返回目录

第八章　XML与AJAX

本章导读

HTML与CSS适合呈现页面效果，不长于数据组织，而这正是XML的强项。XML类似HTML，但标记可以根据需要自定义，而HTML标记有规范约束。

XML数据可以很好地与网页融合，其常用技术是AJAX。AJAX的出现不仅用于XML数据与网页的融合，更重要的是体现新的网页数据交互数据。

学习目标：

1. 认识XML；
2. 了解XML数据与网页融合技术；
3. 认识AJAX；
4. 掌握AJAX技术；

本章目录

第一节　XML

1、XML简介

XML(eXtensible Markup Language，可扩展标记语言)与HTML(HyperText Markup Language)都有共同的ML两个字符，也都是Markup Language的简写，预示二者存在联系。观察例程8-1就会发现，其代码真有某些相似，元素名称开始和元素结束放在一对尖括号中，结束时名称有斜杠(/)，元素可以有属性，也可以没有属性。和很多HTML元素一样，元素内还可以包含子元素，在本例中，student元素包含有name、sex、native元素。但是，和HTML有很大的不同，XML的元素名称似乎都没有在HTML出现过，看起来是自由命名。确实如此，XML的标记(元素)可以根据需要扩展，只要这种标记集合能为使用者所认可。如果程序由一个程序员完成，只要在其开发范围内获得认可即可；如果由多个程序员完成就需要在程序员之间约定标记集合形成规范，使不同程序员开发的程序能互相交换数据；当需要在更大范围内交换数据，则需要形成公司规范、行业标准、国家标准乃至国际标准。当前已有多种国际标准，比如：描述数学公式的标准、矢量图形的标准、电子商务标准等。

例程8-1

```
第1行  <?xml version="1.0" encoding="gb2312"?>
第2行  <root>
第3行      <student id="1100012010">
第4行          <name>灭绝师太</name>
第5行          <sex>女</sex>
第6行          <native>峨眉山</native>
第7行      </student>
第8行      <student id="1100012011">
第9行          <name>张三丰</name>
第10行         <sex>男</sex>
第11行         <native>武当山</native>
第12行     </student>
```

```
第13行                <student id="1100012110">
第14行                    <name>黄药师</name>
第15行                    <sex>男</sex>
第16行                    <native>桃花岛</native>
第17行                </student>
第18行            </root>
```

和HTML不同，XML对大小写是敏感，每个元素都必须有开始和结束标记，其开始和结束标记大小写必须匹配。对于没有内容的XML元素，可以写成形如<native></native>，亦或<native/>，选择后一种更好。XML元素的属性值必须用半角引号包围，而HTML的属性值是否用引号属于可选。XML全部数据必须包含在有且仅有一个根元素内，如文档root元素。

XML文档(数据)第一行通常需要申明该文档(或数据)所采用XML版本号及其编码。在例程8-1中，设定该XML文档版本号为1.0，其编码为gb2312。常用编码标准还有utf-8等。

2、读取XML数据

XML数据可以读取融进网页中，如例程8-2所示。

例程8-2

```
第1行   <HTML>
第2行       <HEAD>
第3行           <TITLE>读取XML数据</TITLE>
第4行           <Script type="Text/JavaScript" language="JavaScript" src="./jquery-1.9.1.min.js"></Script
    >
第5行           <style type=text/css>
第6行               #showData table{width:500px;}
第7行               #showData td{height:30px;line-height:28px;border-bottom:1px dotted gray;text-alig
    n:Center}
第8行               #showData th{height:40px;background-color:#DFDFDF;border-bottom:2px solid black}
第9行           </style>
第10行      </HEAD>
第11行      <BODY>
第12行          <Div id="showData"></Div>
第13行          <SCRIPT LANGUAGE="JavaScript">
第14行              $.get("xmlData.xml",function(xml){
第15行                  var tabHeader="<table cellpadding='0' cellspacing='0'>";
第16行                  var tabFooter="</table>";
第17行                  var tabRow="<tr><th>学号</th><th>姓名</th><th>籍贯</th>";
第18行                  $(xml).find("student").each(function(){
第19行                      tabRow+="<tr>";
第20行                      tabRow+="<td>"+$(this).attr("id")+"</td>";
第21行                      tabRow+="<td>"+$(this).children("name").text()+"</td>";
第22行                      tabRow+="<td>"+$(this).children("native").text()+"</td>";
第23行                      tabRow+="</tr>";
第24行                  });
第25行                  $("#showData").html(tabHeader+tabRow+tabFooter);
第26行              });
第27行          </SCRIPT>
第28行      </BODY>
```

```
第29行    </HTML>
```

