# Day03 AJAX

## AJAX概述

### 什么是AJAX

AJAX（Asynchronous-异步 Javascript And XML）翻译成中文就是“异步Javascript和XML”。即使用Javascript语言与服务器进行异步交互，传输的数据为XML（当然，传输的数据不只是XML，现在比较流行的数据格式JSON）。

Ajax本质还是客户端向服务器端发送请求，并接收服务器响应的数据，解析后显示在浏览器

AJAX还有一个最大的特点就是，当服务器响应时，不用刷新整个浏览器页面，而是可以局部刷新。这一特点给用户的感受是在不知不觉中完成请求和响应过程。

AJAX实现浏览器（即客户端）向服务器发送异步请求

在没有使用AJAX之前，浏览器向服务器发送同步请求

Ajax技术主要包括：

Javascript:dom操作

Xml:解析xml

Ajax优势：用来提升用户体验度

目的：提高用户的体验度

* 与服务器异步交互；
* 浏览器页面局部刷新；

### 2.　同步交互与异步交互

* 同步交互：客户端发出一个请求后，需要等待服务器响应结束后，才能发出第二个请求；
* 异步交互：客户端发出一个请求后，无需等待服务器响应结束，就可以发出第二个请求。

google地图：AJAX

### 3.　AJAX常见应用情景 异步请求方式



当输入用户名后，把光标移动到其他表单项上时，浏览器会使用AJAX技术向服务器发出请求，服务器会查询名为zhangSan的用户是否存在，最终服务器返回true表示名为zhangSan的用户已经存在了，浏览器在得到结果后显示“用户名已被注册！”。

* 整个过程中页面没有刷新，只是局部刷新了；
* 在请求发出后，浏览器不用等待服务器响应结果就可以进行其他操作；

### 4　AJAX的优缺点

优点：

* AJAX使用Javascript技术向服务器发送异步请求；
* AJAX无须刷新整个页面；
* 因为服务器响应内容不再是整个页面，而是页面中的局部，所以AJAX性能高；

缺点：

* AJAX并不适合所有场景，很多时候还是要使用同步交互；
* AJAX虽然提高了用户体验，但无形中向服务器发送的请求次数增多了，导致服务器压力增大；
* 因为AJAX是在浏览器中使用Javascript技术完成的，所以还需要处理浏览器兼容性问题；
  + JQuery

## AJAX技术

1. 原生态的ajax，使用JavaScript实现ajax
   1. JavaScript提供一个XmlHttpRequest对象
2. Jquery封装的ajax实现ajax异步请求

### AJAX第一例

#### 1.1 准备工作

因为AJAX也需要请求服务器(tomcat)，异步请求也是请求服务器，所以我们需要先写好服务器端代码，即编写一个Servlet！

这里，Servlet很简单，只需要输出“Hello AJAX!”。

|  |
| --- |
| **public** **class** AServlet **extends** HttpServlet {  **public** **void** doGet(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  System.*out*.println("Hello AJAX!");  response.getWriter().print("Hello AJAX!");  }  } |

#### 1.2　AJAX核心（XMLHttpRequest）

其实AJAX就是在Javascript中多添加了一个对象：XMLHttpRequest对象。所有的异步交互都是使用XMLHttpRequest对象完成的。也就是说，我们只需要学习一个Javascript的新对象即可。

注意，各个浏览器对XMLHttpRequest的支持也是不同的！大多数浏览器都支持DOM2规范，都可以使用：**var xmlHttp = new XMLHttpRequest()**来创建对象；但IE有所不同，IE5.5以及更早版本需要：**var xmlHttp = new ActiveXObject(“Microsoft.XMLHTTP”)**来创建对象；而IE6中需要：**var xmlHttp = new ActiveXObject(“Msxml2.XMLHTTP”)**来创建对象；而IE7以及更新版本也支持DOM2规范。

简单了解：

google/火狐：支持DOM2

IE5.5及以前：**ActiveXObject(“Microsoft.XMLHTTP”);**

**IE6.0:ActiveXObject(“Msxml2.XMLHTTP”);**

**Ie7.0也支持DOM2： new XMLHttpRequest()**

为了处理浏览器兼容问题，给出下面方法来创建XMLHttpRequest对象：

|  |
| --- |
| **function** createXMLHttpRequest() {  **var** xmlHttp;  // 适用于大多数浏览器，以及IE7和IE更高版本  **try**{  xmlHttp = **new** XMLHttpRequest();  } **catch** (e) {  // 适用于IE6  **try** {  xmlHttp = **new** ActiveXObject("Msxml2.XMLHTTP");  } **catch** (e) {  // 适用于IE5.5，以及IE更早版本  **try**{  xmlHttp = **new** ActiveXObject("Microsoft.XMLHTTP");  } **catch** (e){}  }  }  **return** xmlHttp;  } |

