AJAX编程

1. 一头扎进ajax和json
   1. 第一讲
      1. ajax简介
         1. 是一种网页开发技术（Asynchronous Javascript+XML）一部Javascript和xml
         2. 是异步交互，局部刷新
         3. ajax能减少服务器压力
         4. 能提高用户体验
      2. ajax交互与传统交互比较
         1. 传统交互：网页整体刷新，服务器压力大，用户体验不好
         2. ajax：局部刷新，服务器压力小，用户体验好
   2. 第二讲
      1. XMLHttpRequest对象的创建

所有现代浏览器均支持XMLHttpRequest对象（IE5和IE6使用activeXObject），XMLHTTPRequest用于后台与服务器交换数据，这意味着可以在不重新加载整个页面的情况下，对网页的某部分进行更新

|  |
| --- |
| <script type=*“text/javascript”*>  **function** loadName(){  **var** xhr;  **if**(window.XMLHttpRequest){  xhr = **new** XMLHttpRequest();  }**else**{  xhr = **new** ActiveXObject(“Microsoft.XMLHTTP”);  }  }  </script> |

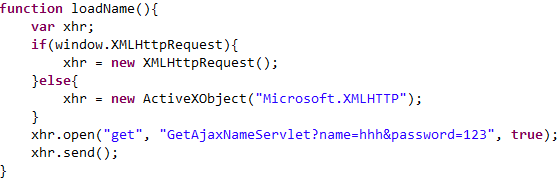
* + 1. XMLHttpRequest对象请求后台

open（method,url,async）规定请求的类型、url、以及是否异步处理请求

method：请求的类型，get或者post

url:文件在服务器上 的位置

async：true异步false同步



send（string）将请求发送到服务器

String ：仅用于Post请求

get还是post？

与post相比，get更简单也更快，并且在大部分情况下都能用

然而，在以下情况下，请使用post请求：

1. 无法使用缓存文件（更新服务器上的文件或数据库）
2. 向服务器发送大量数据（post没有数据量的限制）
3. 发送包含未知字符的用户输入时，post比get更稳定也更可靠

setRequestHeader(header,value)向请求添加http头

header：规定头的名称

value：规定头的值

xhr.setRequestHeader(“Content-type”,”application/x-www-form-urlencoded”);

异步 –true或者false

AJAX指的是异步 JavaScript 和 XML

为true的话，表示的是异步，异步表示程序请求服务器的同时，程序可以继续执行，能提高系统的运行效率

为false的话，表示同步，JavaScript会等到服务器响应就绪才继续执行，入股服务器繁忙或缓慢，应用程序会刮起或停止

一般都是用true

* + 1. XMLHttpRequest对象响应服务器
       1. onreadystatechange事件

当请求被发送到服务器时，我们需要执行一些基于响应的任务。

每当readyState改变是，就会触发onreadystatechange事件

readyState属性存有XMLHttpRequest的状态信息

下面是XMLHttpRequest对象的三个重要属性：

onreadystatechange存储函数（或函数名），每当readystate属性改变时，就会调用该函数

readyState

存有XMLHttpRequest的状态，从0-4发生变化

0：请求未初始化

1：服务器连接已建立

2：请求已接收

3：请求处理中

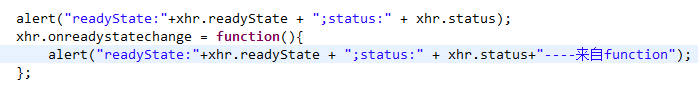
4：请求已完成，且响应已就绪

status

200：”ok”

404：未找到页面

在onreadystatechange事件中，我们规定当服务器响应义做好被处理的准备时所执行的任务



如需获得来自服务器的响应，请使用XMLHttpRequest对象的responseText或者responseXML属性。

|  |  |
| --- | --- |
| 属性 | 描述 |
| responseText | 获得字符串形式的响应数据 |
| ResponseXML | 获得xml形式的响应数据（了解） |

接收响应的时候，是在readyState=4并且status=200时候开始的



* 1. 第三讲
     1. json引入

json：javascript对象表示法（JavaScript object notation）

json是存储和交换文本信息的语法，类似xml

json比xml更加小，更快，更容易解析

* + 1. json格式语法

json对象：

{“name”:”张三”,”password”:”123”}

json数组：

{

“student”:[

{“name”:”张三”,”password”:”123”}

{“name”:”李四”,”password”:”1234”}

{“name”:”王五”,”password”:”12345”}

]

}

json嵌套

{

“student”:[

{“name”:”张三”,”password”:”123”,”score”:{“chinese”:”80”,”math”:”90”}}

{“name”:”李四”,”password”:”14” ,”score”:{“chinese”:”80”,”math”:”90”}}

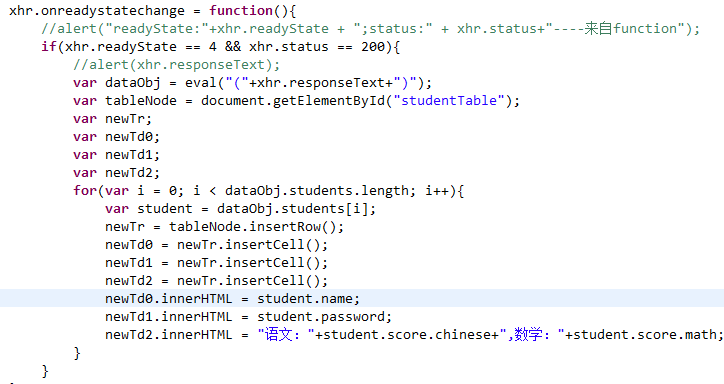
{“name”:”王五”,”password”:”45” ,”score”:{“chinese”:”80”,”math”:”90”}}

