AJAX

1、熟悉什么是Ajax

2、掌握Ajax工作原理

3、掌握Ajax的使用

4、熟练Ajax的综合练习

第一章 JSON概述

1.1 什么是json

JSON(JavaScript Object Notation, JS 对象标记) 是一种轻量级的数据交换格式。它基于 ECMAScript (w3c制定的js规范)的一个子集，采用完全独立于编程语言的文本格式来存储和表示数据。简洁和清晰的层次结构使得 JSON 成为理想的数据交换语言。 易于人阅读和编写，同时也易于机器解析和生成，并有效地提升网络传输效率。

1.2 json语法

[] 表示数组

{} 表示对象

"" 表示是属性名或字符串类型的值

: 表示属性和值之间的间隔符

, 表示多个属性的间隔符或者是多个元素的间隔符

第二章 JSON解析

要解析的字符串：

//对象嵌套数组嵌套对象

String json1="{'id':1,'name':'JAVAEE-1801','stus':[{'id':101,'name':'刘一','age':16}]}";

//数组

String json2="['北京','天津','杭州']";

2.1 原生解析

|  |
| --- |
| public class JSONTest {  //解析JSON  @Test  public void test1() throws JSONException{  //对象嵌套数组嵌套对象  String json1="{'id':1,'name':'JAVAEE-1703','stus':[{'id':101,'name':'刘一','age':16}]}";  //数组  String json2="['北京','天津','杭州']";  //1、  //解析第一层---对象  JSONObject jObject1=new JSONObject(json1);  Grade grade=new Grade();  grade.setId(jObject1.getInt("id"));  grade.setName(jObject1.getString("name"));  ArrayList<Student> stus=new ArrayList<>();  grade.setStus(stus);  //解析第二层----数组  JSONArray jsonArray2=jObject1.getJSONArray("stus");  //遍历数组获取元素----对象  for(int i=0;i<jsonArray2.length();i++){  //解析第三层----对象  JSONObject jObject3=jsonArray2.getJSONObject(i);  Student student=new Student(jObject3.getInt("id"), jObject3.getString("name"), jObject3.getInt("age"));  grade.getStus().add(student);  }  System.out.println(grade);  //2、  //获取数组对象  JSONArray jArray=new JSONArray(json2);  ArrayList<String> list=new ArrayList<>();  //遍历获取元素  for(int i=0;i<jArray.length();i++){  //jArray.optString(i);//等价于getXXX  list.add(jArray.getString(i));  }  System.out.println("解析结果："+list);  }  //生成JSON  @Test  public void test2() throws JSONException{  JSONObject jo1=new JSONObject();  jo1.put("id", 1001);  jo1.put("name", "范老");  jo1.put("age", 20);  JSONObject jo2=new JSONObject();  jo2.put("id", 1002);  jo2.put("name", "平平");  jo2.put("age", 19);  JSONArray ja2=new JSONArray();  ja2.put(jo1);  ja2.put(jo2);  JSONObject jo3=new JSONObject();  jo3.put("id", 11);  jo3.put("name", "JAVAEE-1704");  jo3.put("stus",ja2);  String json=jo3.toString();  System.out.println(json);  }  } |

2.2 FASTJSON解析 ： 阿里巴巴公司提供

|  |
| --- |
| public class FASTJson {  //解析  @Test  public void test1() {  // 对象嵌套数组嵌套对象  String json1 = "{'id':1,'name':'JAVAEE-1703','stus':[{'id':101,'name':'刘铭','age':16}]}";  // 数组  String json2 = "['北京','天津','杭州']";  //1、  //静态方法  Grade grade=JSON.parseObject(json1, Grade.class);  System.out.println(grade);  //2、  List<String> list=JSON.parseArray(json2, String.class);  System.out.println(list);  }  //生成  @Test  public void test2(){  ArrayList<Student> list=new ArrayList<>();  for(int i=1;i<3;i++){  list.add(new Student(101+i, "码子", 20+i));  }  Grade grade=new Grade(100001,"张三", list);  String json=JSON.toJSONString(grade);  System.out.println(json);  }  } |

