# HTTP协议和tomcat服务器

**【学习目标】理解、了解、应用、记忆**

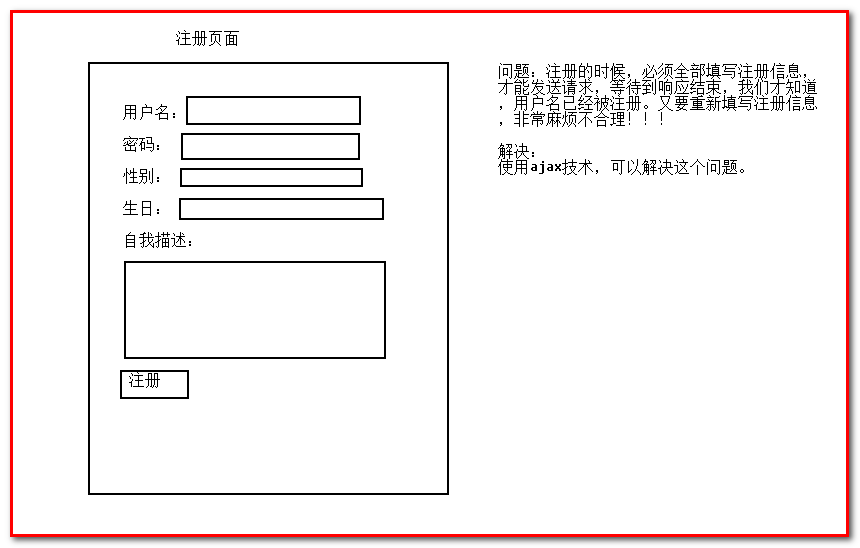
通过今天的学习，参训学员能够：（解释的时候说出二级目标的掌握程度）

1. **【理解】能够阐述Ajax基本概念**
2. **【应用】能够阐述Ajax的运行原理**
3. **【应用】了解JS原生的Ajax技术**
4. **【应用】能够阐述json作用**
5. **【应用】掌握Json的格式与解析**
6. **【应用】掌握 Jquery操作Ajax**

