# Ajax技术简介

## 认识Ajax

每当用户向服务器端发出请求时，哪怕需要的只是简单地更新一点点的局部内容，服务器端都会将一个整体的页面进行刷新，并重新生成代码，这样一来程序的性能肯定会有所降低的。而如果采用Ajax技术，就可以实现局部的内容变更，从而使处理的性能要比前者高很多。因为采用的是局部刷新技术，所以整体页面并不会随着用户的每次请求而整体变化，只会在局部的位置上有所改变，这样的实现方式会使程序的性能更高。

1. Ajax（Asynchronous JavaScript and XML、异步JavaScript和XML，中文读作“阿贾克 斯”）的主要作用是实现页面的局部刷新，提升性能。
2. Ajax技术并不是JavaWeb独有的，各个动态Web实现技术如PHP、ASP都可以使用 Ajax技术。
3. Ajax技术是一门综合性的技术，包含了HTML、JavaScript、DOM、XML、XMLHttpRequest

等页面技术。其中最重要的是XMLHttpRequest对象。

1. Ajax技术也必须依靠服务器。如Tomcat。

## XMLHttpRequest对象

XMLHttpRequest对象Ajax的基础，在Ajax中主要是通过XMLHttpRequest对象处理发送 异步请求和回应的。

**1、创建XMLHttpRequest对象**

创建XMLHttpRequest对象时必须根据浏览器类型采用不同的方式创建 XMLHttpRequest对象，如果为火狐系列内核，则使用new XMLHttpRequest( )方式创建；而如果为低版本IE，则通过new ActiveXObject( )的方式创建。所以经过兼容处理后创建XMLHttpRequest对象的语法如下：



**2、XMLHttpRequest对象的属性**

在XMLHttpRequest对象中定义了许多属性，如下表所示：

|  |  |
| --- | --- |
| **属性名** | **描述** |
| onreadystatechange | 指定当readyState状态改变时的操作，一般都是指定一个具体的回调函数 |
| readyState | 返回当前的请求转态（只读） |
| responseBody | 将回应信息正文以unsigned byte数组形式返回（只读） |
| responseStream | 以Ado Stream对象的形式返回响应信息（只读） |
| responseText | 接收以普通文本返回的数据（只读） |
| responseXML | 接收以XML文档形式回应的数据（只读） |
| status | 返回当前请求的http状态码（只读） |
| statusText | 返回当前请求的响应行状态（只读） |

其中，readyState属性有5种取值，如下：

·0：请求没有发出（在调用open( )函数之前）；

·1：请求已经建立但还没有发出（在调用send( )函数之前）；

·2：请求已经发出正在处理之中（这时通常可以从响应得到内容头部）；

·3：响应已经处理，正在接受服务器的信息，响应中通常有部分数据可用，但是

服务器还没有完成响应；

·4：响应已完成，可以访问服务器响应并使用它。

**3、XMLHttpRequest对象的方法**

在使用XMLHttpRequest对象进行操作时也要使用到此对象中的方法。如下表所示：

|  |  |
| --- | --- |
| **方法** | **描述** |
| abort( ) | 取消当前所发出的请求 |
| getAllResponseHeader( ) | 取得所有的HTTP头信息 |
| getResponseHeader( ) | 取得一个指定的HTTP头信息 |
| open( ) | 创建一个HTTP请求，并指定请求方式，如GET或POST |
| send( ) | 将创建的请求发送到服务器端，并接受回应信息 |
| setRequestHeader( ) | 设置一个指定请求的HTTP头信息 |

XMLHttpRequest对象的open( )和send( )方法在回调函数中出现较多，一般都会用 open( )方法设置一个提交的路径，并通过地址重写的方式设置一些请求的参数，而 真正的发出请求操作可以通过send( )方法完成。

## 向服务器发送请求

使用XMLHttpRequest对象将请求发送到服务器，需要使用XMLHttpRequest对象的open( ) 和send( )方法。

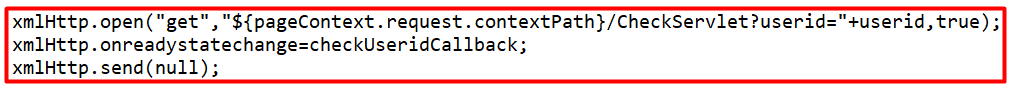
|  |  |
| --- | --- |
| **方法** | **描述** |
| open( method , url , async ) | 指定请求方式、URL以及是否异步处理请求；  method：指定请求类型，如get或post  url：可以是一个路径或一个文件  async：true（异步）或false（同步） |
| send( string ) | 向服务器发出请求  string参数仅用于post请求，且是需要传参时使用，如果是get请求或者post请求不需要传参时，则值设为null； |