]

}

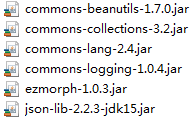
把json串换成json对象

var dataObj = eval(“(“+data+”)”);//转为json对象



* + 1. json第三方jar包的引入

json-lib



* + 1. ajax和json交互简单实例
       1. ajax验证用户名
       2. ajax省市二级联动

1. 黑马程序员-AJAX视频教程
   1. 走进ajax

|  |
| --- |
|  |

* 1. ajax用户名校验实例

|  |
| --- |
| Java  **package** cn.huangwei.ajax.servlet;  **import** javax.servlet.ServletException; **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.net.URLDecoder;   **public class** AJAXServer **extends** HttpServlet {  **protected void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  doGet(request, response);  }   **protected void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  **try**{ *// request.setCharacterEncoding("UTF-8"); // response.setContentType("text/html;charset=gb18030");* response.setContentType(**"text/html;charset=utf-8"**);  PrintWriter out = response.getWriter();   Integer inte = (Integer) request.getSession().getAttribute(**"total"**);  **int** temp = 0;  **if** (inte == **null**) {  temp = 1;  } **else** {  temp = inte.intValue() + 1;  }  request.getSession().setAttribute(**"total"**,temp);   *//1.取参数* String old = request.getParameter(**"name"**);  *//String name = new String(old.getBytes("iso8859-1"),"UTF-8");* String name = URLDecoder.*decode*(old,**"UTF-8"**);  *//2.检查参数是否有问题* **if**(old == **null** || old.length() == 0){  out.println(**"用户名不能为空"**);  } **else**{ *// String name = URLDecoder.decode(old,"UTF-8"); // byte[] by = old.getBytes("ISO8859-1"); // String name = new String(by,"utf-8"); // String name = URLDecoder.decode(old,"utf-8");  //3.校验操作 // String name = old;* **if**(name.equals(**"wangxingkui"**)){  *//4。和传统应用不同之处。这一步需要将用户感兴趣的数据返回给页面段，而不是将一个新的页面发送给用户  //写法没有变化，本质发生了改变* out.println(**"用户名["** + name + **"]已经存在，请使用其他用户名, "** + temp);  } **else**{  out.println(**"用户名["** + name + **"]尚未存在，可以使用该用户名注册, "** + temp);  }  }  } **catch**(Exception e){  e.printStackTrace();  }   } }  html  <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>用户名校验</**title**>  <**script type="text/javascript" src="jslib/jquery-3.2.1.js"** ></**script**>  <**script type="text/javascript" src="jslib/verify.js"** ></**script**> </**head**> <**body**>  *<!--基于一些好习惯，首先标签名字一定要小写，其次标签必须关闭，第三，属性名必须小写，第四，属性名必须写在引号里面-->* 请输入用户名<**br** />  *<!--ajax方式下不需要使用表单来进行数据提交，因此不用写表单标签-->  <!--ajax方式不需要name属性，需要一个id属性-->* <**input type="text" id="username"**/>  <**input type="button" value="校验" onclick="***verify*()**"**/>  *<!--这个div用于存放服务器端返回的信息，开始为空-->  <!--id属性定义是为了利用dom方式找到某一个节点，进行操作-->* <**div id="result"**></**div**>  *<!--<div>sdfdsf</div>  <div>sdfdsf</div>  <span>sdfafadaffadsafdsafsdf</span>  <span>sdfafadaffadsafdsafsdf</span>-->  <!--div和span的直观差异，div中的内容占据一行，span的内容和前后其他内容相处良好，可以接在后面-->* </**body**> </**html**>  Js  *//定义用户名校验的方法* **function** *verify*() {  *//首先测试一下页面的按钮按下，可以调用这个方法   //使用JavaScript的alert方法，显示一个提示窗口  /\*alert("被点击了");\*/   //①获取文本框中的内容  //document.getElementById("username");dom方式  //jquery的查找节点，返回的是jquery对象，可以继续在上面执行jquery方法* **var** jqueryObj = **$**(**"#username"**);  *//获取节点的值* **var** username = jqueryObj.val();   *//alert(username);  //②将文本框中的数据发送给服务器端的servlet  //使用jquery的XMLHttprequest对象的get请求的封装* **$**.get(**"AJAXServer?name="** + username, **null**, *callback*);   }  *//回调函数* **function** *callback*(data) {   *//③接受服务器端返回的数据  //alert(data);  //④将服务器端返回的数据显示在页面上* **$**(**"#result"**).html(data); }  简写  *//定义用户名校验的方法* **function** *verify*() {  **$**.get(**"AJAXServer?name="** +**$**(**"#username"**).val(), **null**, **function** (data) {  **$**(**"#result"**).html(data);  }); } |