# ajax介绍

## 什么是ajax

问题：用户注册的时候，没有，用户名是否可用提示，导致用户需要反复的提交表单，用户体验不好。



优化：用户在书写完用户名，就，有一个提示，告诉当前书写的用户名是否可用。

Ajax特点：

1 ajax执行，不需要用户控制，自动发送请求和接收响应

2 ajax发送请求的时候，依然按照http协议

3 ajax请求和响应，数据量小

总结什么是ajax：自动发送请求，接受响应，数据量非常小，依然按照http协议做网络的传输，这个就是ajax技术。

**在这个需求中，发送请求应该谁来做？**

浏览器

**让浏览器来帮助发送这个请求，那么程序员如何与浏览器沟通，让它帮助我们发送请求？**

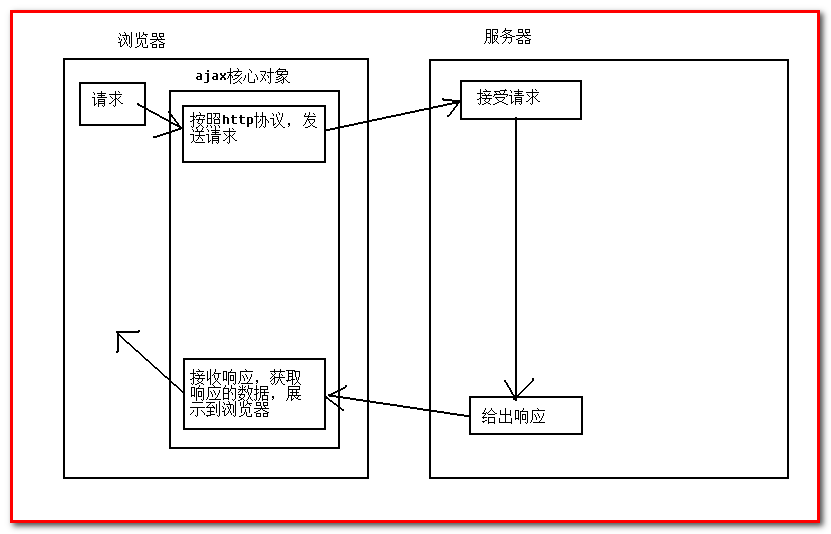
使用的是Javascript代码。

**企业为什么特别喜欢ajax？**

**数据量小，请求和响应速度快，用户体验好**

**钱。企业的通信费用，是按照流量计算，那么数据量越小，越省钱。**

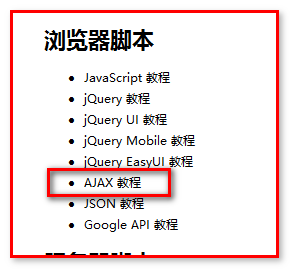
## ajax运行机制



## ajax快速入门案例（获取ajax核心对象、发送请求、接收响应）

API文档：





### 获取XMLHttpRequest对象（ajax核心对象，ajax引擎）

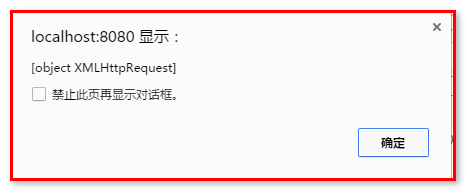
定义：

XMLHttpRequest 是 AJAX 的基础（核心）。

代码演示：

|  |
| --- |
| <script type=*"text/javascript"*>  //获取ajax核心对象的方法  **function** getXHR() {  **var** xmlhttp;  **if** (window.XMLHttpRequest) {  // code for IE7+, Firefox, Chrome, Opera, Safari  xmlhttp = **new** XMLHttpRequest();  } **else** {  // code for IE6, IE5  xmlhttp = **new** ActiveXObject("Microsoft.XMLHTTP");  }  **return** xmlhttp;  }  alert(getXHR());  </script> |

效果：



### 向服务器发送请求——使用open方法和send方法

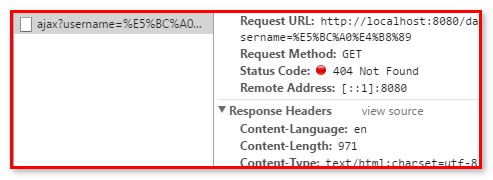
方法截图：



发送请求代码测试：

|  |
| --- |
| //测试使用ajax核心对象，发送请求  **function** test1(){  **var** xhr = getXHR();  xhr.open("get","/day18/ajax?username=张三",**true**);  xhr.send();  }  test1(); |

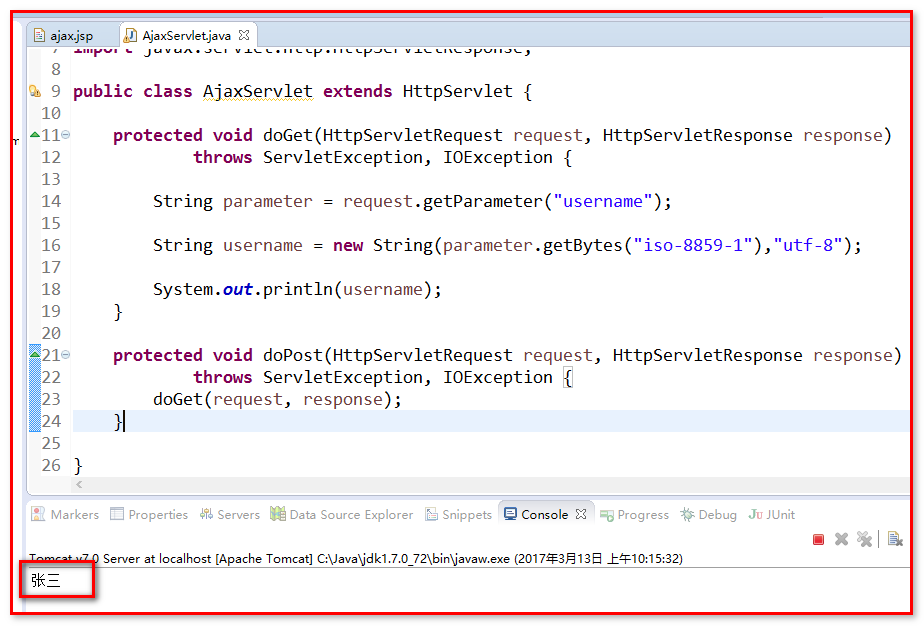
效果：



Servlet接受请求参数代码：

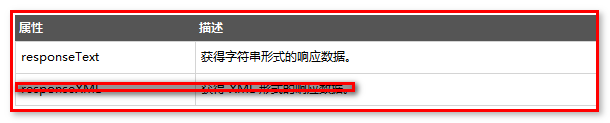
|  |
| --- |
| **package** cn.igeek.web;  **import** java.io.IOException;  **import** javax.servlet.ServletException;  **import** javax.servlet.http.HttpServlet;  **import** javax.servlet.http.HttpServletRequest;  **import** javax.servlet.http.HttpServletResponse;  **public** **class** AjaxServlet **extends** HttpServlet {  **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  String parameter = request.getParameter("username");    String username = **new** String(parameter.getBytes("iso-8859-1"),"utf-8");    System.***out***.println(username);  }  **protected** **void** doPost(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  doGet(request, response);  }  } |

