**a jax学习**

不是新的编程语言，而是一种使用现有标准的新方法

可以再不重新加载整个页面的情况下，与服务器交换数据并更新部分网页内容，不需要任何浏览器插件，但需要用户允许JavaScript在浏览器上执行

**应用：**

* 运用XHTML+CSS来表达资讯；
* 运用JavaScript操作DOM来执行动态效果
* 运用XML和XSLT操作资料
* 运用XMLHttpRequest或新的Fetch API与网页服务器进行异步资料交换

**A jax是基于现有的internet标准**

* XMLHttpRequest对象（异步的与服务器交换数据）
* JavaScript/DOM(信息显示/交互）
* CSS（给数据定义样式）
* XML（作为转换数据的格式）

**注意**： a jax应用程序与浏览器和平台无关的

实例：

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<script>

function loadXMLDoc()

{

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");

}

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();

}

</script>

</head>

<body>

​

<div id="myDiv"><h2>使用 AJAX 修改该文本内容</h2></div>

<button type="button" onclick="loadXMLDoc()">修改内容</button>

​

</body>

</html>

实例解析：

上面的A JAX应用程序包含一个div和一个按钮

div部分用于显示来自服务器的信息。当按钮被点击时，它负责调用名为loadXMLDoc()的函数：

<div id="myDiv"><h2>使用 AJAX 修改该文本内容</h2></div>

<button type="button" onclick="loadXMLDoc()">修改内容</button>

接下来，在页面的head部分添加一个<script>标签，该标签中包含了这个loadXMLDoc()函数：

<head>

<script>

function loadXMLDoc() {

.... AJAX 脚本执行 ...

}

</script>

</head>

**创建XMLHttpRequest对象**

XMLHttpRequest是A JAX的基础

XMLHttpRequest：所有现代浏览器均支持XMLHttpRequest对象，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。

**创建XMLHttpRequest对象的语法：**

*variable*=new XMLHttpRequest();

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

*variable*=new ActiveXObject("Microsoft.XMLHTTP");

实例：

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<script>

function loadXMLDoc()

{

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");

}

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();

}

</script>

</head>

<body>

​

<div id="myDiv"><h2>使用 AJAX 修改该文本内容</h2></div>

<button type="button" onclick="loadXMLDoc()">修改内容</button>

​

</body>

</html>

**向服务器发送请求**

XMLHttpRequest对象用于和服务器交换数据

如需将请求发送到服务器，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 |

**GET还是POST？**

与POST相比，GET更简单也更快，并且在大部分情况下都能用

然而，在以下情况中，请使用POST请求：

* 无法使用缓存文件（更新服务器上的文件或数据库）
* 向服务器发送大量数据（POST没有数量限制）
* 发送包含未知字符的用户输入时，POST比GET更稳定也更可靠

**GET请求**

一个简单的GET请求：

实例：

xmlhttp.open("GET","/try/ajax/demo\_get.php",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();

如果希望通过GET方法发送信息，请向URL添加信息：

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

xmlhttp.send();

**POST请求**

一个简单POST请求：

xmlhttp.open("POST","/try/ajax/demo\_post.php",true);

xmlhttp.send();

如果需要像HTML表单那样POST数据，请使用setRequestHeader()来添加HTTP头。然后在send（）方法中规定希望发送的数据：

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

|  |  |
| --- | --- |
| 方法 | 描述 |
| setRequestHeader(header,value) | 向请求添加HTTP头   * Header:规定头的名称 * Value：规定头的值 |

**URL：服务器上的文件**

open（）方法的url参数是服务器上文件的地址：

xmlhttp.open("GET","ajax\_test.html",true);

该文件可以是任何类型的文件，比如.txt和.xml，或者服务器脚本文件，比如.asp和.php（在传回响应之前，能够在服务器上执行任务）

**异步：true或false**

A JAX指的是异步JavaScript和XML。XMLHttpRequest对象如果要用于A JAX的话，其open（）方法的async参数必须设置为true：

xmlhttp.open("GET","ajax\_test.html",true);

通过A JAX，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
* 在响应就绪后对响应进行处理

**Async=true**

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

实例：

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();

**Async=false**

如果需要使用async=false，请将open（）方法中的第三个参数改为false：

xmlhttp.open("GET","test1.txt",false)；

**注意：**当使用async=false时，请不要编写onreadystatechange函数，把代码放到send（）语句后面即可：

xmlhttp.open("GET","/try/ajax/ajax\_info.txt",false);

xmlhttp.send();

document.getElementById("myDiv").innerHTML=xmlhttp.responseText;

**服务器响应**

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 |

responseText属性

如果来自服务器的响应并非XML，请使用responseText属性

responseText属性返回字符串形式的响应，因此可以这样使用：

document.getElementById("myDiv").innerHTML=xmlhttp.responseText;

## responseXML 属性

如果来自服务器的响应时XML，而且需要作为XML对象进行解析，请使用responseXML属性：

实例：

请求cd\_catalog.xml文件，并解析响应：

xmlDoc=xmlhttp.responseXML;

txt="";

x=xmlDoc.getElementsByTagName("ARTIST");

for (i=0;i<x.length;i++)

{

txt=txt + x[i].childNodes[0].nodeValue + "<br>";

}

document.getElementById("myDiv").innerHTML=txt;

## onreadystatechange 事件

当请求被发送到服务器时，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0-4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 |

在onreadystatechange事件中，规定当服务器响应已做好被处理的准备时所执行的任务。

当readyState等于4且状态为200时，表示响应已就绪：

实例：

xmlhttp.onreadystatechange=function()