* 1. ajax底层（XMLHttprequest）

|  |
| --- |
| *//定义用户名校验的方法 //这个方法将使用XMLHttprequest对象来进行AJAX的异步数据交互* **var *xmlhttp***; **function** *verify*() {  *//①使用dom的方式获取文本框中的值  //document.getElementById("username")是dom中获取元素节点的一种方法* **var** username = **document**.getElementById(**"username"**).value;   *//②创建XMLHttprequest对象  //这是XMLHttprequest对象元素使用中最复杂的一步  //需要针对IE和其他类型的浏览器建立这个对象的不同方式的代码* **if**(**window**.XMLHttpRequest){  *//针对火狐，mozillar，Opera，safari* ***xmlhttp*** = **new** XMLHttpRequest();  *//针对某些特定版本的mozillar浏览器的bug进行修正* **if**(***xmlhttp***.overrideMimeType){  ***xmlhttp***.overrideMimeType(**"text/xml"**);  }  }**else if**(**window**.ActiveXObject){  *//针对ie6以下  //两个可以用于创建XMLHttprequst对象的挂件名称，保存在一个js的数组中  //排在前面的版本较新* **var** activexName = [**"MSXML2.XMLHTTP"**, **"Microsoft.XMLHTTP"**];  **for**(**var** i = 0; i < activexName.**length**; i++){  **try**{  *//取出一个控件名进行创建，如果创建成功就终止循环  //如果创建失败，会排出异常吗，然后可以继续循环，继续尝试创建* ***xmlhttp*** = **new** ActiveXObject(activexName[i]);  **break**;  }**catch** (e){   }  }   }   *//确认XMLHTTPrequest对象创建成功* **if**(!***xmlhttp***){  alert(**"XMLHTTPrequest对象创建失败"**);  **return**;  }**else**{  alert(***xmlhttp***);  }   *//③注册回调函数  //注册回调函数时，只需要函数名，不用加括号  //我们需要将回调函数名注册，如果加上括号，就会把函数的返回值注册上，这是错误的* ***xmlhttp***.**onreadystatechange** = *callback*;   *//④设置连接信息  //第一个参数表示http的请求方式，支持所有http的请求方式，主要使用get和post  //第二个参数表示请求的url地址，get方式请求的参数也在url中  //第三个参数表示采用异步还是同步方式交互，true表示异步* ***xmlhttp***.open(**"get"**, **"AJAXServer?name="** +username, **true**);   *//⑤发送数据，开始和服务器进行交互  //同步方式下，send这句话会在服务器数据回来后执行  //异步方式下，send这句话会立即执行* ***xmlhttp***.send(**null**); }  *//回调函数* **function** *callback*(){  *//⑥接受响应的数据  //判断对象的状态是否是交互完成* **if**(***xmlhttp***.**readyState** == 4){  *//判断http的交互是否成功* **if**(***xmlhttp***.**status** == 200){  *//获取服务端返回的数据  //获取服务端输出的纯文本数据* **var** responseText = ***xmlhttp***.**responseText**;  *//将数据显示在页面上  //通过dom方式找到div标签对应的元素节点* **var** divNode = **document**.getElementById(**"result"**);  *//设置元素节点的html内容* divNode.**innerHTML** = responseText;  }  } } |

* 1. ajax技术核心

|  |
| --- |
|  |

* 1. ajaxXML数据交互

|  |
| --- |
| *//定义用户名校验的方法 //这个方法将使用XMLHttprequest对象来进行AJAX的异步数据交互* **var *xmlhttp***; **function** *verify*() {  *//①使用dom的方式获取文本框中的值  //document.getElementById("username")是dom中获取元素节点的一种方法* **var** username = **document**.getElementById(**"username"**).value;   *//②创建XMLHttprequest对象  //这是XMLHttprequest对象元素使用中最复杂的一步  //需要针对IE和其他类型的浏览器建立这个对象的不同方式的代码* **if**(**window**.XMLHttpRequest){  *//针对火狐，mozillar，Opera，safari* ***xmlhttp*** = **new** XMLHttpRequest();  *//针对某些特定版本的mozillar浏览器的bug进行修正* **if**(***xmlhttp***.overrideMimeType){  ***xmlhttp***.overrideMimeType(**"text/xml"**);  }  }**else if**(**window**.ActiveXObject){  *//针对ie6以下  //两个可以用于创建XMLHttprequst对象的挂件名称，保存在一个js的数组中  //排在前面的版本较新* **var** activexName = [**"MSXML2.XMLHTTP"**, **"Microsoft.XMLHTTP"**];  **for**(**var** i = 0; i < activexName.**length**; i++){  **try**{  *//取出一个控件名进行创建，如果创建成功就终止循环  //如果创建失败，会排出异常吗，然后可以继续循环，继续尝试创建* ***xmlhttp*** = **new** ActiveXObject(activexName[i]);  **break**;  }**catch** (e){   }  }   }   *//确认XMLHTTPrequest对象创建成功* **if**(!***xmlhttp***){  alert(**"XMLHTTPrequest对象创建失败"**);  **return**;  }**else**{  alert(***xmlhttp***);  }   *//③注册回调函数  //注册回调函数时，只需要函数名，不用加括号  //我们需要将回调函数名注册，如果加上括号，就会把函数的返回值注册上，这是错误的* ***xmlhttp***.**onreadystatechange** = *callback*;   *//④设置连接信息  //第一个参数表示http的请求方式，支持所有http的请求方式，主要使用get和post  //第二个参数表示请求的url地址，get方式请求的参数也在url中  //第三个参数表示采用异步还是同步方式交互，true表示异步  //xmlhttp.open("get", "AJAXServer?name=" +username, true);   //POST方式请求的代码* ***xmlhttp***.open(**"POST"**,**"AJAXXMLServer"**,**true**);  *//POST方式需要设置自己的请求头* ***xmlhttp***.setRequestHeader(**"Content-Type"**,**"application/x-www-form-urlencoded"**);  *//POST方式发送数据* ***xmlhttp***.send(**"name="** + username);   *//⑤发送数据，开始和服务器进行交互  //同步方式下，send这句话会在服务器数据回来后执行  //异步方式下，send这句话会立即执行  //xmlhttp.send(null);* }  *//回调函数* **function** *callback*(){  *//⑥接受响应的数据  //判断对象的状态是否是交互完成* **if**(***xmlhttp***.**readyState** == 4){  *//判断http的交互是否成功* **if**(***xmlhttp***.**status** == 200){  *//使用responseXML方式来接受xml数据对象的dom对象* **var** domObj = ***xmlhttp***.**responseXML**;  *//<message>1234</message>  //dom中利用getElementByTagName可以根据标签名来获取元素节点，返回的是一个数组* **var** messageNodes = domObj.getElementsByTagName(**"message"**);  **if**(messageNodes.**length** > 0){  *//获取message节点的文本内容  //message标签中的文本在dom中是message标签对应的元素节点的子节点，firstChild可以获取当前节点的第一个子节点  //通过以下方式就可以获取到文本内容所对硬的的节点* **var** textNode = messageNodes[0].**firstChild**;  *//对于文本节点来说，可以通过nodeValue的方式返回文本节点的文本内容* **var** responseMessage = textNode.**nodeValue**;   *//将数据显示在页面上  //通过dom的方式找到div标签所对应的元素节点* **var** divNode = **document**.getElementById(**"result"**);  *//设置元素节点中的html内容* divNode.**innerHTML** = responseMessage;  }**else**{  alert(**"xml数据格式错误，原始文件内容为："** + ***xmlhttp***.**responseText**);  }  }**else**{  alert(**"出错了"**);  }  } } |