效果：



### 接收服务器响应

API截图：



ajax代码：

|  |
| --- |
| //测试ajax获取响应  **function** test2(){  **var** xhr = getXHR();  xhr.open("get","/day18/ajax?username=张三",**true**);  xhr.send();    /\*\*    有一个妹子，半夜十二点，搜索一些东西  妹子（上海），百度的服务器（北京）（网络延迟）    服务器发送响应到浏览器（网络延迟）    为什么获取响应数据没有效果？  就是没有考虑网络延迟的问题    \*/    //浏览器端准备接受响应  **var** data = xhr.responseText;  alert(data);  } |

servlet代码：

|  |
| --- |
| **package** cn.igeek.web;  **import** java.io.IOException;  **import** javax.servlet.ServletException;  **import** javax.servlet.http.HttpServlet;  **import** javax.servlet.http.HttpServletRequest;  **import** javax.servlet.http.HttpServletResponse;  **public** **class** AjaxServlet **extends** HttpServlet {  **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  String parameter = request.getParameter("username");    String username = **new** String(parameter.getBytes("iso-8859-1"),"utf-8");    System.***out***.println(username);    response.setContentType("text/html;charset=utf-8");  response.getWriter().write("测试ajax响应成功！！！");  }  **protected** **void** doPost(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  doGet(request, response);  }  } |

### 设置onreadystatechange事件执行函数（等待服务器响应）

API截图：



ajax代码演示：

|  |
| --- |
| //测试设置ajax等待服务器响应  **function** test3(){  //获取核心对象  **var** xhr = getXHR();  //设置等待服务器响应  xhr.onreadystatechange=**function**(){  //4: 请求已完成，且响应已就绪,200: "OK"  **if** (xhr.readyState==4 && xhr.status==200){  **var** data=xhr.responseText;  alert(data);  }  }  //发送请求  xhr.open("get","/day18/ajax?username=张三",**true**);  xhr.send();  }  test3(); |

Servlet代码演示：

|  |
| --- |
| **package** cn.igeek.web;  **import** java.io.IOException;  **import** javax.servlet.ServletException;  **import** javax.servlet.http.HttpServlet;  **import** javax.servlet.http.HttpServletRequest;  **import** javax.servlet.http.HttpServletResponse;  **public** **class** AjaxServlet **extends** HttpServlet {  **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  String parameter = request.getParameter("username");    String username = **new** String(parameter.getBytes("iso-8859-1"),"utf-8");    System.***out***.println(username);    response.setContentType("text/html;charset=utf-8");  response.getWriter().write("测试ajax响应成功！！！");  }  **protected** **void** doPost(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  doGet(request, response);  }  } |

