# 14.AJAX

## 14.1.AJAX概述

asynchronous js and xml: 异步的js和xml

可以利用js访问服务器, 而且是异步访问!

通常服务器给浏览器响应的是一个完整的页面, 而在AJAX中, 由于是利用js访问服务器, 再由js接受响应, 局部刷新页面, 所以服务器不用给浏览器响应整个页面了, 而只是数据。

服务器响应的数据:

> text 纯文本 "用户名已存在!!"

> xml

> json: js提供的一种数据交互格式, 在js中很受欢迎

ajax --> ajaj --> aj

**同步交互和异步交互**

**同步:**

向服务器发一个请求, 必须等待响应结束, 才能发送第二个请求, 在服务器处理期间, 浏览器不能干别的事儿

刷新整个页面

**异步:**

向服务器发一个请求, 不用等待响应结束, 就可以发送第二个请求, 在服务器处理期间, 浏览器可以干别的事儿

可以使用js接受服务器的响应, 再利用js局部刷新页面

**AJAX的应用场景**

百度的搜索框

注册用户时, 校验用户名是否被注册过

...

**AJAX的优点和缺点**

**优点:**

异步交互, 提高了用户体验!

服务器只响应部分数据, 而不是整个页面, 所以降低了服务器的压力!

...

**缺点:**

ajax不能应用所有的场景

ajax会无端的增加访问服务器的次数, 给服务器带来了压力!!

## 14.2.JavaScript实现AJAX

**第一步, 获取XMLHttpRequest对象**

var xmlHttp = ajaxFunction();

function ajaxFunction(){

var xmlHttp;

try{

//现代浏览器（IE7+、Firefox、Chrome、Safari 和 Opera）都有内建的 XMLHttpRequest 对象

xmlHttp = new XMLHttpRequest();

}catch(e){

try{

//IE6.0

xmlHttp = new ActiveXObject("Msxml2.XMLHTTP");

}catch(e){

try{

//IE5.0及更早版本

xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");

}catch(e){

alert("...");

throw e;

}

}

}

return xmlHttp;

}

**第二步, 打开与服务器的连接**

xmlHttp.open(method, url, async);

> method: 请求方式, 可以是GET或POST

> url: 所要访问的服务器中资源的路径 如: /Day10/servlet/AServlet

> async: 是否为异步传输, true 表示为异步传输 一般都是true

**第三步, 发送请求** xmlHttp.setRequestHeader("Content-Type","application/x-www-form-urlencoded"); //通知服务器发送的数据是请求参数

xmlHttp.send("xxxx"); //注意, 如果不给参数可能会造成部分浏览器无法发送请求

xmlHttp.send("username="+username+"&psw="+)

> 参数:

如果是GET请求, 可以是null, 因为GET提交参数会拼接在url后面

如果是POST请求, 传入的就是请求参数

"username=张飞&psw=123"

**第四步, 注册监听**

> 在XMLHttpRequest对象的一个事件上注册监听器:

onreadystatechange

> 一共有五个状态:(xmlHttp.readyState)

0状态: 表示刚创建XMLHttpRequest对象, 还未调用open()方法

1状态: 表示刚调用open()方法, 但是还没有调用send()方法发送请求

2状态: 调用完了send()方法了, 请求已经开始

3状态: 服务器已经开始响应, 但是不代表响应结束

4状态: 服务器响应结束!(通常我们只关心这个状态)

> 获取xmlHttp对象的状态:

var state = xmlHttp.readyState;//可能得到0, 1, 2, 3, 4

> 获取服务器响应的状态码

var status = xmlHttp.status;

> 获取服务器响应的内容

var data = xmlHttp.responseText;//得到服务器响应的文本格式的数据

readyState

0 --> 1

/\* 注册监听, 时刻监听服务器处理请求的状态 \*/

xmlHttp.onreadystatechange = function(){

if(xmlHttp.readyState == 4){

//状态为4表示服务器已经处理完成, 但是不代表处理成功

if(xmlHttp.status == 200){//当状态码为200, 表示服务器成功的处理了请求

var result = xmlHttp.responseText();

//xxx

}

}

}

**案例演示：**

新建web工程ajax\_ys，将课前资料中ajax\ajax\_ys下的src和WebRoot目录拷贝到工程中，部署并启动服务器，可以修改regist.jsp中代码（第44行）改变同步和异步访问（false为同步，true为异步）

|  |
| --- |
| xmlHttp.open("POST", "${ pageContext.request.contextPath }/servlet/AjaxTestServlet", **false**); |