* 1. 解决ajax中文乱码和跨域问题

|  |
| --- |
| 这两个配套使用  这种方式前台要做一次encode    跨域访问问题 |

* 1. dom模型和domAPI

|  |
| --- |
|  |

* 1. Jquery练习

|  |
| --- |
| 弹出窗口  Html  <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>jquery实例1：浮动窗口</**title**>  <**link type="text/css" rel="stylesheet" href="css/win.css"** />  <**script type="text/javascript" src="jslib/jquery-3.2.1.js"**></**script**>  <**script type="text/javascript" src="jslib/jquerywin.js"**></**script**> </**head**> <**body**>  <**a onclick="***showWin*()**" href="javascript:void(0)"**>显示浮动窗口</**a**>  *<!--如何表示页面中的一个弹出窗口?可以用div表示-->  <!--目前怎么看都不是一个窗口，因此需要用css来改变一下这个div的样子-->* <**div id="win"**>  <**div id="title"**> 我是一个标题栏<**span id="close" onclick="***hideWin*()**"**>X</**span**></**div**>  <**div id="content"**> 我是内容</**div**>  </**div**> </**body**> </**html**>  Js文件  *//显示浮动窗口的方法* **function** *showWin*() {  */\*alert("显示弹出窗口");\*/  //①找到对应的窗口让他显示出来* **var** winNode = **$**(**"#win"**);  *//②让div对应的窗口显示出来  //方法①修改节点的css的display的值，让窗口显示出来  // winNode.css("display","inline");  //方法②show()方法  //winNode.show("slow");  //方法③fadeIn()方法* winNode.**fadeIn**(**"slow"**); } */\*关闭窗口\*/* **function** *hideWin*() {  **$**(**"#win"**).**fadeOut**(**"slow"**); }  菜单展开闭合  Html  <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>实例2：弹出菜单</**title**>  <**link type="text/css" rel="stylesheet" href="css/menu.css"** />  <**script type="text/javascript" src="jslib/jquery-3.2.1.js"**></**script**>  <**script type="text/javascript" src="jslib/jqueryMenu.js"**></**script**> </**head**> <**body**>  <**ul**>  <**a href="#"**>我是菜单1</**a**>  <**li**>我是子菜单1</**li**>  <**li**>我是子菜单2</**li**>  </**ul**>  <**ul**>  <**a href="#"**>我是菜单2</**a**>  <**li**>我是子菜单3</**li**>  <**li**>我是子菜单4</**li**>  </**ul**>  <**ul**>  <**a href="#"**>我是菜单3</**a**>  <**li**>我是子菜单3</**li**>  <**li**>我是子菜单4</**li**>  </**ul**>  <**ul**>  <**a href="#"**>我是菜单4</**a**>  <**li**>我是子菜单3</**li**>  <**li**>我是子菜单4</**li**>  </**ul**> </**body**> </**html**>  Js  */\*需要点击主菜单的按钮，对应的子菜单可以显示，再次点击就隐藏\*/  /\*需要编写代码，在页面装载时，给所有的主菜单添加onclick事件 \* 保证主菜单点击时可以显示或隐藏子菜单 \* \*/  /\*注册页面装载是执行的方法\*/* **$**(**function** () {  **$**(**"ul a"**).click(**function**(){  */\*菜单展开闭合\*/  //$(this).nextAll("li").toggle("slow");  /\*展开闭合一个，关闭其他\*/* **$**(**this**).nextAll(**"li"**).toggle(**"slow"**);  **$**(**this**).parent().siblings(**"ul"**).children(**"a"**).nextAll(**"li"**).hide(**"slow"**);  }); });  可编辑的表格  Html  <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>可编辑的表格</**title**>  <**link type="text/css" rel="stylesheet" href="css/editTable.css"** />  <**script type="text/javascript" src="jslib/jquery-3.2.1.js"**></**script**>  <**script type="text/javascript" src="jslib/jqueryEditTable.js"**></**script**> </**head**> <**body**>  <**table**>  <**tbody**>  <**tr**>  <**td**>123123</**td**>  <**td**>456456</**td**>  </**tr**>  </**tbody**>  </**table**> </**body**> </**html**>  Js  *//在页面装载时，让所有的td都有一个onclick事件* **$**(**document**).ready(**function**(){  *//找到所有td的节点* **$**(**"td"**).click(*tdClick*); });  **function** *tdClick*(){  **var** tdNode = **$**(**this**);  *//①取出当前td中的文本内容保存起来* **var** text = tdNode.html();  *//②清空ttd里面的内容* tdNode.html(**""**);*//或者tdNode.empyt();  //③建立一个文本框，也就是input* **var** input = **$**(**"<input type='text'/>"**)  *//④设置文本框的值是保存起来的文本内容* input.val(text);  *//4.5让文本框可以响应键盘按下并弹起的事件，主要用处理回车确认* input.keyup(*inputToTdOnEnter*);  input.**blur**(*inputToTdOnBulr*);  *//⑤将文本框加入到td中* tdNode.append(input);  *//5.5让文本框的文字被高亮选中，需要将jquery对象转换成js对象  /\*var inputDom = input.get(0);  inputDom.select();\*/* input.select();*//jquery也可以使用  //⑥需要清除td上的点击事件* tdNode.unbind(**"click"**); } *//点回车是转换input和td* **function** *inputToTdOnEnter*(event){  *//0.获取当前用户按下的键值  //解决不同浏览器获取事件对象的差异* **var** myEvent = event || **window**.**event**;  **var** keyCode = myEvent.**keyCode**;  *//1.判断是否是回车按下* **if**(keyCode == 13){*//回车对应的键码表是13  //2.获取当前文本框的内容* **var** inputNode = **$**(**this**);  *//3.清空td里面的内容* **var** tdNode = inputNode.parent();  *//4.将保存的文本框的内容填充到td中* tdNode.**html**(inputNode.val());  *//5.让td重新拥有点击事件* tdNode.click(*tdClick*);  }  }  *//鼠标焦点变话转换input和td* **function** *inputToTdOnBulr*(event){  *//0.获取当前用户按下的键值  //解决不同浏览器获取事件对象的差异* **var** myEvent = event || **window**.**event**;  **var** keyCode = myEvent.**keyCode**;  *//2.获取当前文本框的内容* **var** inputNode = **$**(**this**);  *//3.清空td里面的内容* **var** tdNode = inputNode.parent();  *//4.将保存的文本框的内容填充到td中* tdNode.**html**(inputNode.val());  *//5.让td重新拥有点击事件* tdNode.click(*tdClick*); } |