学号	姓名	籍贯
1100012010	灭绝师太	峨眉山
1100012011	张三丰	武当山
1100012110	黄药师	桃花岛

图8-1　例程8-2执行效果图

例程8-2第14行代码$.get("xmlData.xml",function(xml){})的功能是获取xmlData.xml中的数据，并交由函数function(xml)去处理，其中xmlData.xml是URL，可以是远程Web地址中的数据，并常常是如此体现，此处为获得当前网页所在文件夹下的xmlData.xml数据。function(xml)中的xml并不是关键词，相当于一个参数，代表获得的数据，可以更改为其他名称如data，xmlData等，最好用含xml的名称，以表明获得的数据是XML格式。$.get()实为AJAX的一个应用，将在本章稍候讲述，此处暂可理解为一个名为$.get()的函数，可以打开XML数据，并交由指定函数处理。

第18行是处理XML数据的关键。$(xml)的含义是将参数xml所代表的数据jQuery化。一旦jQuery化，就可以使用jQuery相关操作，在这里应用了find()查找子元素，通过each()遍历所有元素并显示在页面上。

如果数据很多，没有必要全部显示，可以通过输入id号查找，并显示相关信息。如例程8-3所示。

例程8-3

```
第1行    <HTML>
第2行        <HEAD>
第3行            <TITLE>读取XML数据</TITLE>
第4行            <Script type="Text/JavaScript" language="JavaScript" src="./jquery-1.9.1.min.js"></Script
>
第5行            <style type=text/css>
第6行                #showData table{width:500px;}
第7行                #showData td{height:30px;line-height:28px;border-bottom:1px dotted gray;text-alig
n:Center}
第8行                #showData th{height:40px;background-color:#DFDFDF;border-bottom:2px solid black}
第9行            </style>
第10行       </HEAD>
第11行       <BODY>
第12行           请输入id号:<BR><input type=text id="inputID"><button id="btnOK">确定</button><BR>
第13行           <Div id="showData"></Div>
第14行           <SCRIPT LANGUAGE="JavaScript">
第15行               var myXMLData=null;//用于保存xml数据，初始为null。
第16行               $.get("xmlData.xml",function(xml){
第17行                   myXMLData=xml;//将获得的XML数据赋值给myXMLData;
第18行               });
第19行               $("#btnOK").click(function(){
第20行                   var inputID=$("#inputID").val();
第21行                   var findState=$(myXMLData).find("student[id='"+inputID+"']");
第22行                   if(findState.size()==0)
第23行                       alert("查无此人");
第24行                   else{
第25行                       var tabHeader="<table cellpadding='0' cellspacing='0'>";
第26行                       var tabFooter="</table>";
第27行                       var tabRow="<tr><th>学号</th><th>姓名</th><th>籍贯</th></tr>";
```

```
第28行                                    tabRow+="<tr>";
第29行                                    tabRow+="<td>"+findState.attr("id")+"</td>";
第30行                                    tabRow+="<td>"+findState.children("name").text()+"</td>";
第31行                                    tabRow+="<td>"+findState.children("native").text()+"</td>";
第32行                                    tabRow+="</tr>";
第33行                                    $("#showData").html(tabHeader+tabRow+tabFooter);
第34行                                }
第35行                            });
第36行                    </SCRIPT>
第37行            </BODY>
第38行    </HTML>
```

请输入id号:
1100012011 确定

学号	姓名	籍贯
1100012011	张三丰	武当山

图8-2　例程8-3执行效果图

例程8-3的代码var findState=$(myXMLData).find("student[id='"+inputID+"']")用于查找是否存在指定的编号的数据，其中find()部分与jQuery完全相同。当找到数据则显示相关数据，否则显示“查无此人”。在上述代码中，$(myXMLData)不能用xml代替myXMLData，因为xml在第17行执行完毕已经消失，所以在程序中设置一个全局变量，用于保存参数xml所代表的数据。当单击“确定”按钮时，根据输入内容访问myXMLData数据。如果将click事件绑定代码置于$.get()中，则无须申明全局变量，此时xml参数对btnOK的click事件适用，如例程8-4所示。

例程8-4

```
第1行    <HTML>
第2行            <HEAD>
第3行                    <TITLE>读取XML数据</TITLE>
第4行                    <Script type="Text/JavaScript" language="JavaScript" src="./jquery-1.9.1.min.js"></Script
         >
第5行                    <style type=text/css>
第6行                            #showData table{width:500px;}
第7行                            #showData td{height:30px;line-height:28px;border-bottom:1px dotted gray;text-alig
         n:Center}
第8行                            #showData th{height:40px;background-color:#DFDFDF;border-bottom:2px solid black}
第9行                    </style>
第10行           </HEAD>
第11行           <BODY>
第12行                   请输入id号:<BR><input type=text id="inputID"><button id="btnOK">确定</button><BR>
第13行                   <Div id="showData"></Div>
第14行                   <SCRIPT LANGUAGE="JavaScript">
第15行                           $.get("xmlData.xml",function(xml){
第16行                                   $("#btnOK").click(function(){
第17行                                           var inputID=$("#inputID").val();
第18行                                           var findState=$(xml).find("student[id='"+inputID+"']");
第19行                                           if(findState.size()==0)
第20行                                                   alert("查无此人");
```

```
第21行                                    else{
第22行                                        var tabHeader="<table cellpadding='0' cellspacing='0'>";
第23行                                        var tabFooter="</table>";
第24行                                        var tabRow="<tr><th>学号</th><th>姓名</th><th>籍贯</th></
    tr>";
第25行                                        tabRow+="<tr>";
第26行                                        tabRow+="<td>"+findState.attr("id")+"</td>";
第27行                                        tabRow+="<td>"+findState.children("name").text()+"</td>";
第28行                                        tabRow+="<td>"+findState.children("native").text()+"</td
    >";
第29行                                        tabRow+="</tr>";
第30行                                        $("#showData").html(tabHeader+tabRow+tabFooter);
第31行                                    }
第32行                                });
第33行                            });
第34行                        </SCRIPT>
第35行                </BODY>
第36行    </HTML>
```