#### 1.3　打开与服务器的连接（open方法）

当得到XMLHttpRequest对象后，就可以调用该对象的open()方法打开与服务器的连接了。open()方法的参数如下：

open(method, url, async)：

* method：请求方式，通常为GET或POST；
* url：请求的服务器地址，例如：/ajaxdemo1/AServlet，若为GET请求，还可以在URL后追加参数；
* async：这个参数可以不给，默认值为true，表示异步请求；

|  |
| --- |
| **var** xmlHttp = createXMLHttpRequest();  xmlHttp.open("GET", "/ajaxdemo1/AServlet", **true**); |

#### 发送请求

XmlHttpRequest.send(请求体);

GET：没有请求体

POST：有请求体

当使用open打开连接后，就可以调用XMLHttpRequest对象的send()方法发送请求了。send()方法的参数为POST请求参数，即对应HTTP协议的请求体内容，若是GET请求，需要在URL后连接参数。

注意：若没有参数，需要给出null为参数！若不给出null为参数，可能会导致FireFox浏览器不能正常发送请求！

|  |
| --- |
| xmlHttp.send(**null**); |

#### 1.5　接收服务器响应

当请求发送出去后，服务器端Servlet就开始执行了，但服务器端的响应还没有接收到。接下来我们来接收服务器的响应。

XMLHttpRequest对象有一个onreadystatechange事件，它会在XMLHttpRequest对象的状态发生变化时被调用。下面介绍一下XMLHttpRequest对象的5种状态：

* 0：初始化未完成状态，只是创建了XMLHttpRequest对象，还未调用open()方法；
* 1：请求已开始，open()方法已调用，但还没调用send()方法；
* 2：请求发送完成状态，send()方法已调用；
* 3：开始读取服务器响应；
* 4：读取服务器响应结束。
  + 服务器响应客户端：200成功响应 404 500响应没成功

onreadystatechange事件会在状态为1、2、3、4时引发。

　　下面代码会被执行四次！对应XMLHttpRequest的四种状态！

|  |
| --- |
| xmlHttp.onreadystatechange = **function**() {  alert('hello');  }; |

但通常我们只关心最后一种状态，即读取服务器响应结束时，客户端才会做出改变。我们可以通过XMLHttpRequest对象的readyState属性来得到XMLHttpRequest对象的状态。

|  |
| --- |
| xmlHttp.onreadystatechange = **function**() {  **if**(xmlHttp.readyState == 4) {  alert('hello');  }  }; |

其实我们还要关心服务器响应的状态码是否为200，其服务器响应为404，或500，那么就表示请求失败了。我们可以通过XMLHttpRequest对象的status属性得到服务器的状态码。

最后，我们还需要获取到服务器响应的内容，可以通过XMLHttpRequest对象的responseText得到服务器响应内容。

|  |
| --- |
| xmlHttp.onreadystatechange = **function**() {  **if**(xmlHttp.readyState == 4 && xmlHttp.status == 200) {  alert(xmlHttp.responseText);  }  }; |

#### 1.6　AJAX第一例小结

* 创建XMLHttpRequest对象；
* 调用open()方法打开与服务器的连接；
* 调用send()方法发送请求；
* 为XMLHttpRequest对象指定onreadystatechange事件函数，这个函数会在XMLHttpRequest的1、2、3、4，四种状态时被调用；

XMLHttpRequest对象的5种状态：

* 0：初始化未完成状态，只是创建了XMLHttpRequest对象，还未调用open()方法；
* 1：请求已开始，open()方法已调用，但还没调用send()方法；
* 2：请求发送完成状态，send()方法已调用；
* 3：开始读取服务器响应；
* 4：读取服务器响应结束。

通常我们只关心4状态。

XMLHttpRequest对象的status属性表示服务器状态码，它只有在readyState为4时才能获取到。

XMLHttpRequest对象的responseText属性表示服务器响应内容，它只有在readyState为4时才能获取到！

### 2　AJAX第二例（发送POST请求）

#### 2.1　发送POST请求注意事项

POST请求必须设置ContentType请求头的值为application/x-www.form-encoded。表单的enctype默认值就是为application/x-www.form-encoded！因为默认值就是这个，所以大家可能会忽略这个值！当设置了<form>的enctype=” application/x-www.form-encoded”时，等同与设置了Cotnent-Type请求头。

但在使用AJAX发送请求时，就没有默认值了，这需要我们自己来设置请求头：

|  |
| --- |
| xmlHttp.setRequestHeader("Content-Type", "application/x-www-form-urlencoded"); |