* 1. 后台模拟股票涨跌功能

|  |
| --- |
| GetStocksInfo.java  **package** servlet;  **import** bean.Stock;  **import** javax.servlet.ServletConfig; **import** javax.servlet.ServletException; **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.util.HashMap;   **public class** GetStocksInfo **extends** HttpServlet{  **private** HashMap<String, Stock> **stocks**;  **protected void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  *//返回两只股票的价格信息* **double** sz = Math.*random*() \* 20;  **double** pf = Math.*random*() \* 0.5;   *//通过随机数是奇数还是偶数来判断股票上涨还是下跌* **boolean** flagsz = ((**int**)Math.*random*() \* 10) % 2 == 0;  **boolean** flagpf = ((**int**)Math.*random*() \* 10) % 2 == 0;   *//2.将随机数和股票的当前价进行加或减的操作，得到新的当前价格* Stock szzs = **stocks**.get(**"30001"**);  Stock pfyh = **stocks**.get(**"00001"**);  **double** temp;  **if**(flagsz){  temp = szzs.getNow() + sz;   }**else**{  temp = szzs.getNow() - sz;  }  *//需要对新的当前价格进行截取，只保留小数点后两位* temp = (**int**) (temp \* 100) / 100.0;  szzs.setNow(temp);   **if**(flagpf){  temp = pfyh.getNow() + pf;   }**else**{  temp = pfyh.getNow() - pf;  }   temp = (**int**) (temp \* 100) / 100.0;  pfyh.setNow(temp);   response.setContentType(**"text/html;charset=gb2312"**);  PrintWriter out = response.getWriter();  out.println(szzs + **"<br />"** + pfyh);  *//3.返回两只股票的昨天收盘，今天开盘和当前价格* }   **protected void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  doGet(request,response);  }   @Override  **public void** init(ServletConfig config) **throws** ServletException {  **stocks** = **new** HashMap<String, Stock>();  *//创建股票* Stock szzs = **new** Stock(30000,2900,**"上证指数"**,**"30001"**);  Stock pufa = **new** Stock(30000,2900,**"浦发英航"**,**"00001"**);   *//将两只股票保存在stocks的map中* **stocks**.put(szzs.getId(),szzs);  **stocks**.put(pufa.getId(),pufa);  System.***out***.println(**stocks**);  } }  Stock.java  **package** bean;  */\*\*  \* Created by sshss on 2017/6/19.  \*/* **public class** Stock {  */\*昨天的收盘价\*/* **private double yesterday**;   */\*今天的开盘价\*/* **private double today**;   */\*当前价\*/* **private double now**;   */\*股票名称\*/* **private** String **name**;   */\*股票代码\*/* **private** String **id**;   **public** Stock(**double** yesterday, **double** today, String name, String id) {  **this**.**yesterday** = yesterday;  **this**.**today** = today;  **this**.**name** = name;  **this**.**id** = id;  **this**.**now** = today;  }   **public double** getYesterday() {  **return yesterday**;  }   **public void** setYesterday(**double** yesterday) {  **this**.**yesterday** = yesterday;  }   **public double** getToday() {  **return today**;  }   **public void** setToday(**double** today) {  **this**.**today** = today;  }   **public double** getNow() {  **return now**;  }   **public void** setNow(**double** now) {  **this**.**now** = now;  }   **public** String getName() {  **return name**;  }   **public void** setName(String name) {  **this**.**name** = name;  }   **public** String getId() {  **return id**;  }   **public void** setId(String id) {  **this**.**id** = id;  }   @Override  **public** String toString() {  **return "Stock{"** +  **"yesterday="** + **yesterday** +  **", today="** + **today** +  **", now="** + **now** +  **", name='"** + **name** + **'\''** +  **", id='"** + **id** + **'\''** +  **'}'**;  } }  {"30001":{name:"上证指数",yes:30000.0,tod:2900.0,now:2910.39},"00001":{name:"浦发英航",yes:30000.0,tod:2900.0,now:2900.36}}  {"30001":{name:"上证指数",yes:30000.0,tod:2900.0,now:2910.39},"00001":{name:"浦发英航",yes:30000.0,tod:2900.0,now:2900.36}} |