综上所述，XML作为一种有效的数据存储管理方式，能很好地与网页融合，被jQuery方便操作。

第二节　了解AJAX

AJAX的英文含义是“Asynchronous JavaScript and XML”(异步JavaScript和XML)。AJAX不是一种新的编程语言，而是1998年前后逐步得到广泛认可而推广的一种技术应用，是一种使用现有标准的新方法，是一种不重新加载整个页面而与服务器交换数据并更新部分网页的艺术。AJAX能创建更好更快交互性更强的Web应用。

网页与服务器交换数据的传统方式是在网页输入数据，单击按钮后转入一个新的页面，这种方式现在也还在大量使用。如图8-3所示的网站简单注册界面。在传统方式下，输入帐号和密码后，单击确定即转入一个新的页面。如果该帐号已经被注册，则提示用户重新输入帐号，回到原注册界面。这种用户体验不好。如果能在输入帐号后，自动判断帐号是否可用，并显示提示信息且不离开原页面，用户体验将大为提升。自动判断帐号时，需要将用户输入的帐号传输到远程后台服务器，服务器经过处理后，返回处理结果，页面程序根据返回信息处理将提示信息显示在页面上。这种不跳转更新页面且与远程后台服务器交换数据的技术就是AJAX的重要特征之一，这种技术现在得到广泛应用。

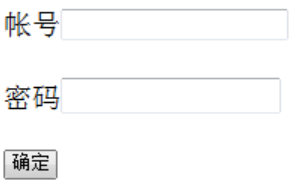

图8-3　网站简单注册界面

网站注册页面不会简单到两个文本框和一个按钮，肯定还包含其他内容，比如：网站标志、其他说明性文字等等诸多内容。如果采用传统方式，单击按钮转向一个新页面，这些内容都将被重复传输，虽然仅仅是确认“帐号”是否已被使用，信息量非常有限，但是也不得不传输大量信息。因此，采用AJAX技术，将减少请求服务器次数，降低服务器负载；降低网络传输数据量，提升网速；增强用户体验等。

当前主流浏览器都支持AJAX技术，虽然互相之间略有区别，但通过jQuery封装，AJAX变得更加易用。

为了能对AJAX有所了解，还是从例程8-2第14行代码的$.get("xmlData.xml",function(xml){})说起，该行代码的格式如下：

```
$.get(url,[data],[function(){}])
```

其中URL是数据来源的地址，[data]和[function(){}]是可选参数。假定远程服务器上有一个程序名为getData.asp，其位于“http://xxx.xxx.xxx.xxx/student/”，则其URL为“http://xxx.xxx.xxx.xxx/student/getData.asp”，其中xxx.xxx.xxx.xxx为域名或IP地址。URL参数也可以是网页所在位置的数据或者程序，例程8-2即为同在网页文件所在位置的xmlData.xml文件。

[data]以key/value值对形式出现，即匿名对象，如$.get("http://xxx.xxx.xxx.xxx/student/getData.asp",{id:"1100012011"})，其id的值为1100012011，该key/value和URL一起传递到目标位置，当服务器收到该请求后，会根据程序(getData.asp)和id值处理该请求，并返回处理结果。[data]可以认为是程序(getData.asp)的参数，如同函数有参数一样。程序的参数还有等同的写法，如：“http://xxx.xxx.xxx.xxx/student/getData.asp?id=1100012011”，实际上jQuery在向服务器发起情切时，也会将[data]转换为上述格式，不过采用[data]模式更加灵活，使用更加方便。

[function(){}]用于处理服务器执行后返回的结果。如果仅仅是向服务器提交数据，而不需要处理返回结果，该部分可以省略。$.get()会自动根据返回结果判断其格式(xml、html、json、 script)。例程8-5是返回页面数据的示例。