当没有设置Content-Type请求头为application/x-www-form-urlencoded时，Web容器会忽略请求体的内容。所以，在使用AJAX发送POST请求时，需要设置这一请求头，然后使用send()方法来设置请求体内容。

|  |
| --- |
| xmlHttp.send("b=B"); |

　　这时Servlet就可以获取到这个参数！！！

AServlet

|  |
| --- |
| **public** **void** doPost(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  System.*out*.println(request.getParameter("b"));  System.*out*.println("Hello AJAX!");  response.getWriter().print("Hello AJAX!");  } |

ajax2.jsp

|  |
| --- |
| <script type=*"text/javascript"*>  **function** createXMLHttpRequest() {  **try** {  **return new** XMLHttpRequest();//大多数浏览器  } **catch** (e) {  **try** {  **return new** ActiveXObject("Msxml2.XMLHTTP");  } **catch** (e) {  **return new** ActiveXObject("Microsoft.XMLHTTP");  }  }  }  **function** send() {  **var** xmlHttp = createXMLHttpRequest();  xmlHttp.onreadystatechange = **function**() {  **if**(xmlHttp.readyState == 4 && xmlHttp.status == 200) {  **var** div = document.getElementById("div1");  div.innerHTML = xmlHttp.responseText;  }  };  xmlHttp.open("POST", "/ajaxdemo1/AServlet?a=A", **true**);  xmlHttp.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");  xmlHttp.send("b=B");  }  </script> |
| <h1>AJAX2</h1>  <button onclick="send()">测试</button>  <div id=*"div1"*></div> |

### 3　AJAX第三例（用户名是否已被注册）

#### 3.1　功能介绍

在注册表单中，当用户填写了用户名后，把光标移开后，会自动向服务器发送异步请求。服务器返回true或false，返回true表示这个用户名已经被注册过，返回false表示没有注册过。

客户端得到服务器返回的结果后，确定是否在用户名文本框后显示“用户名已被注册”的错误信息！

#### 3.2　案例分析

课后练习：

补充需求：在数据库查询后判断用户名是否被占用

实现步骤：bookStore添加一个表User(id,userName,userPwd)

1. mysql数据库添加一个数据表
2. 查询用户名是否被占用：select count(\*) from user where username=?

* regist.jsp页面中给出注册表单；
* 在username表单字段中添加onblur事件，调用send()方法；
* send()方法获取username表单字段的内容，向服务器发送异步请求，参数为username；
* RegistServlet：获取username参数，判断是否为“admin”，如果是响应true，否则响应false；

#### 3.3　代码

regist.jsp

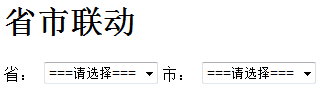
|  |
| --- |
| <script type=*"text/javascript"*>  **function** createXMLHttpRequest() {  **try** {  **return new** XMLHttpRequest();  } **catch** (e) {  **try** {  **return new** ActiveXObject("Msxml2.XMLHTTP");  } **catch** (e) {  **return new** ActiveXObject("Microsoft.XMLHTTP");  }  }  }  **function** send() {  **var** xmlHttp = createXMLHttpRequest();  xmlHttp.onreadystatechange = **function**() {  **if**(xmlHttp.readyState == 4 && xmlHttp.status == 200) {  **if**(xmlHttp.responseText == "true") {  document.getElementById("error").innerHTML = "用户名已被注册！";  } **else** {  document.getElementById("error").innerHTML = "";  }  }  };  xmlHttp.open("POST", "/ajaxdemo1/BServlet", **true**);  xmlHttp.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");  **var** username = document.getElementById("username").value;  xmlHttp.send("username=" + username);  }  </script> |
| <h1>注册</h1>  <form action=*""* method=*"post"*>  用户名：<input id=*"username"* type=*"text"* name=*"username"* onblur="send()"/><span id=*"error"*></span><br/>  密　码：<input type=*"text"* name=*"password"*/><br/>  <input type=*"submit"* value=*"注册"*/>  </form> |

RegistServlet.java

|  |
| --- |
| **public** **void** doPost(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  request.setCharacterEncoding("utf-8");  response.setContentType("text/html;charset=utf-8");    String username = request.getParameter("username");  **if**("admin".equals(username)) {  response.getWriter().print(**true**);  } **else** {  response.getWriter().print(**false**);  }  } |

### 4 AJAX第四例（省市二级联动）

#### 4.1　功能介绍



select.jsp

|  |
| --- |
| <h1>省市联动</h1>  省：  <select name=*"province"* id=*"province"*>  <option>===请选择===</option>  </select>  市：  <select name=*"city"* id=*"city"*>  <option>===请选择===</option>  </select> |

分析：

思考方式二:

1. xml格式进行数据存储，在jsp解析xml
2. 省市存储数据库，可以查询数据将数据凭借字符串传递给jsp，jsp解析字符串显示====三级联动

自定义三级联动（国家、省份、市 省份、市、区）

　　当select.jsp页面打开时，向服务器发送异步请求，得到所有省份的名称（文本数据）。然后使用每个省份名称创建<option>，添加到<select name=”province”>中。

　　并且为<select name=”province”>元素添加onchange事件监听。当选择的省份发生变化时，再向服务器发送异常请求，得到当前选中的省份下所有城市（XML数据）。然后客户端解析XML文档，使用每个城市名称创建<option>，添加到<select name=”city”>元素中。