* 1. 动态股票

|  |
| --- |
| Jquerystock.js  */\*期望进入页面后就可以开始从服务器端获取数据，然后在页面显示价格\*/* **var *obj***; **var *sid***; **$**(**document**).ready(**function**(){  *getInfo*();  **var** stockNode = **$**(**"#stock"**);  stockNode.css(**"border"**,**"1px solid red"**);  stockNode.css(**"width"**,**"150px"**);  stockNode.css(**"background"**,**"yellow"**);  stockNode.hide(**"slow"**);  *//页面载入时隐藏弹出框* **var** as = **$**(**"a"**);  *//定义鼠标进入的股票名称的操作* as.mouseover(**function**(e){  stockNode.show(**"slow"**);  stockNode.css(**"position"**,**"absolute"**).css(**"left"**,e.**pageX**).css(**"top"**,e.**pageY**);  ***sid*** = **$**(**this**).parent().attr(**"id"**);  *update*();  });  *//定义鼠标移除的股票名称的操作* as.mouseout(**function**(){  stockNode.hide(**"slow"**);  });  *//③每隔一秒钟和服务器交互一次，用户不用刷新页面就可以不断地看到最新的股票信息* setInterval(*getInfo*,1000);  }); *//更新当前* **function** *update*(){  **var** stockobj = ***obj***[***sid***];  **$**(**"#yes"**).children(**"span"**).html(stockobj.yes);  **$**(**"#tod"**).children(**"span"**).html(stockobj.tod);  **$**(**"#now"**).children(**"span"**).html(stockobj.**now**); }  **function** *getInfo*(){  *//①向服务器发起请求，获取数据* **$**.get(**"GetStocksInfo"**, **null**, **function** (data) {  *//②接受并解析数据* ***obj*** = eval(data);  *//2.1获取两只股票的当前指数信息* **szzs** = ***obj***[**"30001"**];  **pfyh** = ***obj***[**"00001"**];  */\*  \* 遍历一个js对象  \* for(var stockId in obj){  \* var stock = obj[stockId];  \* }  \* \*/  //2.2找到页面中对应的节点，然后填充最新的股票价格* **var** spanNode1 = **$**(**"#30001"**).children(**"span"**);  **var** spanNode2 = **$**(**"#00001"**).children(**"span"**);  spanNode1.html(**szzs**.**now**);  **if**(**szzs**.**now** > **szzs**.yes){  *//当前价格大于昨天收盘，则显示红色* spanNode1.css(**"color"**,**"red"**);  }**else**{  spanNode1.css(**"color"**,**"green"**);  }  spanNode2.html(**pfyh**.**now**);  **if**(**pfyh**.**now** > **pfyh**.yes){  *//当前价格大于昨天收盘，则显示红色* spanNode2.css(**"color"**,**"red"**);  }**else**{  spanNode2.css(**"color"**,**"green"**);  }  *update*();  }); }  jqueryStock.html  <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>动态股票信息</**title**>  <**script type="text/javascript" src="jslib/jquery-3.2.1.js"**></**script**>  <**script type="text/javascript" src="jslib/jqueryStock.js"**></**script**> </**head**> <**body**>  <**div id="30001"**><**a href="#"**>上证指数：</**a**><**span**></**span**></**div**>  <**div id="00001"**><**a href="#"**>浦发英航:</**a**><**span**></**span**></**div**>  <**div id="stock"**>  <**div id="yes"**>昨收：<**span**></**span**></**div**>  <**div id="tod"**>今开：<**span**></**span**></**div**>  <**div id="now"**>当前：<**span**></**span**></**div**>  </**div**> </**body**> </**html**> |