例程8-5

```
第1行  <HTML>
第2行      <HEAD>
第3行          <TITLE>读取网页数据</TITLE>
第4行          <Script type="Text/JavaScript" language="JavaScript" src="./jquery-1.9.1.min.js"></Script
       >
第5行      </HEAD>
第6行      <BODY>
第7行          <Div ID="htmlContainer" style="width:600px"></Div>
第8行          <SCRIPT LANGUAGE="JavaScript">
第9行              $.get("http://127.0.0.1/zzz.html",function(data){
第10行                 $("#htmlContainer").html(data);
第11行             });
第12行         </SCRIPT>
第13行     </BODY>
第14行 </HTML>
```

在例程8-5第9行代码表示取得127.0.0.1下的zzz.html的内容，并将内容交付function(data)处理，$("#htmlContainer").html(data)代码表示将取得的数据显示在htmlContainer中。zzz.html的代码如例程8-6所示。由此可见，$.get()可以会自动根据内容判断数据格式。

例程8-6

```
第1行  <HTML>
第2行      <Head>
第3行          <Meta http-equiv="Content-Type" content="text/html; charset=GB2312"/>
第4行      </Head>
第5行      <BODY>
第6行          AJAX的英文含义是“Asynchronous JavaScript and XML”(异步JavaScript和XML)。
第7行          AJAX不是一种新的编程语言，而是1998年前后逐步得到广泛认可而推广的一
第8行          种技术应用，是一种使用现有标准的新方法，是一种不重新加载整个页面而
第9行          与服务器交换数据并更新部分网页的艺术。AJAX能创建更好更快交互性更强
第10行         的Web应用。
第11行     </BODY>
第12行 </HTML>
```

例程8-7是一个运行在服务器端的程序，其功能是查询邮政编码对应的地名(此处数据仅为示例)。由于有较多注释，理解可能不存在困难。例程8-8是其应用，图8-4是其执行效果。

例程8-7

```
第1行  <%@ Page language="vb"%>
第2行  <%
第3行      Dim mySeek as string=Request.QueryString("postcode")'取得地址栏输入的postcode值
第4行
第5行      Dim myData as String
第6行      myData="110101东城,110102西城,110103崇文,110104宣武,110105朝阳,110106丰台,"
第7行      myData+="110107石景山,110108海淀,110109门头沟,110111房山,110112通州,110113顺义,110114昌平,"
第8行      myData+="110115大兴,110116怀柔,110117平谷,110228密云,110229延庆," 'rem myData为字符串
第9行
第10行     Dim posCode as integer=InStr(myData,mySeek)'计算邮政编码出现位置,instr计算mySeek在myData中的位置
第11行     If posCode>0 Then '如果存在该邮政编码
第12行         Dim posComma as integer=InStr(posCode+6,myData,",")'查找邮政编码后逗号出现的位置
第13行         Dim areaName as String '申明areaName为一个字符串型变量,区域名称
第14行         areaName=Mid(myData,posCode+6,posComma-posCode-6)'取出邮政编码逗号前的字符内容
第15行         response.write(areaName)'response.write表示输出，此处输出areaName变量的内容
第16行     Else'如果不存在输入的邮政编码
第17行         response.write("没有该编码!")
第18行     End if
第19行 %>
```

例程8-8

```
第1行  <HTML>
第2行      <HEAD>
第3行          <TITLE>邮政编码与地名</TITLE>
第4行          <Script type="Text/JavaScript" language="JavaScript" src="./jquery-1.9.1.min.js"></Script
       >
第5行      </HEAD>
第6行      <BODY>
第7行          请输入邮政编码:<BR><input type=text><button>确定</button><BR><BR>
第8行          <Div id="seekResult"></Div>
第9行          <SCRIPT LANGUAGE="JavaScript">
第10行             $("button").click(function(){
第11行                 var postCode=$("input").val();
第12行                 $.get("http://127.0.0.1/getName.aspx",{postcode:postCode},function(data){
第13行                     $("#seekResult").html(postCode+"对应的地名是"+data);
第14行                 });
第15行             });
第16行         </SCRIPT>
第17行     </BODY>
第18行 </HTML>
```

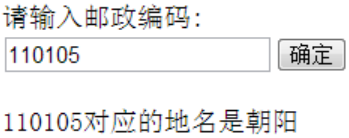

图8-4　网站简单注册界面

当输入错误代码时，例程8-8将显示：“110122对应的地名是没有该编码!”，让人莫名其妙，可以修改如例程8-9所示(仅是例程8-8的JavaScript替换)。另外，URL改变为http://127.0.0.1/getName.aspx?postcode="+postCode，其效果与例程8-8完全相同。

例程8-9

```
<SCRIPT LANGUAGE="JavaScript">
        $("button").click(function(){
                var postCode=$("input").val();
                $.get("http://127.0.0.1/getName.aspx?postcode="+postCode,function(data){
                        if(data=="没有该编码!"){
                                $("#seekResult").html(postCode+"对应地名不存在，请核实邮政编码是否正确！");
                        }else{
                                $("#seekResult").html(postCode+"对应的地名是"+data);
                        }
                });
        });
</SCRIPT>
```