#### 4.2　代码实现

服务器端：使用china.xml保存所有省份和城市名称：

china.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"utf-8"*?>  <china>  <province name=*"北京"*>  <city>东城区</city>  <city>西城区</city>  ……  </province>  <province name=*"天津"*>  <city>和平区</city>  <city>河东区</city>  ……  </province>  <province name=*"河北"*>  <city>石家庄</city>  <city>衡水</city>  ……  </province>  ……  </china> |

* ProvinceServlet：负责把所有省份名称响应给客户端，这需要使用dom4j解析china.xml，得到所有<province>元素的name属性值，连接成一个字符串发送给客户端；
* CityServlet：负责得到某个省份元素，然后以字符串形式发送给客户端；

ProvinceServlet.java

|  |
| --- |
| **public** **class** ProvinceServlet **extends** HttpServlet {  **public** **void** doGet(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  // 注意设置编码  response.setContentType("text/html;charset=utf-8");    // 使用DOM4J解析xml文档  InputStream input = **this**.getClass().getClassLoader().getResourceAsStream("china.xml");  SAXReader reader = **new** SAXReader();  **try** {  Document doc = reader.read(input);  // xpath查询所有province元素的name属性  List<Attribute> provinceNameAttributeList = doc.selectNodes("//province/@name");  // 用来装载所有name属性值  List<String> provinceNames = **new** ArrayList<String>();  // 遍历每个属性，获取属性名称，添加到list中  **for**(Attribute proAttr : provinceNameAttributeList) {  provinceNames.add(proAttr.getValue());  }  // System.out.println(provinceNames);  // 把list转换成字符串  String str = provinceNames.toString();  // 把字符串前后中的[]去除发送给客户端  response.getWriter().print(str.substring(1, str.length()-1));  } **catch** (DocumentException e) {  **throw** **new** RuntimeException(e);  }  }  } |

CityServlet.java

|  |
| --- |
| **public** **class** CityServlet **extends** HttpServlet {  **public** **void** doPost(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  request.setCharacterEncoding("utf-8");  // 注意，这里内容类型必须是text/xml，不然客户端得到的就不是xml文档对象，而是字符串了。  response.setContentType("text/xml;charset=utf-8");  // 获取省份参数  String provinceName = request.getParameter("provinceName");    InputStream input = **this**.getClass().getClassLoader().getResourceAsStream("china.xml");  SAXReader reader = **new** SAXReader();  **try** {  Document doc = reader.read(input);  // 查询指定省份名称的<province>元素  Element provinceElement = (Element)doc.selectSingleNode("//province[@name='" + provinceName + "']");  // System.out.println(provinceElement.asXML());  // 把元素转换成字符串发送给客户端  response.getWriter().print(provinceElement.asXML());  } **catch** (Exception e) {  **throw** **new** RuntimeException(e);  }  }  } |

客户端：

* 在打开select.jsp页面时就向服务器请求所有省份的名称，添加到<select name=”province”>元素中。
* 给<select name=”province”>元素添加onchange事件监听，内容为向服务器发送请求，得到XML文档：<province>元素，然后解析它，添加到<select name=”city”>中。

