等待服务器的响应之后，才能继续向下执行时，应该使用同步。

* 注意事项：Firefox的某些版本（低版本，如3、4）对于同步的支持比较特殊：

不能使用xhr.onreadystatechange来绑定一个事件处理函数，而应该在send方法执行之后，才调用xhr.responseText方法来获得服务器返回的数据。例如：

function getTyoe(){//返回浏览器类型

if(navigator.userAgent.indexOf('Firefox')!=-1){ return 'firefox';

}else{ return 'other'; }//其他浏览器 }

function quoto(){

if(getType() != 'firefox'){

xhr.onreadystatechange=function(){

if(xhr.readyState == 4){ var txt = xhr.responseText; …… } }; }

xhr.send('username=' + $F('username'));

if(getType() == 'firefox'){ var txt = xhr.responseText; …… } }

## 2.9案例：修改1.8案例step1中的JS代码（使用同步请求）

step1：form表单增加属性onsubmit="return beforesubmit();"

step2：ActionServlet中service方法中的if判断

if(action.equals("/check\_username")){

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

if(username.equals("chang")){

out.print("error");//不能用println服务器会把换行也返回，则永远不会匹配成功

}else{ out.print("ok"); }

}else if(action.equals("/check\_number")) {

String number1=request.getParameter("number");

HttpSession session=request.getSession();

String number2=(String)session.getAttribute("number");

if(number1.equalsIgnoreCase(number2)){ out.print("ok");

}else{ out.print("error"); } }

step3：增加JS代码

function check\_username(){//step1：检查用户用是否为空

$('username\_msg').innerHTML="";//先清空之前的提示信息

if($F('username').strip().length==0){

$('username\_msg').innerHTML='用户名不能为空';

return false; }

var flag=false;//step2：检查用户名是否被占用

var xhr=getXhr(); xhr.open('post','check\_username.do',false);

xhr.setRequestHeader('content-type','application/x-www-form-urlencoded');

xhr.onreadystatechange=function(){

if(xhr.readyState==4){ var txt=xhr.responseText;

if(txt=='ok'){//注意服务器端println问题

$('username\_msg').innerHTML='用户名可以使用'; flag=true;

}else{ $('username\_msg').innerHTML='用户名被占用'; flag=false; }

} };

xhr.send('username='+$F('username'));

//如果是同步请求，浏览器不会执行以下的代码，而是等待服务器响应回来，在此期间，浏览器会锁定当前页面

return flag; }

function check\_number(){//检查验证码

$('number\_msg').innerHTML="";

if($F('number').strip().length==0){

$('number\_msg').innerHTML='验证码不能为空'; return false; }

var flag=false; var xhr=getXhr();

xhr.open('get','check\_number.do?number='+$F('number'),false);

xhr.onreadystatechange=function(){

if(xhr.readyState==4){

var txt=xhr.responseText;

if(txt=='ok'){//注意服务器端println问题

$('number\_msg').innerHTML='验证码正确'; flag=true;

}else{ $('number\_msg').innerHTML='验证码错误'; flag=false; }

} };

xhr.send(null); return flag; }

function beforesubmit(){//提交之前先检查

var flag=check\_username() && check\_name() && check\_number();

return flag; }