可以让服务器返回XML格式的文档，修改getName.aspx为getNameXML.aspx，如例程8-10，对应的网页代码也修改为例程8-11。

例程8-10

```
<%@ Page language="vb"%>
<%
        Dim mySeek as string=Request.QueryString("postcode")'取得地址栏输入的postcode值

        Dim myData as String
        myData="110101东城,110102西城,110103崇文,110104宣武,110105朝阳,110106丰台,"
        myData+="110107石景山,110108海淀,110109门头沟,110111房山,110112通州,110113顺义,110114昌平,"
        myData+="110115大兴,110116怀柔,110117平谷,110228密云,110229延庆,"  'rem myData为字符串

        Dim posCode as integer=InStr(myData,mySeek)'计算邮政编码出现位置,instr计算mySeek在myData中的位置
        Dim seekState as boolean'保存查询状态
        Dim areaName as String '申明areaName为一个字符串型变量,区域名称
        If posCode>0 Then '如果存在该邮政编码
                seekState=true
                Dim posComma as integer=InStr(posCode+6,myData,",")'查找邮政编码后逗号出现的位置
                areaName=Mid(myData,posCode+6,posComma-posCode-6)'取出邮政编码逗号前的字符内容
        Else'如果不存在输入的邮政编码
                seekState=false
        End If
        Response.ContentType="text/xml" '数据流格式定义
        response.write("<?xml version='1.0' encoding='GB2312'?>")
        response.write("<Root>")
        response.write("<postcode>"+mySeek+"</postcode>")
        If seekState=True Then
                response.write("<state>1</state>")'找到对应地名
                response.write("<areaname>"+areaName+"</areaname>")
```

```
第27行          Else
第28行                  response.write("<state>0</state>")'未找到对应地名
第29行          End If
第30行          response.write("</Root>")
第31行  %>
```

例程8-11

```
第1行   <SCRIPT LANGUAGE="JavaScript">
第2行           $("button").click(function(){
第3行                   var postCode=$("input").val();
第4行                   $.get("http://127.0.0.1/getNameXML.aspx",{postcode:postCode},function(data){
第5行                           var rtnXML=$(data);
第6行                           var rtnState=rtnXML.find("state").text();
第7行                           if(rtnState==0){
第8行                                   $("#seekResult").html(postCode+"没有对应地名，请核实输入是否正确!");
第9行                           }else{
第10行                                  $("#seekResult").html(postCode+"对应地名是"+rtnXML.find("areaname").text
    ());
第11行                          }
第12行                  });
第13行          });
第14行  </SCRIPT>
```

上述程序实现了从网页将数据提交到服务器，服务器可以接受网页输入，经过处理后服务器返回处理结果，另外，服务器还可以存储数据以供长期使用。另外，大规模的数据，通常保存在数据库中，如mySQL、SQL Server、Oracle等。相关内容，请参考其他相关文献。

为了降低网络流量，如果对同一网页发出相同请求，浏览器将把缓存中的数据作为结果，这可能导致误判，即服务器数据可能已经修改，但由于请求相同而不能更新，此时可以在URL中增加时间项，由于时间总是在流逝，因此每次请求都不能相同，迫使浏览器从服务器取得最新数据。如：$.get("http://127.0.0.1/getNameXML.aspx",{postcode:postCode,timer:new Date()})，new Date()每次都取得计算机以毫秒为单位的时间，因此总是能保持更新，浏览器也就每次都回到远程服务器端取得数据。

第三节　jQuery中的AJAX

jQuery让AJAX的使用变得非常简单。AJAX的使用大致可以分为AJAX请求(用于从网页向服务器发送数据或者从服务器获得数据)、AJAX事件(AJAX请求的执行有多种状态，如：启动、停止、结束、成功和错误等)。

1、AJAX请求

AJAX请求分为：$.get()、$.post()、$.getJSON()、$.getScript()、$.load()以及$.ajax()。$.get()前面已经讲述，$.ajax()实际上$.get()、$.post()等AJAX请求的底层实现，$.get()、$.post()等是$.ajax()的简单应用。

(1)　$.post()

$.post()与$.get()的语法格式完全相同，其使用方式也相同，但是其数据交互机制则有很大的区别。网页与远程服务器交互数据的方式可以分为：get和post模式。当采用get模式时，数据跟随在URL中，其提交的数据量比较有限，且采用明文方式提交，安全性较低；而采用post模式，其数据跟随http协议，可以提交大量数据，安全性也较高。服务器端程序处理方式也不同。例程8-12是$.post()的使用，例程8-13是服务器端相应处理代码。