## 14.3.jQuery实现AJAX

**load方法**

$(selector).load(url,data,callback);

selector -- 选择器, 将从服务器获取到的数据加载到指定的元素中

url -- 发送请求的URL地址

data -- 可选, 向服务器发送的数据 key/value数据 如:{"username" : "张飞", "psw" : "123"}

callback -- 可选, load方法完成后所执行的函数

示例:

$("#username\_msg").load("<%= request.getContextPath() %>/AjaxCheckUsernameServlet", {"username": username});

**$.get方法或$.post**

$.get(url, [data], [callback]);

url -- 发送请求的URL地址

data -- 可选, 向服务器发送的数据

callback -- 可选, 请求成功后所执行的函数

示例:

$.get("<%= request.getContextPath() %>/AjaxCheckUsernameServlet", {"username": username}, function(result){

$("#username\_msg").html("<font style='color:red'>"+result+"</font>");

});

**$.ajax方法**

$.ajax(url, [data], [async], [success]);

url -- 发送请求的URL地址

data -- 可选, 发送至服务器的key/value数据

async -- 可选, 默认为true, 表示异步交互

type -- 可选, 请求方式 , 默认为"GET"。

success -- 可选, 请求成功后执行的函数, 函数参数:

result -- 服务器返回的数据

示例:

$.ajax({

"url" : "<%= request.getContextPath() %>/AjaxCheckUsernameServlet",

"data" : {"username": username},

"async" : true,

"type" : "POST",

"success" : function(result){

$("#username\_msg").html("<font style='color:red'>"+result+"</font>")

}

});

## 14.4.Easymall用户名检查

代码如下：

|  |
| --- |
| 在EasyMall项目下，regist.jsp页面中，用户名对应的<span>标签中添加id属性：  <tr>  <td class=*"tds"*>用户名：</td>  <td>  <input type=*"text"* name=*"username"* value=*"*<%=request.getParameter("username") == **null**?"":request.getParameter("username")%>*"*/>  <span id=*"username\_msg"*></span>  </td>  </tr> |
| 在regist.jsp页面中，<script>标签中失去输入焦点事件下的用户名中添加、修改如下代码：  $("input[name='username']").blur(**function**(){  //如果用户名为空，则直接return  **if**(!formObj.checkNull("username", "用户名不能为空！")){  **return**;  }  //如果用户名不为空，使用ajax检查用户名是否存在，并提示  **var** username = $(**this**).val();  //使用load函数实现ajax  $("#username\_msg").load("<%= request.getContextPath() %>/servlet/AjaxCheckUsernameServlet", {"username",username});  }); |
| 在web包下创建servlet：AjaxCheckUsernameServlet，并添加如下代码：  //1.处理乱码（load方法为post提交）  request.setCharacterEncoding("utf-8");  response.setContentType("text/html;charset=utf-8");  //2.获取请求参数（用户名）  String username = request.getParameter("username");  //3.检查用户名是否存在  Connection conn = **null**;  PreparedStatement ps = **null**;  ResultSet rs = **null**;  **try** {  //从连接池中获取连接  conn = JDBCUtils.*getConnection*();    //>>用户名是否存在（如果存在，跳转回注册页面并提示）  String sql = "select \* from user where username=?";  ps = conn.prepareStatement(sql);  ps.setString(1, username);  rs = ps.executeQuery();  **if**(rs.next()){ //用户名已存在  response.getWriter().write("用户名已存在!");  }**else**{  response.getWriter().write("恭喜您，用户名可以使用!");  }  } **catch** (Exception e) {  e.printStackTrace();  **throw** **new** RuntimeException();  }**finally**{  //释放资源  JDBCUtils.*close*(conn, ps, rs);  } |

## 14.5.作业练习

1. 什么是ajax ？

答：异步的js和xml，是一种在无需重新加载整个网页的情况下，能够更新部分网页的技术。

通常服务器给浏览器响应的是一个完整的页面, 而在AJAX中, 由于是利用js访问服务器, 再由js接受响应, 局部刷新页面, 所以服务器不用给浏览器响应整个页面了, 而只是数据。

1. 同步和异步的区别？

答：同步：阻塞的

浏览器向服务器请求数据，服务器比较忙，浏览器一直等着，直到服务器返回数据，浏览器才能显示页面

异步：非阻塞的

浏览器向服务器请求数据，服务器比较忙，浏览器可以自如的干原来的事情，服务器返回数据的时候通知浏览器一声，浏览器把返回的数据再渲染到页面，局部更新