1. **open( )方法的async参数**

open( )方法的async参数用于指定该请求时异步还是同步，当值为true时，表示请求是异步的，也就是执行完send( )方法后不等待服务器的执行结果，转而继续执行下面的脚本代码；当值为false时，表示请求时同步的，也就是执行完成send( )方法后等待服务器的执行结果的返回，若在等待过程中超时，则不再等待，继续执行后面的脚本代码。

1. **发出get请求**

发出get请求示例如下，直接在URL地址后传递请求参数：

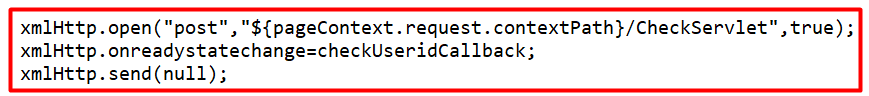


但需要注意，因为浏览器会缓存get请求，如果传递的参数是相同的，很可能得到 的是浏览器的缓存结果，为了避免这种情况，可以在URL添加一个动态变化的参 数，一般是当前时间：

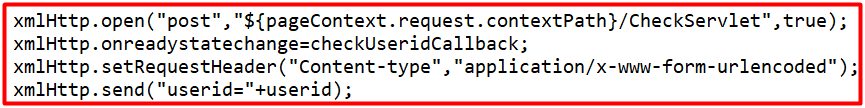


**3、发出post请求**

发出post请求如果不需要参数时，send( )方法参数设置为null：



如果需要传递参数，先设置请求头，再将send( )方法的参数设置为请求参数拼凑的 字符串：



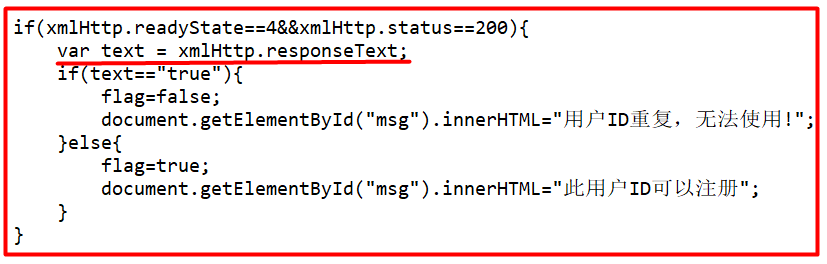
## 获取服务器响应数据

使用XMLHttpRequest对象的responseText和responseXML属性可以接收来自服务器的 响应数据。

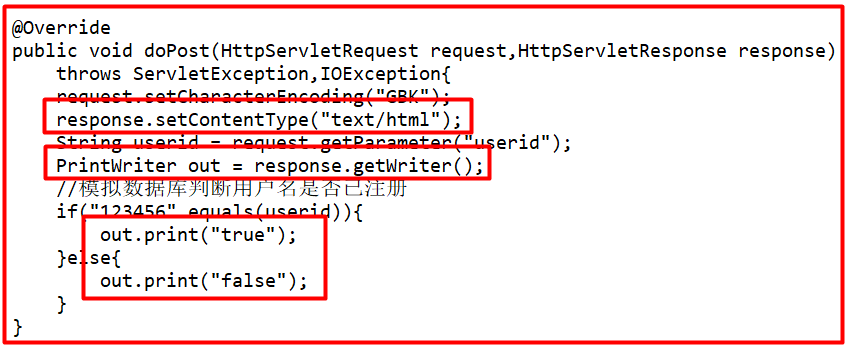
|  |  |
| --- | --- |
| **属性** | **描述** |
| responseText | 获得字符串形式的响应数据 |
| responseXML | 获得XML形式的响应数据 |