|  |
| --- |
| // 文档加载完成后  // 加载所有省份名称  window.onload = **function**() {  /\*  请求服务器，加载所有省名称到<select>中  \*/  /\*  1. ajax四步  \*/  **var** xmlHttp = createXMLHttpRequest();  xmlHttp.onreadystatechange = callback;//服务器响应完成后执行callback函数  xmlHttp.open("GET", "/ajaxdemo1/ProvinceServlet", **true**);//向服务器发送GET请求  xmlHttp.send(**null**);//发送请求  }; |
| // 本方法获取服务器响应的所有省份的名称  **function** callback() {  **if**(**this**.readyState == 4 && **this**.status == 200) {  // 把服务器响应的省份名称，使用逗号分割成字符串数组  **var** provinceNameArray = **this**.responseText.split(", ");  // 遍历每个省份名称，使用每个省份名称创建<option>元素，添加到province的<select>中  **for**(**var** i = 0; i < provinceNameArray.length; i++) {  addProvinceOption(provinceNameArray[i]);  }  // 为province的<select>元素添加onchange事件监听  document.getElementById("province").onchange = loadCities;  }  } |
| // 本函数在province的<select>元素发送变化时执行！  // 本函数会使用当前选中的省份名称为参数，向服务器发送请求，获取当前省份下的所有城市！  **function** loadCities() {  **var** proName = **this**.value;//获取<select>选择的省份名称  /\*  AJAX4步  \*/  **var** xmlHttp = createXMLHttpRequest();//创建异常对象  // 指定回调函数  xmlHttp.onreadystatechange = **function**() {  **if**(xmlHttp.readyState == 4 && xmlHttp.status == 200) {  // 得到服务器响应的xml文档对象  **var** doc = xmlHttp.responseXML;//注意，这里使用的是resopnseXML属性，不是resopnseText  // 获取文档中所有city元素  **var** cityElementList = doc.getElementsByTagName("city");  // 获取html元素：city的<select>  **var** citySelect = document.getElementById("city");  // 删除city的<select>元素的所有子元素  removeChildNodes(citySelect);    // 创建<option>元素，指定文本内容为“请选择”  **var** qxzOption = document.createElement("option");  **var** textNode = document.createTextNode("===请选择===");  qxzOption.appendChild(textNode);  // 把"请选择"这个<option>添加到<select>元素中  citySelect.appendChild(qxzOption);    // 循环遍历每个服务器端响应的每个<city>元素  **for**(**var** i = 0; i < cityElementList.length; i++) {  **var** cityEle = cityElementList[i];  **var** cityName = **null**;  // 获取<city>元素的文本内容！处理浏览器差异！  **if**(window.addEventListener) {  cityName = cityEle.textContent;  } **else** {  cityName = cityEle.text;  }  // 使用城市名称创建<option>，并添加到<select>元素中  addCityOption(cityName);  }  }  };  xmlHttp.open("POST", "/ajaxdemo1/CityServlet", **true**);//打开与服务器的连接  // 因为是POST请求，所以要设置Content-Type请求头  xmlHttp.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");  // 参数为当前选中的省份名称  xmlHttp.send("provinceName=" + proName);  } |
| // 使用proName创建<option>元素添加到<select>元素中  **function** addProvinceOption(proName) {  **var** option = document.createElement("option");//创建<option>元素  **var** textNode = document.createTextNode(proName);//使用省份名称创建文本节点  option.appendChild(textNode);//把省份名称的文本节点添加到<option>元素中  option.setAttribute("value", proName);//使用省份名称来设置<option>元素的value属性  document.getElementById("province").appendChild(option);//把<option>元素添加到<select>元素中  } |
| // 本函数用来创建城市的<option>，并添加到<select>元素中  **function** addCityOption(cityName) {  **var** citySelect = document.getElementById("city");//获取id为city的<select>  **var** cityOption = document.createElement("option");//创建<option>元素  **var** textNode = document.createTextNode(cityName);//使用城市名称创建文本节点  cityOption.appendChild(textNode);//把文本节点添加到<option>元素中  cityOption.setAttribute("value", cityName);//设置<option>元素的value属性为城市名称  citySelect.appendChild(cityOption);//把<option>元素添加到<select>元素中  } |
| //删除指定元素的所有子元素  **function** removeChildNodes(ele) {  **var** nodes = ele.childNodes;//获取当前元素的所有子元素集合  **while**(nodes.length > 0) {//遍历所有子元素  ele.removeChild(nodes[0]);//删除子元素  }  } |

## Jquery实现Ajax

|  |
| --- |
| $.ajax({  url : "/UserServlet.htm",//action处理请求的Servlet的访问路径  type : "get",//method 请求方式：get post  data : {  "username" : uname, //请求提交给服务器的数据 username=uname&userpwd=upwd  "userpwd":upwd  },  dataType : "text",//text(即String)/xml(即xml文档格式)/json(即json数据格式),dataType设置服务器响应客户端的数据对应的数据类型  success : **function**(data) {//监听器=fun，监听器执行时机：响应码是200的时候  $("#checkuser").html(data);//data就是dataType属性指定的数据类型，而值就是服务器响应的数据  },  error:function(){}//异常情况处理  }); |

### JSON

Xml html 一种文档或数据定义的格式

#### 什么是JSON

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。理解类似html文档格式、xml文档、doc文档、ppt、json、JavaScript

JSON是用字符串来表示Javascript对象，例如可以在Servlet中发送一个JSON格式的字符串给客户端Javascript，Javascript可以执行这个字符串，得到一个Javascript对象。

XML也可以用来数据交换，前面已经学习过在Servlet中发送XML给Javascript，然后Javascript再去解析XML。

#### 2　JSON对象语法

JSON 语法：

* 数据在名称/值对中
* 数据由逗号分隔
* 花括号保存对象
* 方括号保存数组

|  |
| --- |
| **//JavaScript支持OOP js定义对象var obj={“属性名”:”属性值”，“方法名”:function(){}}**  **var** person = {"name":"zhangSan", "age":18, "sex":"male"};  Var people=[{"name":"zhangSan", "age":18, "sex":"male"},{"name":"zhangSan", "age":18, "sex":"male"},{"name":"zhangSan", "age":18, "sex":"male"}];  alert(person.name + ", " + person.age + ", " + person.sex); |

　　注意，key也要在双引号中！

JSON值：

* [数字](http://baike.baidu.com/view/37407.htm" \t "_blank)（整数或浮点数），不需要””
* 字符串（在双引号中），需要使用””
* 逻辑值（true 或 false），不需要””
* 数组（在方括号中）使用[]表达值
* 对象（在花括号中）使用{} Function()
* null

|  |
| --- |
| **var** person = {"name":"zhangSan", "age":"18", "sex":"male", "hobby":["cf", "sj", "ddm"]};  alert(person.name + ", " + person.age + ", " + person.sex + ", " + person.hobby); |

