# AJAX

## 简介

* AJAX 是 Asynchronous JavaScript And XML 的简称。直译为，异步的JS和XML。
* AJAX的实际意义是，不发生页面跳转、异步载入内容并改写页面内容的技术。
* AJAX也可以简单的理解为通过JS向服务器发送请求。
* AJAX这门技术很早就被发明，但是直到2005年被谷歌的大量使用，才在市场中流行起来，可以说Google为AJAX的推广起到到推波助澜的作用。
* 异步处理：
  + 同步处理：
    - AJAX出现之前，我们访问互联网时一般都是同步请求，也就是当我们通过一个页面向服务器发送一个请求时，在服务器响应结束之前，我们的整个页面是不能操作的，也就是直观上来看他是卡主不动的。
    - 这就带来了非常糟糕的用户体验。首先，同步请求时，用户只能等待服务器的响应，而不能做任何操作。其次，如果请求时间过长可能会给用户一个卡死的感觉。最后，同步请求的最大缺点就是即使整个页面中只有一小部分内容发生改变我们也要刷新整个页面。
  + 异步处理：
    - 而异步处理指的是我们在浏览网页的同时，通过AJAX向服务器发送请求，发送请求的过程中我们浏览网页的行为并不会收到任何影响，甚至主观上感知不到在向服务器发送请求。当服务器正常响应请求后，响应信息会直接发送到AJAX中，AJAX可以根据服务器响应的内容做一些操作。
    - 使用AJAX的异步请求基本上完美的解决了同步请求带来的问题。首先，发送请求时不会影响到用户的正常访问。其次，即使请求时间过长，用户不会有任何感知。最后，AJAX可以根据服务器的响应信息局部的修改页面，而不需要整个页面刷新。

## 请求对象

* 向服务器发送请求，毫无疑问需要使用Http协议，也就是我们需要通过JS来向服务器发送一个请求报文。这里我们来回顾一下请求报文的格式：

|  |
| --- |
| 请求首行 |
| 请求头 |
| 空行 |
| 请求体 |

* + 这是一个请求报文的格式，那我们如果手动的创建这么一个报文格式来发送给服务器想必是非常麻烦呢，于是浏览器为我们提供了一个XMLHttpRequest对象。

### XMLHttpRequest

* XMLHttpRequest对象是AJAX中非常重要的对象，所有的AJAX操作都是基于该对象的。
* XMLHttpRequest对象用来封装请求报文，我们向服务器发送的请求信息全部都需要封装到该对象中。
* 这里需要稍微注意一下，XMLHttpRequest对象并没有成为标准，但是现在的主流浏览器都支持该对象，而一些如IE6的老版本浏览器中的创建方式有一些区别，但是问题不大。
* 获取XMLHttpRequest对象：
  + 由于浏览器之间的差异，不同浏览器中获取XMLHttpRequest的方式不同，但总的来说一共有三种方式：
    - var xhr = new XMLHttpRequest()
      * 目前主流浏览器都支持
    - var xhr = new ActiveXObject("Msxml2.XMLHTTP")
      * IE6支持的方式
    - var xhr = new ActiveXObject("Microsoft.XMLHTTP")
      * IE5.5一下支持的方式
  + 根据三种不同的方式编写通用方法来获取XMLHttpRequest对象：

|  |
| --- |
| //获取XMLHttpRequest的通用方法  **function** getXMLHttpRequest(){  **var** xhr;  **try**{  //大部分浏览器都支持  xhr = **new** XMLHttpRequest();  }**catch**(e){  **try**{  //如果不支持，在这里捕获异常并且采用IE6支持的方式  xhr = **new** ActiveXObject("Msxml2.XMLHTTP");  }**catch**(e){  //如果还不支持，在这里捕获异常并采用IE5支持的方式  xhr = **new** ActiveXObject("Microsoft.XMLHTTP");  }  }  **return** xhr;  } |

* XMLHttpRequest对象的方法：
  + open(method,url,async)
    - open()用于设置请求的基本信息，接收三个参数。
      1. method
         * 请求的方法：get或post
         * 接收一个字符串
      2. url
         * 请求的地址，接收一个字符串
      3. Assync
         * 发送的请求是否为异步请求，接收一个布尔值。
         * true 是异步请求
         * false 不是异步请求（同步请求）
  + send(string)
    - send()用于将请求发送给服务器，可以接收一个参数
      1. string参数
         * 该参数只在发送post请求时需要。
         * string参数用于设置请求体
  + setRequestHeader(header,value)
    - 用于设置请求头
      1. header参数
         * 字符串类型，要设置的请求头的名字
      2. value参数
         * 字符串类型，要设置的请求头的值
* XMLHttpRequest对象的属性：
  + readyState
    - 描述XMLHttpRequest的状态
    - 一共有五种状态分别对应了五个数字：
      * 0 ：请求尚未初始化，open()尚未被调用
      * 1 ：服务器连接已建立，send()尚未被调用
      * 2 ：请求已接收，服务器尚未响应
      * 3 ：请求已处理，正在接收服务器发送的响应
      * 4 ：请求已处理完毕，且响应已就绪。
  + status
    - 请求的响应码
      * 200 响应成功
      * 404 页面为找到
      * 500 服务器内部错误

… … … …

* + onreadystatechange
    - 该属性需要指向一个函数
    - 该函数会在readyState属性发生改变时被调用
  + responseText
    - 获得字符串形式的响应数据。
  + responseXML（用的比较少）
    - 获得 XML 形式的响应数据。