ajax小结（ajax代码怎么写）：

1 获取核心对象

2 设置等待服务器响应

3 发送请求

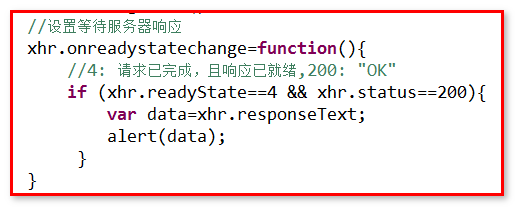
## XMLHttpRequest API 详解

### onreadystatechange属性（重点）

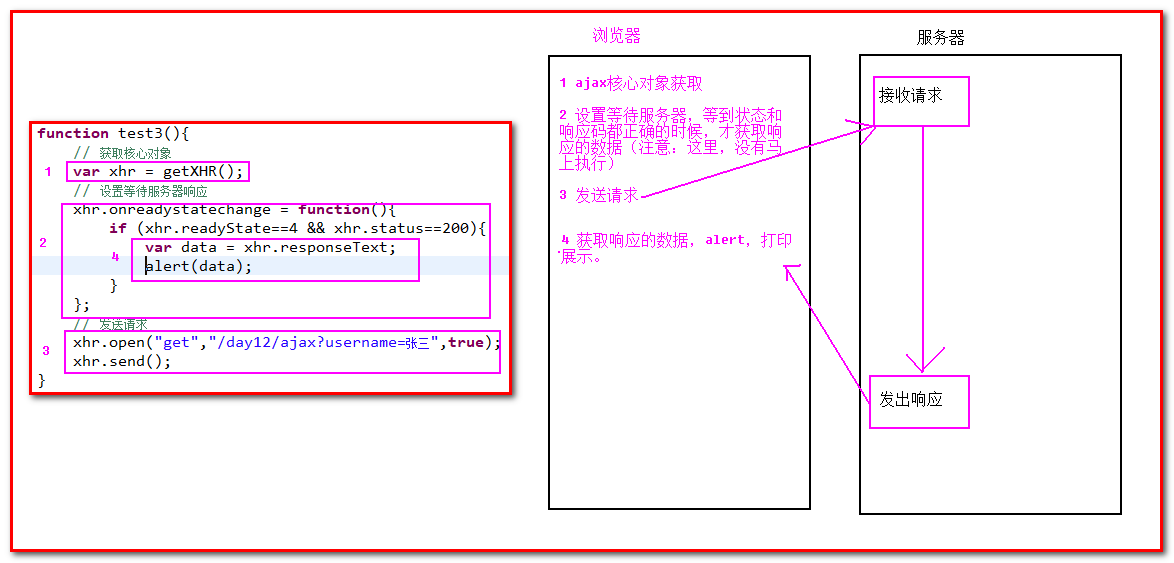
是什么：

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

代码截图：



执行机制图解：



### open方法（重点）

是什么：规定请求的类型、URL 以及是否异步处理请求。

代码截图:



**使用post方式还是get方式?**

官方建议：



演示发送post请求和中文请求参数:

|  |
| --- |
| //测试ajax发送post请求  **function** test4(){  **var** xhr = getXHR();  xhr.onreadystatechange=**function**(){  **if** (xhr.readyState==4 && xhr.status==200){  **var** data=xhr.responseText;  document.getElementById("msg").innerHTML = data;  }  }  //async false 所有的请求和响应，需要排队执行（同步）  //async true 所有的请求和响应,不需要排队执行（插队执行）（异步）  xhr.open("post","/day18/ajax",**true**);  //设置请求模拟表单提交数据  xhr.setRequestHeader("Content-type","application/x-www-form-urlencoded");  xhr.send("username=张三");  }  //test4(); |

servlet修改：

|  |
| --- |
| **package** cn.igeek.web;  **import** java.io.IOException;  **import** javax.servlet.ServletException;  **import** javax.servlet.http.HttpServlet;  **import** javax.servlet.http.HttpServletRequest;  **import** javax.servlet.http.HttpServletResponse;  **public** **class** AjaxServlet **extends** HttpServlet {  **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  request.setCharacterEncoding("utf-8");  String username = request.getParameter("username");    //String username = new String(parameter.getBytes("iso-8859-1"),"utf-8");    System.***out***.println(username);  **try** {  Thread.*sleep*(5000);  } **catch** (InterruptedException e) {  e.printStackTrace();  }  response.setContentType("text/html;charset=utf-8");  response.getWriter().write("测试ajax响应成功！！！");  }  **protected** **void** doPost(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  doGet(request, response);  }  } |