带有方法的JSON对象：

|  |
| --- |
| **var** person = {"name":"zhangSan", "getName":**function**() {**return this**.name;}};  alert(person.name);  alert(person.getName()); |

#### JSON与XML比较

* 可读性：XML胜出；
* 解码难度：JSON本身就是JS对象，所以简单很多；
* 流行度：XML已经流行好多年，但在AJAX领域，JSON更受欢迎。

问题：

Java 没有json数据类型？？？

处理过程：java没有json格式，格式良好的json字符串============用阿里提供fastJson

Servlet处理数据库数据之后，response回写String “{\”name\”:\”jack\”}”

Java程序响应给客户端数据类型依然：String

AJAX接收服务器响应数据时：xmlHttpRequest.responseText接收服务器响应json字符串， 不能使用xmlHttpRequest.responseXML接收。

AJAX接收数据之后，不能直接解析。怎么办？？？？？？ String🡺JSON对象==========🡺json\_parse.js

实现方式有两种：

Javascript原生态方法：eval() eval(“1+2”);//3

JSON官方网站提供快速将String解析成json对象的类库。Json\_parse.js

Json是javascript语言描述一种对象数据表示格式

##### 3-1 json和ajax的基本交互案例入门

1、jsp页面的JavaScript脚本代码

|  |
| --- |
| function createXMLHttpRequest(){  try {  return new XMLHttpRequest();  } catch (e) {  try {  return new ActiveXObject("Msxml2.XMLHTTP");  } catch (e) {  try {  return new ActiveXObject("Microsoft.XMLHTTP");  } catch (e) {    }  }  }  }  function clickMe(){  //定义JSON数据  //var json={"name":"jack","age":18,"sex":true,"getInfo":function(){alert("jack的基本信息")}};  //Json字符串  //var jsonStr="{\"name\":\"jack\",\"age\":18,\"sex\":true,\"getInfo\":function(){alert(\"jack的基本信息\")}}";  //alert(jsonStr);  //向servlet发出ajax，ajax请求 响应的文本“{json字符串格式}”  //1.创建核心对象：xmlHttpRequest  var xmlHttp=createXMLHttpRequest();  //2.打开客户端和服务器的连接 open(method,url,true)  xmlHttp.open("GET","${pageContext.request.contextPath}/JsonServlet",true);  //3.发送请求send(null)  xmlHttp.send(null);  //4.监听服务器响应客户端的状态 onreadystatechange()  xmlHttp.onreadystatechange=function(){  if(xmlHttp.readyState==4 && xmlHttp.status==200){  //接收服务器响应的json字符串，并解析  var str=xmlHttp.responseText;  //将字符串转换为json对象  var json=eval("("+str+")");  //将JSON中保存的数据显示在div1内部  var oDiv=document.getElementById("div1");  oDiv.innerText="姓名："+json.name;    oDiv.innerText+="年龄："+json.age;  oDiv.innerText+="性别：";  //NULL undefined作为if条件时，当做false解析  oDiv.innerText+=json.sex?"男":"女";  //json中的函数  json.getInfo();  }  }  } |

2、servlet中服务器的处理java代码

|  |
| --- |
| protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  response.setContentType("text/html;charset=utf-8");  String jsonQuery="{\"name\":\"jack\",\"age\":18,\"sex\":true,\"getInfo\":function(){alert(\"jack的基本信息\")}}";  response.getWriter().write(jsonQuery);  } |

#### 4 fastJSON概述

Fastjson，是阿里巴巴提供的一个Java语言编写的高性能功能完善的JSON库。Fastjson是一个Java语言编写的JSON处理器。

其优势体现在如下几个方面：

1、遵循 http://json.org 标准，为其官方网站收录的参考实现之一。

2、功能强大，支持JDK的各种类型，包括基本的JavaBean、Collection、Map、Date、Enum、泛型。

3、无依赖，不需要引用额外的jar，能够直接跑在JDK上。

4、开源，使用Apache License 2.0协议开源。

5、支持注解、支持全类型序列化，这些都是很好的特性，功能强大

那么，如果获得Fastjson呢？：[https://github.com/alibaba/fastjson](https://github.com/alibaba/fastjson" \t "_blank)

##### 4.1　faseJSON的jar包



##### 4.2　fastjson中的核心类

* fastjson的最主要的使用入口是com.alibaba.fastjson.JSON
* SerializeWriter：相当于StringBuffer
* JSONArray：相当于List<Object>
* JSONObject：相当于Map<String, Object>