例程8-12

```
第1行   <SCRIPT LANGUAGE="JavaScript">
第2行           $("button").click(function(){
第3行                   var postCode=$("input").val();
```

```
第4行                $.post("http://127.0.0.1/getNameXML_Post.aspx",{postcode:postCode,timer:new Date()},funct
ion(data){
第5行                    var rtnXML=$(data);
第6行                    var rtnState=rtnXML.find("state").text();
第7行                    if(rtnState==0){
第8行                        $("#seekResult").html(postCode+"没有对应地名，请核实输入是否正确!");
第9行                    }else{
第10行                       $("#seekResult").html(postCode+"对应地名是"+rtnXML.find("areaname").text
());
第11行                   }
第12行               });
第13行       });
第14行   </SCRIPT>
```

例程8-13

```
第1行    <%@ Page language="vb"%>
第2行    <%
第3行        Dim mySeek as string=Request.Form("postcode")'取得地址栏输入的postcode值
第4行
第5行        Dim myData as String
第6行        myData="110101东城,110102西城,110103崇文,110104宣武,110105朝阳,110106丰台,"
第7行        myData+="110107石景山,110108海淀,110109门头沟,110111房山,110112通州,110113顺义,110114昌平,"
第8行        myData+="110115大兴,110116怀柔,110117平谷,110228密云,110229延庆,"  'rem myData为字符串
第9行
第10行       Dim posCode as integer=InStr(myData,mySeek)'计算邮政编码出现位置,instr计算mySeek在myData中的位置
第11行       Dim seekState as boolean'保存查询状态
第12行       Dim areaName as String '申明areaName为一个字符串型变量,区域名称
第13行       If posCode>0 Then '如果存在该邮政编码
第14行           seekState=true
第15行           Dim posComma as integer=InStr(posCode+6,myData,",")'查找邮政编码后逗号出现的位置
第16行           areaName=Mid(myData,posCode+6,posComma-posCode-6)'取出邮政编码逗号前的字符内容
第17行       Else'如果不存在输入的邮政编码
第18行           seekState=false
第19行       End If
第20行       Response.ContentType="text/xml" '数据流格式定义
第21行       response.write("<?xml version='1.0' encoding='GB2312'?>")
第22行       response.write("<Root>")
第23行       response.write("<postcode>"+mySeek+"</postcode>")
第24行       If seekState=True Then
第25行           response.write("<state>1</state>")'找到对应地名
第26行           response.write("<areaname>"+areaName+"</areaname>")
第27行       Else
第28行           response.write("<state>0</state>")'未找到对应地名
第29行       End If
第30行       response.write("</Root>")
```

```
%>
```

比较例程8-12与例程8-11，仅仅第4行的$.get()被修改为$.post()，然后URL有所调整，其他用法完全相同。例程8-13与例程8-10则仅有第3行Request.QueryString()被修改为Request.Form()，其余则完全相同。$.get()与$.post()在使用中主要是数据交互机制不同。

(2) $.load()

$.load()能载入远程HTML代码并插入至选中的元素中，其格式为：

```
$.load(url,[data],[function(){}])
```

$.load()默认使用get方式，当附加参数时，自动转换为post模式。例程8-14是$.load()的应用，其调用远程服务器URL代码如例程8-15所示，执行效果如图8-5所示。

例程8-14

```
<HTML>
    <HEAD>
        <TITLE>邮政编码与地名</TITLE>
        <Script type="Text/JavaScript" language="JavaScript" src="./jquery-1.9.1.min.js"></Script>
        <Style type="text/css">
            #PostCodeList table{width:200px}
            #PostCodeList table td{border-bottom:1px dotted gray;text-align:center;height:30px}
            #PostCodeList table th{height:40px;border-bottom:2px solid gray}
        </Style>
    </HEAD>
    <BODY>
        <Div id="PostCodeList"></Div>
        <SCRIPT LANGUAGE="JavaScript">
            $("#PostCodeList").load("http://127.0.0.1/someHtmlCode.aspx");
        </SCRIPT>
    </BODY>
</HTML>
```

例程8-15

```
<%@ Page language="vb"%>
<%
    Dim myData as String
    myData="<table cellpadding='0' cellspacing='0'>"
    myData+="<tr><th>邮政编码</th><th>地名</th></tr>"
    myData+="<tr><td>110101</td><td>东城</td></tr>"
    myData+="<tr><td>110102</td><td>西城</td></tr>"
    myData+="<tr><td>110103</td><td>崇文</td></tr>"
    myData+="</table>"
    response.write(myData)
%>
```

邮政编码	地名
110101	东城
110102	西城
110103	崇文

图8-5　例程8-14执行效果图

例程8-14第14行代码$("#PostCodeList").load("http://127.0.0.1/someHtmlCode.aspx")能让id名为PostCodeList的元素内容来自远程服务器，即其显示内容可以更具远程服务器上的内容变化，其load()方法(函数)的URL可以输入参数，或者通过函数可选惨[data]附加参数。这种方法使得网页能组织来自不同来源的数据，而数据来源可以根据各种情况发生变化，使得网页变得更加动态。

(3) $.getScript()