2.3 Gson : google公司提供

|  |
| --- |
| Gson gson = **new** Gson();  //对象转json字符串  String jsonstr3 = gson.toJson(per);  System.***out***.println(jsonstr3);  String jsonstr4 = gson.toJson(pers);  System.***out***.println(jsonstr4);  // json字符串转对象  String str= "{'girl':['Mary','翠花','冰冰姐'],'height':1.8,'name':'李高强','weight':180}";  Person per2 = gson.fromJson(str, Person.**class**);  System.***out***.println(per2);  String str2 = "[{'girl':['冰冰姐'],'height':1.8,'name':'李高强','weight':180},{'girl':['靓颖妹'],'height':1.8,'name':'马金','weight':180},{'girl':['mary','rose'],'height':1.8,'name':'张迪','weight':180},{'girl':['韩梅梅'],'height':1.8,'name':'边东雪','weight':180}]";  List<Person> pers2 = gson.fromJson(str2, **new** TypeToken<List<Person>>(){}.getType());  System.***out***.println(pers2); |

第三章 Ajax概述

3.1 什么是AJAX

AJAX = 异步 JavaScript 和 XML。

AJAX 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。

AJAX 是一种用于创建快速动态网页的技术。

通过在后台与服务器进行少量数据交换，AJAX 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

传统的网页（不使用 AJAX）如果需要更新内容，必需重载整个网页面。

3.2 创建XMLHttpRequest对象

XMLHttpRequest对象是AJAX的基础。

所有现代浏览器均支持 XMLHttpRequest 对象（IE5 和 IE6 使用 ActiveXObject）。

XMLHttpRequest 用于在后台与服务器交换数据。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

所有现代浏览器（IE7+、Firefox、Chrome、Safari 以及 Opera）均内建 XMLHttpRequest 对象。创建 XMLHttpRequest 对象的语法：

var xmlhttp=new XMLHttpRequest();

老版本的 Internet Explorer （IE5 和 IE6）使用 ActiveX 对象：

var xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

|  |
| --- |
| var xmlhttp;  if (window.XMLHttpRequest)  {  // IE7+, Firefox, Chrome, Opera, Safari 浏览器执行代码  xmlhttp=new XMLHttpRequest();  }  else  {  // IE6, IE5 浏览器执行代码  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");  } |

3.5 XMLHttpRequest请求

如需将请求发送到服务器，我们使用 XMLHttpRequest 对象的 open() 和 send() 方法：

|  |
| --- |
| xmlhttp.open("GET","ajax\_info.txt",true);  xmlhttp.send(); |

|  |  |
| --- | --- |
| 方法 | 描述 |
| 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 POST\*url\*：文件在服务器上的位置\*async\*：true（异步）或 false（同步） |
| send(\*string\*) | 将请求发送到服务器。\*string\*：仅用于 POST 请求 |

一个简单的 GET 请求：

|  |
| --- |
| xmlhttp.open("GET","/try/ajax/demo\_get2.php?fname=Henry&lname=Ford",true);  xmlhttp.send(); |

在上面的例子中，您可能得到的是缓存的结果。

为了避免这种情况，请向 URL 添加一个唯一的 ID：

|  |
| --- |
| xmlhttp.open("GET","/try/ajax/demo\_get.php?t=" + Math.random(),true);  xmlhttp.send(); |

一个简单 POST 请求：

|  |
| --- |
| xmlhttp.open("POST","/try/ajax/demo\_post2.php",true);  xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");  xmlhttp.send("fname=Henry&lname=Ford"); |