注意：JSON反序列化没有真正数组，本质类型都是List<Object>

##### 4.3　JSON类的主要功能

|  |
| --- |
| public static final Object parse(String text); // 把JSON文本parse为JSONObject或者JSONArray  public static final JSONObject parseObject(String text)； // 把JSON文本parse成JSONObject  public static final <T> T parseObject(String text, Class<T> clazz); // 把JSON文本parse为JavaBean  public static final JSONArray parseArray(String text); // 把JSON文本parse成JSONArray  public static final <T> List<T> parseArray(String text, Class<T> clazz); //把JSON文本parse成JavaBean集合  public static final String toJSONString(Object object); // 将JavaBean序列化为JSON文本  public static final String toJSONString(Object object, boolean prettyFormat); // 将JavaBean序列化为带格式的JSON文本  public static final Object toJSON(Object javaObject); 将JavaBean转换为JSONObject或者JSONArray。 |

##### 4.4　fastJSON的基本使用

###### 1 JavaBean对象序列化为json字符串

|  |
| --- |
| @Test  **public** **void** test2() {  BookType bt=**new** BookType();  bt.setId(1);  bt.setTypeName("少儿");  String str1 = JSON.*toJSONString*(bt);  System.***out***.println(str1);  }  控制台输出的结果：{"id":1,"typeName":"少儿"} |
| @Test  **public** **void** test2() {  BookType bt=**new** BookType();  bt.setId(1);  bt.setTypeName("少儿");  String str1 = JSON.*toJSONString*(bt,true);  System.***out***.println(str1);  }  控制台输出的结果：  {  "id":1,  "typeName":"少儿"  } |

###### 2 json字符串转换为JavaBean对象

|  |
| --- |
| @Test  **public** **void** test2() {  BookType type = JSON.*parseObject*(str1,BookType.**class**);  System.***out***.println(type);  }  控制台输出的结果：BookType [id=1, typeName=少儿] |

###### 3 JavaBean对象转换为JSON字符串（JavaBean中含有java.util.Date类型）

|  |
| --- |
| @Test  **public** **void** test2() {  BookInfo book=**new** BookInfo("b001","java开发","詹姆斯","清华大学出版社",**new** Date());  //格式化日期  SerializeConfig dateConfig=**new** SerializeConfig();  dateConfig.put(Date.**class**, **new** SimpleDateFormatSerializer("yyyy-MM-dd"));  String str2 = JSON.*toJSONString*(book,dateConfig);  System.***out***.println(str2);  }  控制台输出的结果：{"bookAuthor":"詹姆斯","bookCode":"b001","bookName":"java开发","publishDate":"2018-09-06","publishPress":"清华大学出版社"} |

或

|  |
| --- |
| @Test  **public** **void** test2() {  BookInfo book=**new** BookInfo("b001","java开发","詹姆斯","清华大学出版社",**new** Date());  String str = JSON.*toJSONStringWithDateFormat*(book, "yyyy-MM-dd");  System.***out***.println(str);  }  控制台输出的结果：{"bookAuthor":"詹姆斯","bookCode":"b001","bookName":"java开发" ,"publishDate":"2018-09-06","publishPress":"清华大学出版社"} |

###### 4 JavaBean对象转换为json字符串（设置对象中null的默认显示值）

|  |
| --- |
| @Test  **public** **void** test2() {  BookInfo book = **new** BookInfo(**null**, "java开发", "詹姆斯", "清华大学出版社", **new** Date());  String str = JSON.*toJSONStringWi****t****hDateFormat*(  book,  "yyyy-MM-dd", SerializerFeature.***WriteNullNumberAsZero***,  SerializerFeature.***WriteNullStringAsEmpty***  );  System.***out***.println(str);  }  控制台输出的结果：{"bookAuthor":"詹姆斯","bookId":0,"bookName":"java开发","bookType":0,"isBorrow":0,"publishDate":"2018-09-06","publishPress":"清华大学出版社"} |

###### 5 JavaBean对象List集合转换为json字符串

|  |
| --- |
| @Test  **public** **void** test2() {  List<BookType> barList = **new** ArrayList<>();  barList.add(**new** BookType(1,"少儿"));  barList.add(**new** BookType(2,"计算机"));  barList.add(**new** BookType(3,"科学"));  String json= JSON.*toJSONString*(barList, **true**);  System.***out***.println(json);}  控制台输出的结果：  [  {  "id":1,  "typeName":"少儿"  },  {  "id":2,  "typeName":"计算机"  },  {  "id":3,  "typeName":"科学"  }  ] |

###### json字符串转换为JavaBean对象List集合

|  |
| --- |
| @Test  **public** **void** test2() {  List<BookType> list = JSON.*parseArray*(json, BookType.**class**);  System.***out***.println(list);  }  控制台输出结果：  [BookType [id=1, typeName=少儿], BookType [id=2, typeName=计算机], BookType [id=3, typeName=科学]] |