* 1. 仿照google进行补全

|  |
| --- |
| AutoComplete.java  **package** servlet;  **import** org.w3c.dom.Document; **import** org.w3c.dom.Element; **import** org.w3c.dom.Node; **import** org.w3c.dom.NodeList;  **import** javax.servlet.ServletException; **import** javax.servlet.http.HttpServlet; **import** javax.servlet.http.HttpServletRequest; **import** javax.servlet.http.HttpServletResponse; **import** javax.xml.parsers.DocumentBuilder; **import** javax.xml.parsers.DocumentBuilderFactory; **import** java.io.File; **import** java.io.IOException; **import** java.util.ArrayList; **import** java.util.List;  */\*\*  \* Created by sshss on 2017/6/23.  \* 接收用户端请求  \*/* **public class** AutoComplete **extends** HttpServlet {  **protected void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  doGet(request,response);  }   **protected void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  *//表示页面传过来的字符串，用于和服务端的单词进型完整匹配* String word = request.getParameter(**"word"**);  *//将字符串保存在request对象中* request.setAttribute(**"word"**, word);  */\*将请求转发给视图层，（注意ajax中，这个所谓的视图层不返回页面，只返回数据，所以  \*也可以称作是一个数据层  \* \*/* String path = **this**.getServletConfig().getServletContext().getRealPath(**"/"**);  Element rootElement = getRootElement(path);  List<String> wordList = **new** ArrayList<String>();  NodeList childNodes = rootElement.getChildNodes();  **for**(**int** i = 0; i < childNodes.getLength(); i++){  **if**(childNodes.item(i).getNodeName().equals(**"word"**)){  NodeList nodeList = childNodes.item(i).getChildNodes();  Node node = nodeList.item(0);  **if**(node.getNodeValue().startsWith(word)){  wordList.add(node.getNodeValue());  }   }  }  request.setAttribute(**"wordList"**, wordList);  request.getRequestDispatcher(**"wordxml.jsp"**).forward(request,response);  }  **private** Element getRootElement(String path){  DocumentBuilderFactory builderFactory = DocumentBuilderFactory  .*newInstance*();  DocumentBuilder builder = **null**;  **try** {  builder = builderFactory.newDocumentBuilder();  */\*  \* builder.parse()方法将给定文件的内容解析为一个 XML 文档， 并且返回一个新的 DOM Document对象。  \*/* Document document = builder.parse(**new** File(path + **"word.xml"**));  *//获取文档的根元素，赋值给rootElement变量* Element rootElement = document.getDocumentElement();  **return** rootElement;  } **catch** (Exception e) {  e.printStackTrace();  }  **return null**;  } }  word.xml  *<?***xml version="1.0" encoding="utf-8"** *?>* <**words**>  <**word**>absolute</**word**>  <**word**>anyone</**word**>  <**word**>anything</**word**>  <**word**>apache</**word**>  <**word**>abandon</**word**>  <**word**>breach</**word**>  <**word**>boolean</**word**> </**words**>  Wordxml.jsp  *<%--ajax自动补全实例--%> <%--与传统应用的视图层不同，这个jsp返回的是xml的数据，因此contentType的值是text/xml--%>* <%@ **taglib uri**="**http://java.sun.com/jsp/jstl/core**" **prefix**="**c**"%> <%@ **page contentType**="**text/html; charset=UTF-8**" **language**="**java**" %> *<%--返回xml数据的‘视图层暂时不做任何逻辑判断，现将所有单词全部返回，待前后台应用完整 协作的时候，在限制返回的内容’--%>* <**words**>  <**c:forEach var="word" items="${**wordList **}"**>  <**word**>**${**word**}**</**word**>  </**c:forEach**> </**words**>  JQueryAutoComplete.html  <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>jquery实例:仿google的自动补全</**title**>  <**script type="text/javascript" src="jslib/jquery-3.2.1.js"**></**script**>  <**script type="text/javascript" src="jslib/jqueryAutoComplete.js"**></**script**> </**head**> <**body**> 自动补全实例  <**input type="text" id="word"**/>  <**input type="button" value="提交"**/>  <**div id="auto"**></**div**> </**body**> </**html**>  **jqueryAutoComplete.js**  */\*自动补全\*/ /\*表示当前高亮的节点\*/* **var *highLightIndex*** = -1; **var *timeoutId***; **$**(**function**(){  **var** wordInput = **$**(**"#word"**);  *//offset() 方法返回或设置匹配元素相对于文档的偏移（位置）* **var** wordInputOffset = wordInput.offset();  *//自动补全的框最开始应该藏起来,解决跟着文本框的位置关系* **$**(**"#auto"**).hide().css(**"border"**,**"1px black solid"**).css(**"position"**,**"absolute"**)  .css(**"top"**,wordInputOffset.**top** + wordInput.height() + 5 + **"px"**)  .css(**"left"**,wordInputOffset.