第三个参数async： True（异步） 或 False（同步）？

准备：

async false 所有的请求和响应，需要排队执行（同步）

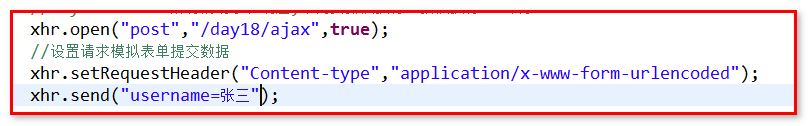
async true 所有的请求和响应,不需要排队执行（插队执行）（异步）

### setRequestHeader方法

是什么：

如果需要像 HTML 表单那样 POST 数据，请使用 setRequestHeader() 来添加 HTTP 头

代码截图：



注意：这个方法相当于，设置了表单的enctype属性的默认值，来模拟表单发送数据

### send方法

是什么：

将请求发送到服务器。

代码截图

仅限post请求方式



### readyState属性（重点）

是什么：

存有 XMLHttpRequest 的状态。从 0 到 4 发生变化。

代码截图：



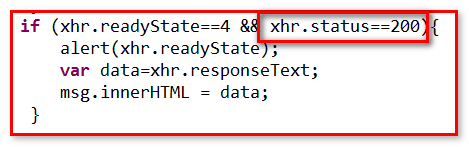
**ReadyState其他状态的应用（提示用户请求处理中）：**

|  |
| --- |
| //测试使用readyState 其他的值  //优化页面的提示信息，告诉用户，服务器在处理你的请求  window.onload = **function** test5(){  **var** xhr = getXHR();  **var** msg = document.getElementById("msg");  xhr.onreadystatechange=**function**(){  **if** (xhr.readyState==1){  alert(xhr.readyState);  msg.innerHTML = "1: 服务器连接已建立 ";  }  **if** (xhr.readyState==2){  alert(xhr.readyState);  msg.innerHTML = "2: 请求已接收 ";  }  **if** (xhr.readyState==3){  alert(xhr.readyState);  msg.innerHTML = "3: 请求处理中 ";  }  **if** (xhr.readyState==4 && xhr.status==200){  alert(xhr.readyState);  **var** data=xhr.responseText;  msg.innerHTML = data;  }  }  xhr.open("get","/day18/ajax?username=张三",**true**);  xhr.send();  } |
| <div id=*"msg"*></div> |

### status属性

是什么：存储响应状态码

代码截图：



复习下响应状态码：

404：找不到

500：出错了

302：重定向

200：OK

405：没有这个方法

401：权限不足

### responseText属性

是什么：获取字符串形式的响应数据

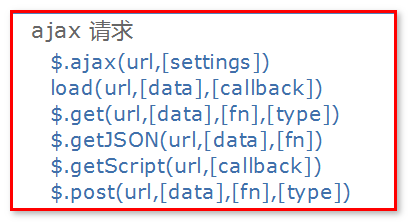
代码截图：



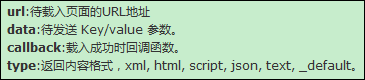
## jQuery Ajax支持的API

jQuery对Ajax的使用提供了非常好的封装，简单的很！

jQuery提供了6个编写Ajax的方法：



* **$.ajax(url, [settings])** 最基本的ajax编程方法，推荐使用！
* **load(url, [data], [callback])** 载入HTML代码并插入至DOM中，例如：$(“#mydiv”).load(“a.html”);
* **$.get(url, [data], [callback], [type])** Ajax的get方式请求，一般用于获取数据
* **$.post(url, [data], [callback], [type])** Ajax的post方式请求，一般用于发送表单数据



* **$.getJSON(url, [data], [fn])** Ajax跨域获取服务器数据 例子：[www.taobao.com](http://www.taobao.com) 支付 [www.alipay.com](http://www.alipay.com)
* **$.getScript(url, [callback])** 载入并执行一个跨域的js文件

我们学习使用最基本的$.ajax() $.get() $.post()

**$.ajax(url, [settings])**

<script type="text/javascript">

$.ajax({

//请求方式

type: "POST",

//请求路径

url: "${root}/ajax",

//发送参数

data: "name=John&location=Boston",

//返回的数据类型：

/\*\*

"xml": 返回 XML 文档，可用 jQuery 处理。

"html": 返回纯文本 HTML 信息；包含的script标签会在插入dom时执行。

"script": 返回纯文本 JavaScript 代码。

"json": 返回 JSON 数据 。

"jsonp": JSONP 格式。

"text": 返回纯文本字符串

\*/

dataType: "text",

//回调函数

//请求成功的回调函数

success: **function**(msg){

alert( "Data Saved: " + msg );

},

//请求失败的回调函数

error:**function**(msg){

alert( "Error");

}

});

</script>

效果：

get代码：

/\*

url:待载入页面的URL地址

data:待发送 Key/value 参数。

callback:载入成功时回调函数。

type:返回内容格式，xml, html, script, json, text, \_default。

\*/

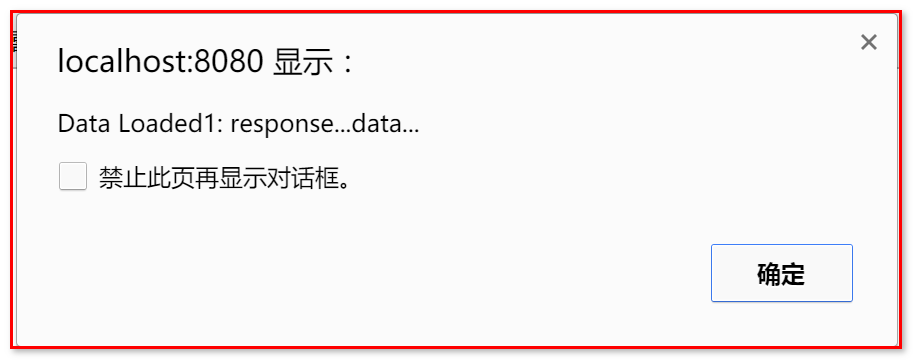
$.get("ajax", { name: "John", time: "2pm" },

function(data){

alert("Data Loaded1: " + data);