通过以get模式请求远程服务器载入并执行一个JavaScript文件。

```
$.getScript(url,[function(){}])
```

$.getScript()可以需要载入不同的JS文件，而通过脚本元素(script)则肯定载入指定的JS文件。因此$.getScirpt()则可以有条件载入。

(4) $.getJSON()

JSON是一种轻便的数据组织模式，有些类似XML，jQuery的AJAX提供了JSON数据的请求模式，其格式如下：

```
$.getJSON(url,[data],[function(){}])
```

(5) $.ajax(options)

实际上jQuery提供各种请求模式都是$.ajax()的简写。在$.ajax()中可以设置各种各选参数，其options如下表所示。

表1：options可选参数表

参数名称	说明
dataType	数据类型，可设置值为: • xml: 返回XML文档，可用 jQuery 处理； • html: 返回纯文本HTML信息，包含script元素； • script: 返回纯文本JavaScript代码，默认不缓存结果，除非设置cache参数； • json: 返回JSON 数据； • jsonp: JSONP 格式。使用JSONP形式调用函数时； • text: 返回纯文本字符串；
cache	布尔型数据，默认值为 true。当dataType为script时默认为false。设置为false将不会从浏览器缓存中加载请求信息。
async	布尔型数据，默认值为true。当为true时所有请求均为异步请求。如果需要发送同步请求，请将此选项设置为false。注意，同步请求将锁住浏览器，用户其它操作必须等待请求完成才可以执行。
beforeSend	回调函数，function(){}。发送请求前执行的函数。
complete	回调函数，function(){}。请求执行完成后回调函数(请求成功或失败均调用)。
success	回调函数，funciton(){}。请求成功后执行的函数。
error	回调函数，function(){}。请求失败时执行的函数。
contentType	字符串型数据，默认为"application/x-www-form-urlencoded"，默认值适合大多数应用场合。
data	数据对象格式。发送到服务器的数据，将自动转换为请求字符串格式。
dataFilter	回调函数，function(){}。对Ajax返回的原始数据进行预处理的函数。提供data和type两个参数：data是Ajax返回的原始数据，type是调用jQuery.ajax时提供的dataType参数。函数返回值由jQuery进一步处理。
global	布尔型数据，默认为true。是否触发全局AJAX 事件，设置为false将不会触发全局 AJAX 事件，如ajaxStart或

	ajaxStop。
ifModified	布尔型数据，默认为false。仅在服务器数据改变时获取新数据，根据http包的Last-Modified信息判断。
username	字符串型数据，用于响应HTTP访问认证请求的用户。
password	字符串型数据，用于响应HTTP访问认证请求的密码。
url	字符串型数据，向服务器发送请求时的地址。
type	字符串型数据，默认GET模式，取值为POST或GET，注意也可选如PUT和DELETE模式，但仅部分浏览器支持。
processData	布尔型数据，默认为true。默认情况下，发送的数据将被转换为对象，如果希望不转换数据，设置为 false。
scriptCharset	字符串型数据，只有当dataType为"jsonp"或"script"，且请求模式为GET时用于强制修改charset，通常在本地和远程的内容编码不同时使用。

(6) $.ajaxSetup(options)

$.ajaxSetup(options)用于设置AJAX默认项，其options与$.ajax(options)相同。当通过$.ajaxSetup(options)设置后，调用AJAX时将直接采用默认选项进行工作，如例程例程8-16所示。

例程8-16

```
第1行  <SCRIPT LANGUAGE="JavaScript">
第2行  $("button").click(function(){
第3行          var postCode=$("input").val();
第4行          $.ajaxSetup({url:"http://127.0.0.1/getNameXML_Post.aspx",type:"POST",cache:false});
第5行
第6行          $.ajax({
第7行                  data:{postcode:postCode,timer:new Date()},
第8行                  success:
第9行                          function(data){
第10行                                 var rtnXML=$(data);
第11行                                 var rtnState=rtnXML.find("state").text();
第12行                                 if(rtnState==0){
第13行                                         $("#seekResult").html(postCode+"没有对应地名，请核实输入是否正确!");
第14行                                 }else{
第15行                                         $("#seekResult").html(postCode+"对应地名是"+rtnXML.find("areaname").text());
第16行                                 }
第17行                         }
第18行                 });
第19行         });
第20行 </SCRIPT>
```

2、AJAX事件

AJAX从启动到结束，会根据状态触发很多事件。与此同时，AJAX事件还可以分为局部事件和全局事件。局部事件可以在$.ajax()中定义并调用，而全局事件如同普通事件(如click)，可以绑定在元素之上，形同$("body").click(function(){})，如$("P").ajaxStart(function(){})，其中ajaxStart是AJAX事件。自jQuery1.8自后，AJAX全局事件只能绑定在document对象之上，如例程8-17所示。