**left** + **"px"**).css(**"width"**,wordInput.width() + 4);   *//给文本框添加键盘按下并弹出的事件* wordInput.keyup(**function**(event){  *//处理文本框的键盘事件* **var** myEvent = event || **window**.**event**;  **var** keyCode = myEvent.**keyCode**;  *//如果输入的是字母，应该将文本框中的最新信息发送给服务器  //或者是delete和backspace* **if**(keyCode >=65 && keyCode <= 90 || keyCode == 8 || keyCode == 46) {  *//1.首先获取文本框中的内容* **var** wordText = wordInput.val();  **if**(wordText != **""**) {  *//对上一次未完成的延时进行取消* clearTimeout(***timeoutId***);  *//考虑到用户打字比较慢，设定延迟* ***timeoutId*** = setTimeout(**function** () {  *//2.将文本框中的内容发送给服务器端* **$**.post(**"AutoComplete"**, {**word**: wordText}, **function** (data) {  *//将dom对象data转换成jquery对象* **var** jqueryObj = **$**(data);  *//找到所有的word节点* **var** wordNodes = jqueryObj.find(**"word"**);  *//需要清空原来的内容* **$**(**"#auto"**).html(**""**);  **if** (wordNodes.length > 0) {  *//遍历多有word节点中，取出单词内容，然后将单词内容添加到弹出框* wordNodes.each(**function** (index) {  *//获取单词内容* **var** wordNode = **$**(**this**);   *//新建div节点，将单词内容加入到新建的节点中* **var** wordText = wordNode.html();  **var** divNode = **$**(**"<div>"**).attr(**"id"**,index).html(wordText);  *//为div节点的鼠标进入添加高亮事件  //进入事件* divNode.mouseover(**function**(){  *//原来就有高亮的，就取消高亮* **if**(***highLightIndex*** != -1){  **$**(**"#auto"**).children(**"div"**).**eq**(***highLightIndex***)  .css(**"background-color"**, **"white"**);  }  ***highLightIndex*** = **$**(**this**).attr(**"id"**);  **$**(**this**).css(**"background-color"**, **"red"**);  }).mouseout(**function**(){  *//原来就有高亮的，就取消高亮* **if**(***highLightIndex*** != -1){  **$**(**"#auto"**).children(**"div"**).**eq**(***highLightIndex***)  .css(**"background-color"**, **"white"**);  }  *//移除事件，高亮取消* **$**(**this**).css(**"background-color"**, **"white"**);  ***highLightIndex*** = -1;  }).click(**function**(){  *//高亮则填充文本框* **var** highLightText = **$**(**"#auto"**).children(**"div"**).**eq**(***highLightIndex***).html();  **$**(**"#auto"**).hide(**"formal"**);  ***highLightIndex*** = -1;  wordInput.val(highLightText);  });  *//将新建的节点加入到弹出框的div中* **$**(**"#auto"**).append(divNode);  });  **$**(**"#auto"**).show(**"formal"**);  } **else** {  **$**(**"#auto"**).hide(**"formal"**);  }  }, **"xml"**);  },500);   }**else**{  **$**(**"#auto"**).hide(**"formal"**);  *//弹出框隐藏的同时，高亮节点重置* ***highLightIndex*** = -1;  }  }**else if**(keyCode == 38 || keyCode == 40){  *//针对上下键，进行高亮的处理  /\*向上\*/  //先取到自动补全的div的所有子节点* **var** autoNodeChildren = **$**(**"#auto"**).children(**"div"**);  **if**(keyCode == 38){  **if**(***highLightIndex*** != -1){  *//取到索引值为highLightIndex的div节点,设定背景色为白色* autoNodeChildren.**eq**(***highLightIndex***).css(**"background-color"**, **"white"**);  ***highLightIndex***--;  }**else**{  *//处理第一次highLightIndex是-1的问题* ***highLightIndex*** = autoNodeChildren.**length** - 1;  }   **if**(***highLightIndex*** == -1){  *//如果修改索引值以后变成-1，则将索引值指向最后一个元素节点* ***highLightIndex*** = autoNodeChildren.**length** - 1;  }  *//让现在索引值的节点变成红色* autoNodeChildren.**eq**(***highLightIndex***).css(**"background-color"**, **"red"**);  }  */\*向下\*/* **if**(keyCode == 40){  **if**(***highLightIndex*** != -1){  *//取到索引值为highLightIndex的div节点,设定背景色为白色* autoNodeChildren.**eq**(***highLightIndex***).css(**"background-color"**, **"white"**);  }  ***highLightIndex***++;  **if**(***highLightIndex*** == autoNodeChildren.**length**){  *//如果修改索引值以后变成z最后的值，则将索引值指向第一个元素节点* ***highLightIndex*** = 0;  }  *//让现在索引值的节点变成红色* autoNodeChildren.**eq**(***highLightIndex***).css(**"background-color"**, **"red"**);  }   }**else if**(keyCode == 13){  *//针对回车进行判断* **if**(***highLightIndex*** != -1){  *//高亮则填充文本框 var highLightText = $("#auto").children("div").eq(highLightIndex).html();* **var** highLightText = **$**(**this**).html();  **$**(**"#auto"**).hide(**"formal"**);  ***highLightIndex*** = -1;  wordInput.val(highLightText);  }**else** {  *//否则就是提交  submitInput*();  **$**(**"#auto"**).hide(**"formal"**);  *//让文本框失去焦点* wordInput.get(0).blur();  }  }  });    *//给按钮添加事件，表示文本框中的数据被提交* **$**(**"input[type=button]"**).click(**function**() {  *submitInput*();  }); });  *//提交事件* **function** *submitInput*() {  alert(**"文本框中的["**+ **$**(**"#word"**).val() + **"]被提交了"**); } |