});

效果：



Post代码：

/\*

url:待载入页面的URL地址

data:待发送 Key/value 参数。

callback:载入成功时回调函数。

type:返回内容格式，xml, html, script, json, text, \_default。

\*/

$.post("ajax", { name: "John", time: "2pm" },

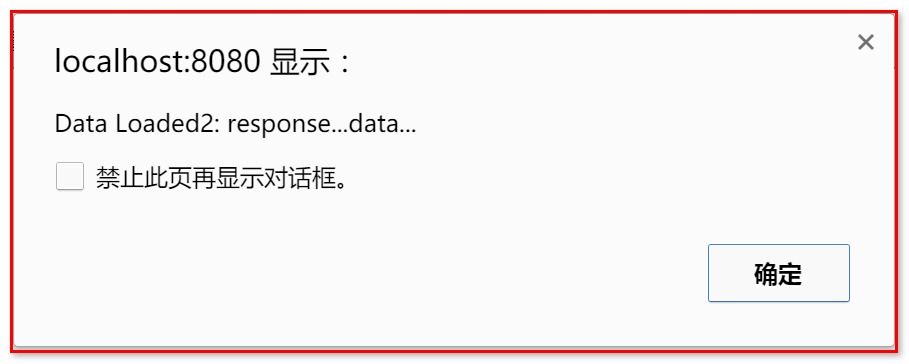
function(data){

alert("Data Loaded2: " + data);

});

});

效果：



# JSON技术与ajax综合应用（重点：必须掌握）

## Json介绍

json技术的由来：

设计一个场景：使用ajax，处理提交订单请求，获取到的响应数据为： -1 888 201701014xxxx

第一眼，看到这个数据程序员是：懵逼！！！

因此，我们在发送数据的是最好数据本身带有意义！！！

数据本身携带意义：

<msg>

<status>-1</status>

<time>888</time>

<orderid> 201701014xx </orderid>

</msg>

Xml问题：导致传输的数据量变大，解析和创建的操作复杂

于是，有人出现新的办法：

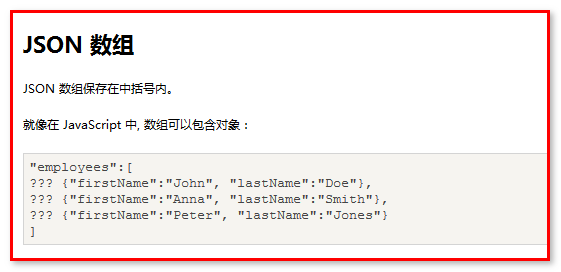
A公司 Status : -1 time:888 orderid: 20170913xxxx

B公司 Status =-1 time=888 orderid=20170913xxxx

…………………..

格式太多导致开发困难，为了统一以下数据传递的格式，定义出了json：





Json格式数据，仅仅是一个文本操作非常麻烦：

## JSON 字符串转换为 JavaScript 对象：

"firstName":"John"

直接操作对象：

Obj. firstName

直接对象.属性的方式获取数据。



|  |
| --- |
| **function** test5(){  //定义json格式数据  **var** text = '{ "employees" : [' +  '{ "firstName":"John" , "lastName":"Doe" },' +  '{ "firstName":"Anna" , "lastName":"Smith" },' +  '{ "firstName":"Peter" , "lastName":"Jones" } ]}';  //将数据转换成js对象，方便操作  **var** obj = JSON.parse(text);  alert(obj.employees[0].firstName);  }  test5(); |