例程8-17

```
第1行  <HTML>
第2行          <HEAD>
第3行                  <TITLE>邮政编码与地名</TITLE>
```

```
第4行                <Script type="Text/JavaScript" language="JavaScript" src="./jquery-1.9.1.min.js"></Script
     >
第5行                <Style type="text/css">
第6行                        #PostCodeList table{width:200px}
第7行                        #PostCodeList table td{border-bottom:1px dotted gray;text-align:center;height:30p
     x}
第8行                        #PostCodeList table th{height:40px;border-bottom:2px solid gray}
第9行                        #runningState #ajaxStart,#runningState #ajaxComplete{display:none}
第10行               </Style>
第11行          </HEAD>
第12行          <BODY>
第13行               <input type=text><button>确定</button>
第14行               <Div id="seekResult"></Div>
第15行               <Div id="runningState">
第16行                       <Div id="ajaxStart">ajax启动，请稍候!</Div>
第17行                       <Div id="ajaxComplete">ajax结束，恭喜你! </Div>
第18行               </Div>
第19行
第20行               <SCRIPT LANGUAGE="JavaScript">
第21行                       $("button").click(function(){
第22行                               var postCode=$("input").val();
第23行                               $.get(
第24行                                       "http://127.0.0.1/getNameXML.aspx",
第25行                                       {postcode:postCode,timer:new Date()},
第26行                                       function(xmlData){
第27行                                               var rtnXML=$(xmlData);
第28行                                               var rtnState=rtnXML.find("state").text();
第29行                                               if(rtnState==0){
第30行                                                       $("#seekResult").html(postCode+"没有对应地名，请
     核实输入是否正确!");
第31行                                               }else{
第32行                                                       $("#seekResult").html(postCode+"对应地名是"+rtnXM
     L.find("areaname").text());
第33行                                               }
第34行                                       });
第35行                       });
第36行                       $(document).bind({
第37行                               ajaxComplete:function(){
第38行                                       $("#runningState").children().hide();
第39行                                       $("#ajaxComplete").show();
第40行                               },
第41行                               ajaxStart:function(){
第42行                                       $("#runningState").children().hide();
第43行                                       $("#ajaxStart").show();
第44行                               }
```

```
第45行                          });
第46行                  </SCRIPT>
第47行          </BODY>
第48行  </HTML>
```

在例程8-17的第36-45行，document对象被绑定两个事件，分别是ajaxComplete和ajaxStart。在实际应用中，一旦ajaxStart被启动，可以让其对象的html代码显示一幅正在运行的图片，修改第16行代码即可，同时，还可以让文本框、命令按钮处于不能输入或者点击状态。当执行到ajaxComplete事件时，根据状态调整相应的网页代码。

表2：ajax事件类别、描述及其触发顺序

事件名称	类别	事件描述	触发顺序
ajaxStart	全局事件	开始新的Ajax请求，并且此时没有其他ajax请求正在进行。	1
beforeSend	局部事件	当一个Ajax请求开始时触发。如果需要，你可以在这里设置XHR对象。	2
ajaxSend	全局事件	请求开始前触发的全局事件	3
success	局部事件	请求成功时触发。即服务器没有返回错误，返回的数据也没有错误。	4
ajaxSuccess	全局事件	全局的请求成功	5
error	局部事件	仅当发生错误时触发。你无法同时执行success和error两个回调函数。	6
ajaxError	全局事件	全局的发生错误时触发	7
complete	局部事件	不管你请求成功还是失败，即便是同步请求，你都能在请求完成时触发这个事件。	8
ajaxComplete	全局事件	全局的请求完成时触发	9
ajaxStop	全局事件	当没有Ajax正在进行中的时候，触发。	10

在全局事件中，除了ajaxStart和ajaxStop之外，其他事件都有3个参数event, XMLHttpRequest, ajaxOptions，第一个是事件，第二个是XHR对象(浏览器内置对象)，第三个参数最有用，是调用ajax时的参数。对于ajaxError，还有第四个参数thrownError，只有当异常发生时才会被传递。如例程8-18所示。

例程8-18

```
第1行   <HTML>
第2行           <HEAD>
第3行                   <TITLE>邮政编码与地名</TITLE>
第4行                   <Script type="Text/JavaScript" language="JavaScript" src="./jquery-1.9.1.min.js"></Script
>
第5行           </HEAD>
第6行           <BODY>
第7行                   <Div id="PostCodeList"></Div>
第8行                   <SCRIPT LANGUAGE="JavaScript">
第9行                           $(document).ajaxComplete(function(e,xhr,o){
第10行                                  alert(o.url);//url是ajax的一个选项，属性，显示为http://127.0.0.1/someHtml
Code.aspx
第11行                                  alert(o.type);//ajax调用模式，显示为GET
第12行                                  alert(o.dataType);//数据类型，显示为html
第13行                          });
第14行                          $("#PostCodeList").load("http://127.0.0.1/someHtmlCode.aspx");
第15行                  </SCRIPT>
第16行          </BODY>
第17行  </HTML>
```

在网页中，如果有多个AJAX启动，在全局事件中可以通过ajax选项的url予以区分。