通过 AJAX，JavaScript 无需等待服务器的响应，而是：

- 在等待服务器响应时执行其他脚本

- 当响应就绪后对响应进行处理

当使用 async=true 时，请规定在响应处于 onreadystatechange 事件中的就绪状态时执行的函数：

|  |
| --- |
| xmlhttp.onreadystatechange=function()  {  if (xmlhttp.readyState==4 && xmlhttp.status==200)  {  document.getElementById("myDiv").innerHTML=xmlhttp.responseText;  }  }  xmlhttp.open("GET","/try/ajax/ajax\_info.txt",true);  xmlhttp.send(); |

3.6 XMLHttpRequest响应

如需获得来自服务器的响应，请使用 XMLHttpRequest 对象的 responseText 或 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 |

3.7 readyState

\*\*\*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 改变时，就会触发 onreadystatechange 事件。

readyState 属性存有 XMLHttpRequest 的状态信息。

下面是 XMLHttpRequest 对象的三个重要的属性：

|  |  |
| --- | --- |
| 属性 | 描述 |
| 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 到 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 |

|  |
| --- |
| xmlhttp.onreadystatechange=function()  {  if (xmlhttp.readyState==4 && xmlhttp.status==200)  {  document.getElementById("myDiv").innerHTML=xmlhttp.responseText;  }  }  xmlhttp.open("GET","/try/ajax/ajax\_info.txt",true);  xmlhttp.send(); |

\*\*注意：\*\* onreadystatechange 事件被触发 5 次（0 - 4），对应着 readyState 的每个变化。

第四章 Ajax的使用

4.1 Ajax与服务器交互

\* 模拟登陆验证

1.编写注册页面

<%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>

<%

String path = request.getContextPath();

String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";

%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

<head>

<base href="<%=basePath%>">

<title>My JSP 'index.jsp' starting page</title>

<meta http-equiv="pragma" content="no-cache">

<meta http-equiv="cache-control" content="no-cache">

<meta http-equiv="expires" content="0">

<meta http-equiv="keywords" content="keyword1,keyword2,keyword3">

<meta http-equiv="description" content="This is my page">

<script type="text/javascript" src="./ajax.js"></script>

<!--

<link rel="stylesheet" type="text/css" href="styles.css">

-->

</head>

<body>

<center>

<font color="red" size="7">qq注册页面</font>

<input type="text" name="username" onkeyup="kp(this)" /> <span id="sp"></span> <br/>

<input type="password" name="password"/><br/>

<input type="submit" value="注册"/>

</center>

<script type="text/javascript">

//当用户名输入框输入内容就调用此方法

function kp(ipt){

//1.获取input输入框的value

var value = ipt.value;

//2.进行验证

if(value != null && value !=""){

//1-5

//1.创建Ajax

var xmlhttp = getAjax();

//2.设置状态改变监听

xmlhttp.onreadystatechange = function(){

//5获取响应数据

if(xmlhttp.readyState == 4 && xmlhttp.status ==200)

{

var result = xmlhttp.responseText; //获取结果

// 1 行 2 不行

//1.找到span标签

var sp = document.getElementById("sp");

if(result == "1"){

//成功的 span 提示一句绿色的话

sp.innerHTML=""; //清空

var ft = document.createElement("font");

var fttext = document.createTextNode("恭喜您!可以注册!!"); //文本标签

ft.setAttribute("color", "green");

ft.appendChild(fttext);

sp.appendChild(ft);

}else{

//失败的 span 提示一句红色的话

sp.innerText="用户已经被注册!";

sp.style.color = "red";

}

}

}

//3.设置ajax method url

xmlhttp.open("POST",

"${pageContext.request.contextPath}/servlet/DealServlet");

//4.发送请求

//设置一个请求头

xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");

xmlhttp.send("value="+value);

}

}

</script>

</body>

</html>

4.2 编写Ajax处理servlet

public class DealServlet extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

//0.设置编码格式

request.setCharacterEncoding("UTF-8");

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

//1.

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

//2.

String result = null;

if (value.equals("root") | value.equals("admin")) {

result = "2";

//代表已经存在

}else{

//可以注册

result = "1";

}

//3.

response.getWriter().write(result);

}

}