### 使用json数据和ajax技术，页面显示省市县信息（重点:必须掌握）

需求：完成省市县三级联动

页面截图：



效果：我要做到什么样子？

数据库分析：



*#获取省和直辖市的数据*

**select** \* **from** province **where** parentid = 0;

*#获取陕西下所有市*

**select** \* **from** province **where** parentid = 61;

*#获取商洛地区下所有的县*

**select** \* **from** province **where** parentid = 6125;

思路：

1. 页面加载完成之后，省地区的数据如何加载完成？

Window.onload,控制页面加载事件

使用ajax发送请求获取数据（parentid=0）

Servlet Service Dao 获取数据

将数据发出响应给浏览器

将数据转换成option标签添加进去

1. 在用户选择下拉省一级选项之后，市地区数据如何加载完成？

Onchange控制数据变更

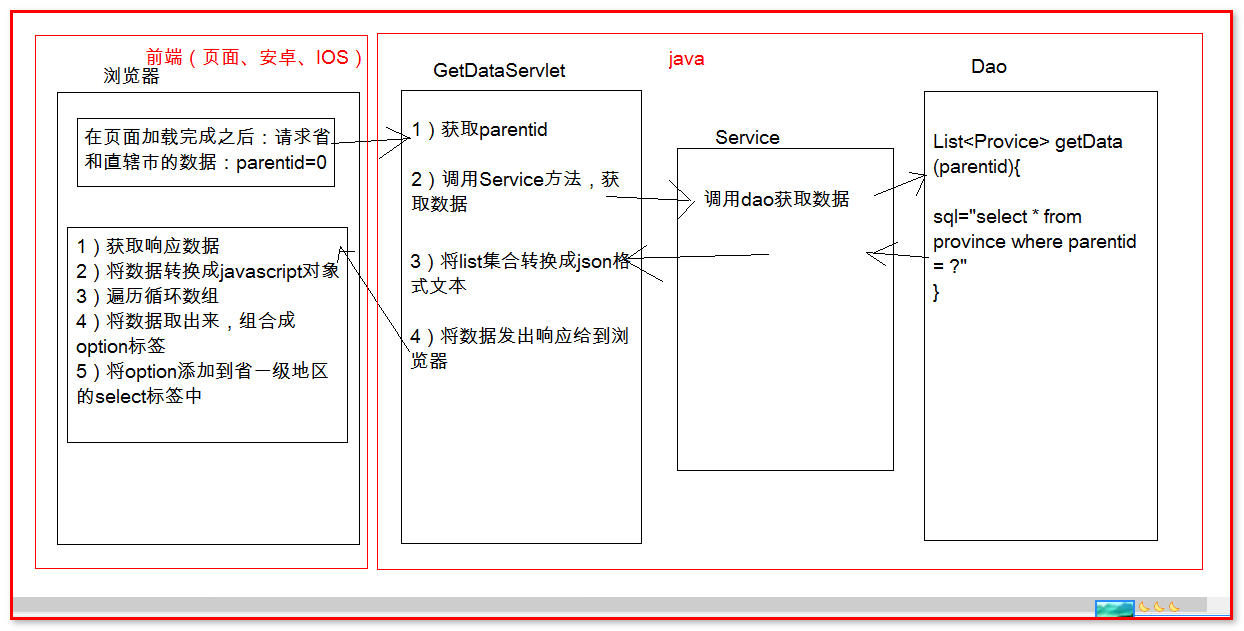
使用ajax发送请求获取数据（parentid=当前选中的地区代码）

Servlet Service Dao 获取数据

将数据发出响应给浏览器

将数据转换成option标签添加进去

流程图：



页面js：

|  |
| --- |
| <%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"UTF-8"*%>  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  <html>  <head>  <title>省市页面</title>  <script type=*"text/javascript"*>  **function** getXHR() {  **var** xmlhttp;  **if** (window.XMLHttpRequest) {  // code for IE7+, Firefox, Chrome, Opera, Safari  xmlhttp = **new** XMLHttpRequest();  } **else** {  // code for IE6, IE5  xmlhttp = **new** ActiveXObject("Microsoft.XMLHTTP");  }  **return** xmlhttp;  }  window.onload = **function**(){    **var** xhr = getXHR();  xhr.onreadystatechange=**function**(){    **if** (xhr.readyState==4 && xhr.status==200){  **var** data=xhr.responseText;  //将数据转换成javascript对象  **var** arr = JSON.parse(data);  //遍历数组，将数据取出，创建新的option标签  **var** province = document.getElementById("province");  **for** (**var** i = 0; i < arr.length; i++) {  **var** option = document.createElement("option");  option.innerHTML = arr[i].cityName;  option.setAttribute("value",arr[i].codeid);    province.appendChild(option);  }  }  }  xhr.open("get","/day18/getData?parentid=0",**true**);  xhr.send();  };  **function** \_getCity(\_this){  **var** city = document.getElementById("city");  city.length = 1;  **var** area = document.getElementById("area");  area.length = 1;    **var** xhr = getXHR();  xhr.onreadystatechange=**function**(){    **if** (xhr.readyState==4 && xhr.status==200){  **var** data=xhr.responseText;  //将数据转换成javascript对象  **var** arr = JSON.parse(data);  //遍历数组，将数据取出，创建新的option标签    **for** (**var** i = 0; i < arr.length; i++) {  **var** option = document.createElement("option");  option.innerHTML = arr[i].cityName;  option.setAttribute("value",arr[i].codeid);    city.appendChild(option);  }  }  }  xhr.open("get","/day18/getData?parentid="+\_this.value,**true**);  xhr.send();  }  **function** \_getArea(\_this){  **var** area = document.getElementById("area");  area.length = 1;    **var** xhr = getXHR();  xhr.onreadystatechange=**function**(){    **if** (xhr.readyState==4 && xhr.status==200){  **var** data=xhr.responseText;  //将数据转换成javascript对象  **var** arr = JSON.parse(data);  //遍历数组，将数据取出，创建新的option标签    **for** (**var** i = 0; i < arr.length; i++) {  **var** option = document.createElement("option");  option.innerHTML = arr[i].cityName;  option.setAttribute("value",arr[i].codeid);    area.appendChild(option);  }  }  }  xhr.open("get","/day18/getData?parentid="+\_this.value,**true**);  xhr.send();    }  </script>    </head>    <body>  <center>  <select id=*"province"* name=*"province"* onchange=*"\_getCity(this);"*>  <option value=*"none"*>--请选择省--</option>  </select>  <select id=*"city"* name=*"city"* onchange=*"\_getArea(this);"*>  <option value=*"none"*>--请选择市--</option>  </select>  <select id=*"area"* name=*"area"* >  <option value=*"none"*>--请选择县或区--</option>  </select>  </center>  </body>  </html> |