###### JavaBean对象Map集合转换为json字符串

注意：标准的json规范中，要求对象的key必须为string

|  |
| --- |
| @Test  **public** **void** test5() {  Map<String,BookType> barList = **new** HashMap<>();  barList.put(“1”,**new** BookType(1,"少儿"));  barList.put(“2”,**new** BookType(2,"计算机"));  barList.put(“3”,**new** BookType(3,"科学"));  String json= JSON.*toJSONString*(barList, **true**); //true:设置转换后JSON字符串格式良好阅读格式  System.***out***.println(json);  }  控制台输出结果：  {  "3":{  "id":3,  "typeName":"科学"  },  "2":{  "id":2,  "typeName":"计算机"  },  "1":{  "id":1,  "typeName":"少儿"  }  } |

###### 8 json字符串转换为JavaBean对象Map集合

|  |
| --- |
| @Test  **public** **void** test2() {  JSONObject jsonObject = JSON.*parseObject*(json);  Set<String> keys = jsonObject.keySet();  **for** (String key : keys) {  System.***out***.println(key+"==="+jsonObject.getObject(key, BookType.**class**));  }  }  控制台输出的结果：  3===BookType [id=3, typeName=科学]  2===BookType [id=2, typeName=计算机]  1===BookType [id=1, typeName=少儿] |

###### 9 数组转换为JSON字符串

|  |
| --- |
| @Test  **public** **void** test2() {  **int**[] arr={23,45,67,88,99};  String json = JSON.*toJSONString*(arr,**true**);  System.***out***.println(json);  }  控制台输出的结果：  [23,45,67,88,99] |

###### 8 json字符串转换为数组

|  |
| --- |
| @Test  **public** **void** test2() {  JSONArray jsonArray = JSON.*parseArray*(json);  **for** (Object o : jsonArray) {  System.***out***.println(o);  }  }  控制台输出的结果：  23  45  67  88  99 |

#### JQuery版本AJAX

1.jsp页面

|  |
| --- |
| <script src=*"*${pageContext.request.contextPath }*/js/jquery-1.8.3.js"* type=*"text/javascript"*></script>  <script>  $(**function**(){  $(":button").click(**function**(){  //发送一个ajax请求，获取一个字符串，显示  $.ajax({  url:"/AjaxDemo/Servlet1",//服务器资源路径  type:"post",//请求方式：get/post  data:{"user":"tom"},//请求体内容  dataType:"xml",//请求数据格式：text==ResponseText string xml==>ResponseXML document对象 json===>jsonObject  success:**function**(data){//readystate==4 && status==200,处理方式  //data表示服务器响应给客户端的字符串或者对象  // $("#h1").text(data.name+","+data.age);  **var** book=data.getElementsByTagName("name")[0];  $("#h1").text(book.textContent);  }  });    $.post();//请求方式post  $.get();//请求方式get  });  });  </script> |

2.servlet的java代码

|  |
| --- |
| **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  // response.getWriter().write("HelloWorld");  String name = request.getParameter("user");  System.***out***.println(name);  // response.getWriter().write("{\"name\":\"jack\",\"age\":18}");  response.setContentType("text/xml;charset=utf-8");  response.getWriter().write("<books><book><name>aaa</name></book></books>");  } |

## JQuery实现ajax的方案语法参考

### 1 $.ajax()的语法示例

|  |
| --- |
| Jquery中提供的ajax功能的语法：  $.ajax({--------------------------------掌握------------------  type : "POST",  url : "/web13/CityServlet",  data : "pid=" + this.value,  dataType : "json",  success : function(cities) {  //清空之前的选项 下拉列表中除了第一项，其他都删除  $("#city>option:not(:first)").remove();  $.each(cities, function(i, city) {  var opt = $("<option value='" + city.cid + "'>"  + city.cname + "</option>");  $("#city").append(opt);  });  }  }); |

### 2 $.get()的语法示例

|  |
| --- |
| $.get(  "/web13/CityServlet",{"pid=" + this.value},  function(cities) {  //清空之前的选项 下拉列表中除了第一项，其他都删除  $("#city>option:not(:first)").remove();  $.each(cities, function(i, city) {  var opt = $("<option value='" + city.cid + "'>"+ city.cname + "</option>");  $("#city").append(opt);  });},  "json"); |

### 3 $.post()的语法示例

|  |
| --- |
| $.post( "/web13/CityServlet",  {pid=" + this.value},  function(cities) {  //清空之前的选项 下拉列表中除了第一项，其他都删除  $("#city>option:not(:first)").remove();  $.each(cities, function(i, city) {  var opt = $("<option value='" + city.cid + "'>"  + city.cname + "</option>");  $("#city").append(opt);  });  },"json"); |

### 4 $.getJSON()的语法示例

|  |
| --- |
| $.getJSON("/web13/CityServlet",{"pid": this.value},  function(cities) {  //清空之前的选项 下拉列表中除了第一项，其他都删除  $("#city>option:not(:first)").remove();  $.each(cities, function(i, city) {  var opt = $("<option value='" + city.cid + "'>"  + city.cname + "</option>");  $("#city").append(opt);  });  }); |

## 作业：

1、用户名的数据库查询，判断是否占用======必做

2、三级联动 基于数据库======选做