1. **使用responseText属性获得响应数据**

responseText属性是最常用的，它返回字符串形式的响应，可以直接渲染到HTML页面。



如果Ajax请求的是一个Servlet，该Servlet可以中可以设置响应类型并使用 PrintWriter向客户端进行响 应。如下：



**2、使用responseXML获得响应数据**

如果服务器端返回的数据是XML格式，而且需要作为XML对象进行解析，可以使 用responseXML属性。

# 使用Ajax

## 第一个Ajax程序

1. 编写返回数据的页面：content.html



该页面中没有任何的HTML元素的修饰。

1. 调用Ajax请求获取内容：ajax\_receive\_content.html

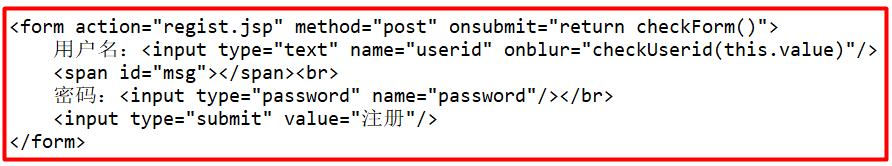


在本程序中，当单击按钮会调用showMsg( )函数，在showMsg( )函数中首先调用 createXMLttp( )函数创建XMLHttpRequest对象，通过open( )函数指定请求方式 和 路径，之后添加showMsgCallback( )函数对状态进行监听，然后调用send( )发送请 求。当状态改变时，调用showMsgCallback( )函数，通过readStatus==4和status==200 （请求状态码200表示请求成功）判断请求是否发送成功，如果发送成功，则通过 responseText接收返回的内容，并将其设置到id为msg的元素中进行显示。

以上Ajax程序本身非常简单，容易理解，且请求的是一个静态页面。

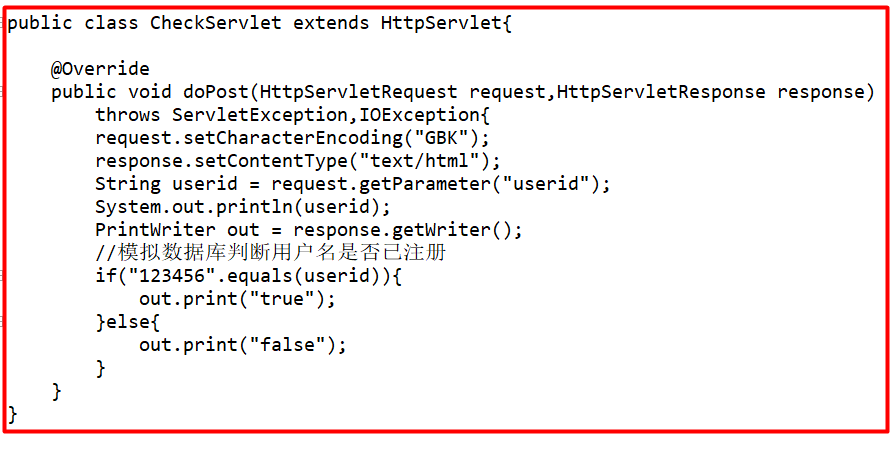
## 异步验证

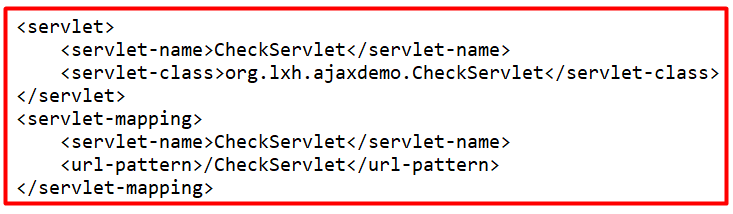
使用Ajax可以完成异步验证的功能。例如用户在注册账号时，当填写了用户名后， 其该表单失去焦点后，可以通过Ajax判断其输入的用户名是否已注册，并给出提示 信息，如果用户名已存在，则提供不能注册，如果不存在，则提示该用户名可以注册。 代码示例如下：

1. 注册页：regist.jsp



1. 验证用户名是否存在---CheckServlet.java





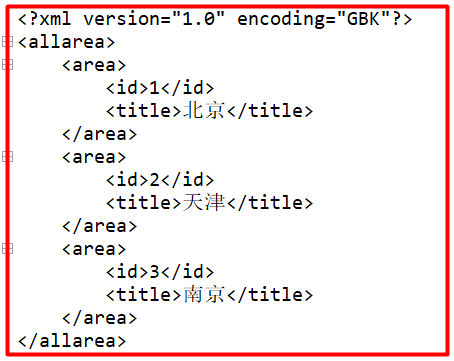
除此之外，Ajax常用的应用还有搜索框提示文字下拉列表、列表数据无刷新分 页、页面 上拉加载更多数据等。

## Ajax操作XML

在Ajax中，可以使用XMLHttpRequest对象中的responseXML属性读可以取XML文 件中的数据。

示例：将XML文件中的城市数据读取出来，生成下拉列表框。

1. 存在allarea.xml文件：



1. 使用Ajax解析XML，并生成下拉列表框：ajax\_select.html