Servlet代码：

|  |
| --- |
| **package** cn.igeek.web;  **import** java.io.IOException;  **import** java.sql.SQLException;  **import** java.util.List;  **import** javax.servlet.ServletException;  **import** javax.servlet.http.HttpServlet;  **import** javax.servlet.http.HttpServletRequest;  **import** javax.servlet.http.HttpServletResponse;  **import** org.apache.commons.dbutils.QueryRunner;  **import** org.apache.commons.dbutils.handlers.BeanListHandler;  **import** cn.igeek.domain.Province;  **import** cn.igeek.utils.JDBCUtils;  **import** flexjson.JSONSerializer;  **public** **class** GetDataServlet **extends** HttpServlet {  **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  //接受参数  String parameter = request.getParameter("parentid");  **int** parentid = Integer.*parseInt*(parameter);    //调用Service方法，获取数据  QueryRunner qr = **new** QueryRunner(JDBCUtils.*getDataSource*());  String sql = "select \* from province where parentid = ?";  **try** {  List<Province> list = qr.query(sql, **new** BeanListHandler<Province>(Province.**class**), parentid);  //将数据转换成json格式的字符串  JSONSerializer serializer = **new** JSONSerializer();  String serialize = serializer.serialize(list);  //将数据发出响应给浏览器  response.setContentType("text/html;charset=utf-8");  response.getWriter().write(serialize);  } **catch** (SQLException e) {  e.printStackTrace();  }  }  **protected** **void** doPost(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  doGet(request, response);  }  } |

|  |
| --- |
| <%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"UTF-8"*%>  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  <html>  <head>  <title>省市页面</title>  <script type=*"text/javascript"* src=*"js/jquery.min.js"*></script>  <script type=*"text/javascript"*>  //ele.onclick = function(){alert(this.value)};  /\*    this:  1 onchange="aaa(this.value);" 当前标签  2 ele.onclick = function(){alert(this.value)}; 当前调用事件的元素  3 $(data).each(function(){alert(this)}); 当前被循环到的元素    \*/  //优化思路： 1 抽取相同代码 2 不同的变量作为方法的参数  **function** loadData(value,ele){  $.get("${pageContext.request.contextPath}/getData?parentid="+value,**function**(data){  $(data).each(**function**(){  //创建option标签  ele.append('<option value="'+**this**.codeid+'">'+**this**.cityName+'</option>');  });  },"json");  }  $(**function**(){  **var** $pro = $("#province");  //加载省和直辖市数据  loadData(0,$pro);  //加载市  $("#province,#city").change(**function**(){  //清空原来的数据  $(**this**).nextAll().prop("length",1);  loadData(**this**.value,$(**this**).next());  });  });  </script>  </head>    <body>  <center>  <select id=*"province"*>  <option value=*"none"*>--请选择省--</option>  </select>  <select id=*"city"*>  <option value=*"none"*>--请选择市--</option>  </select>  <select id=*"area"*>  <option value=*"none"*>--请选择县或区--</option>  </select>  </center>  </body>  </html> |

## 作业：

1）使用ajax发送请求（open方法和send方法）

2）使用ajax发送请求，并且接收数据（onreadystatechange、readyState、status属性）

3）完成省市县三级联动案例（使用flexJson解析成json格式文本）