## 示例代码

* 使用AJAX发送GET请求

|  |
| --- |
| //获取xhr对象  **var** xhr = getXMLHttpRequest();  //设置请求信息  xhr.open("get","AjaxServlet?&t="+Math.random(),**true**);  //发送请求  xhr.send();  //监听请求状态  xhr.onreadystatechange = **function**(){  //当响应完成  **if**(xhr.readyState == 4){  //且状态码为200时  **if**(xhr.status == 200){  //接收响应信息（文本形式）  **var** text = xhr.responseText;  //弹出消息  alert(text);  }  };  }; |

* + 这是一个最简单的AJAX代码，向AjaxServlet发送了一个get请求，并且在页面中输出响应的内容
* 使用AJAX发送POST请求

|  |
| --- |
| //获取xhr对象  **var** xhr = getXMLHttpRequest();  //设置请求信息  xhr.open("post","2.jsp",**true**);  //设置请求头  xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");  //发送请求  xhr.send("hello=123456");  //监听请求状态  xhr.onreadystatechange = **function**(){  //当响应完成  **if**(xhr.readyState == 4){  //且状态码为200时  **if**(xhr.status == 200){  //接收响应信息（文本形式）  **var** text = xhr.responseText;  //弹出消息  alert(text);  }  }; |

## JSON

* 上边两个示例返回都是一个字符串，那当我们需要返回一个复杂的数据时，比如说需要返回一个对象时，就需要定义一下数据的格式。
* AJAX一开始使用的时XML的数据格式，XML的数据格式非常简单清晰，容易编写，但是由于XML中包含了过多的标签，以及十分复杂的结构，解析起来也相对复杂，所以目前来讲，AJAX中已经几乎不使用XML来发送数据了。取而代之的是一项新的技术JSON。
* JSON是JavaScript Object Notation 的缩写，是JS提供的一种数据交换格式。
* JSON对象本质上就是一个JS对象，但是这个对象比较特殊，它可以直接转换为字符串，在不同语言中进行传递，通过工具又可以转换为其他语言中的对象。
* 例，有如下一个JSON对象：
  + {“name”:”sunwukong” , ”age”:18 , ”address”:”beijing” }
  + 这个对象中有三个属性name、age和address
  + 如果将该对象使用单引号引起了，那么他就变成了一个字符串
  + ‘{“name”:”sunwukong” , ”age”:18 , ”address”:”beijing” }’
  + 变成字符串后有一个好处，就是可以在不同语言之间传递。
  + 比如，将JSON作为一个字符串发送给Servlet，在Java中就可以把JSON字符串转换为一个Java对象。
* JSON通过6种数据类型来表示：
  + 字符串
    - 例子：”字符串”
    - 注意：不能使用单引号
  + 数字：
    - 例子：123.4
  + 布尔值：
    - 例子：true、false
  + null值:
    - 例子：null
  + 对象
    - 例子：{“name”:”sunwukong”, ”age”:18}
  + 数组
    - 例子：[1,”str”,true]
* 在JS中操作JSON
  + 创建JSON对象
    - var json = {“name1”:”value1”,”name2”:”value2” , “name3”:[1,”str”,true]};
    - var json = [{“name1”:”value1”},{“name2”:”value2”}];
  + JSON对象转换为JSON字符串
    - JSON.stringify(JSON对象)
  + JSON字符串转换为JSON对象
    - JSON.parse(JSON字符串)
* 在Java中操作JSON
  + 在Java中可以从文件中读取JSON字符串，也可以是客户端发送的JSON字符串，所以第一个问题，我们先来看如何将一个JSON字符串转换成一个Java对象。
  + 首先解析JSON字符串我们需要导入第三方的工具，目前主流的解析JSON的工具大概有三种json-lib、jackson、gson。三种解析工具相比较json-lib的使用复杂，且效率较差。而Jackson和gson解析效率较高。使用简单，这里我们以gson为例讲解。
  + Gson是Google公司出品的解析JSON工具，使用简单，解析性能好。
  + Gson中解析JSON的核心是Gson的类，解析操作都是通过该类实例进行。
  + 废话不多说直接上代码。
    - 解析外部文本文件中的JSON：
      1. JSON字符串转换为对象

|  |
| --- |
| String json = "{\"name\":\"张三\",\"age\":18}";  Gson gson = **new** Gson();  //转换为集合  Map<String,Object> stuMap = gson.fromJson(json, Map.**class**);  //如果编写了相应的类也可以转换为指定对象  Student fromJson = gson.fromJson(json, Student.**class**); |

2．对象转换为JSON字符串

|  |
| --- |
| Student stu = **new** Student("李四", 23);  Gson gson = **new** Gson();  //{"name":"李四","age":23}  String json = gson.toJson(stu);    Map<String , Object> map = **new** HashMap<String, Object>();  map.put("name", "孙悟空");  map.put("age", 30);  //{"age":30,"name":"孙悟空"}  String json2 = gson.toJson(map);    List<Student> list = **new** ArrayList<Student>();  list.add(**new** Student("八戒", 18));  list.add(**new** Student("沙僧", 28));  list.add(**new** Student("唐僧", 38));  //[{"name":"八戒","age":18},{"name":"沙僧","age":28},{"name":"唐僧","age":38}]  String json3 = gson.toJson(list); |