{

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

{

document.getElementById("myDiv").innerHTML=xmlhttp.responseText;

}

}

**注意**：onreadystatechange事件被触发4次（0-4），分别是0-1,1-2,2-3,3-4，对应着readtState的每个变化

**使用回调函数**

回调函数是一种以参数形式传递给另一个函数的函数

如果网站上存在多个A JAX任务，那么应该为创建XMLHttpRequest对象编写一个标准的函数，并为每个A JAX任务调用该函数

该函数调用应该包含URL以及发生onreadystatechange事件时执行的任务（每次调用可能不尽相同）

实例：

function myFunction()

{

loadXMLDoc("/try/ajax/ajax\_info.txt",function()

{

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

document.getElementById("myDiv").innerHTML=xmlhttp.responseText;

}

});

}

A JAX Database实例

可以用来与数据库进行动态通信

实例：网页如何通过A JAX从数据库读取信息

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<script>

function showCustomer(str)

{

var xmlhttp;

if (str=="")

{

document.getElementById("txtHint").innerHTML="";

return;

}

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");

}

xmlhttp.onreadystatechange=function()

{

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

{

document.getElementById("txtHint").innerHTML=xmlhttp.responseText;

}

}

xmlhttp.open("GET","/try/ajax/getcustomer.php?q="+str,true);

xmlhttp.send();

}

</script>

</head>

<body>

​

<form action="">

<select name="customers" onchange="showCustomer(this.value)" style="font-family:Verdana, Arial, Helvetica, sans-serif;">

<option value="APPLE">Apple Computer, Inc.</option>

<option value="BAIDU ">BAIDU, Inc</option>

<option value="Canon">Canon USA, Inc.</option>

<option value="Google">Google, Inc.</option>

<option value="Nokia">Nokia Corporation</option>

<option value="SONY">Sony Corporation of America</option>

</select>

</form>

<br>

<div id="txtHint">客户信息将显示在这...</div>

​

</body>

</html>

**showCustomer()函数执行以下任务：**

* 检查是否已选择某个客户
* 创建XMLHttpRequest对象
* 当服务器响应就绪时执行所被创建的函数
* 把请求发送到服务器上的文件
* 请注意向URL添加了一个参数q(带有输入域中的内容）

**A JAX服务器页面**

上面的JavaScript调用的服务器页面时PHP文件，名为"getcustomer.php"。

用PHP编写服务器文件也很容易，或者用其他服务器语言。"getcustomer.php"中的源代码负责对数据库进行查询，然后用HTML表格返回结果：

<%

response.expires=-1

sql="SELECT \* FROM CUSTOMERS WHERE CUSTOMERID="

sql=sql & "'" & request.querystring("q") & "'"

set conn=Server.CreateObject("ADODB.Connection")

conn.Provider="Microsoft.Jet.OLEDB.4.0"

conn.Open(Server.Mappath("/db/northwind.mdb"))

set rs=Server.CreateObject("ADODB.recordset")

rs.Open sql,conn

response.write("<table>")

do until rs.EOF

for each x in rs.Fields

response.write("<tr><td><b>" & x.name & "</b></td>")

response.write("<td>" & x.value & "</td></tr>")

next

rs.MoveNext

loop

response.write("</table>")

%>

**XML实例**

可以用来与XML文件进行交互式通信

实例：

下面的例子将演示网页如何使用A JAX来读取来自XML文件的信息：

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<style>

table,th,td {

border : 1px solid black;

border-collapse: collapse;

}

th,td {

padding: 5px;

}

</style>

</head>

<body>

​

<h1>XMLHttpRequest 对象</h1>

​

<button type="button" onclick="loadXMLDoc()">获取我收藏的 CD</button>

<br><br>

<table id="demo"></table>

​

<script>

function loadXMLDoc() {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

myFunction(this);

}

};

xhttp.open("GET", "cd\_catalog.xml", true);

xhttp.send();

}

function myFunction(xml) {

var i;

var xmlDoc = xml.responseXML;

var table="<tr><th>Artist</th><th>Title</th></tr>";

var x = xmlDoc.getElementsByTagName("CD");

for (i = 0; i <x.length; i++) {

table += "<tr><td>" +

x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +

"</td><td>" +

x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +

"</td></tr>";

}

document.getElementById("demo").innerHTML = table;

}

</script>

​

</body>

</html>

实例解析：loadXMLDoc()函数

当用户点击上面的“获取我收藏的CD”这个按钮，就会执行loadXMLDoc（）函数。LoadXMLDoc()函数创建XMLHttpRequest对象，添加当服务器响应就绪时执行的函数，并将请求发送到服务器。当服务器响应就绪时，会构建一个HTML表格，从XML文件中提取节点（元素），最后使用XML数据的填充id=”demo”的表格元素：

**异步加载XML文档**

function loadXMLDoc() {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

myFunction(this);

}

};

xhttp.open("GET", "cd\_catalog.xml", true);

xhttp.send();

}

function myFunction(xml) {

var i;

var xmlDoc = xml.responseXML;

var table="<tr><th>Artist</th><th>Title</th></tr>";

var x = xmlDoc.getElementsByTagName("CD");

for (i = 0; i <x.length; i++) {

table += "<tr><td>" +

x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +

"</td><td>" +

x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +

"</td></tr>";

}

document.getElementById("demo").innerHTML = table;

}

**这个例子中使用的服务器页面实际上是一个名为“cd\_catalog.xml”XML文件**