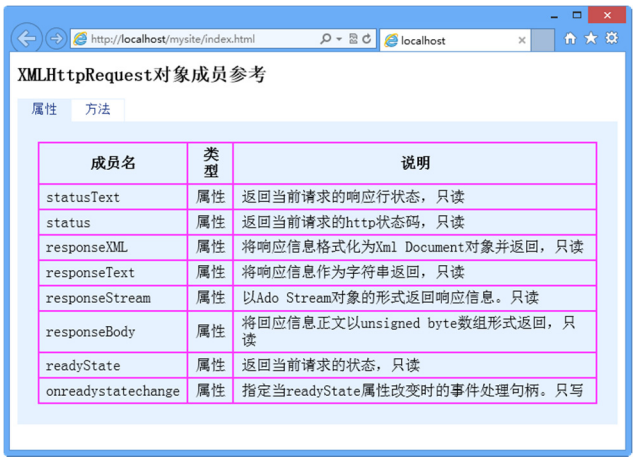
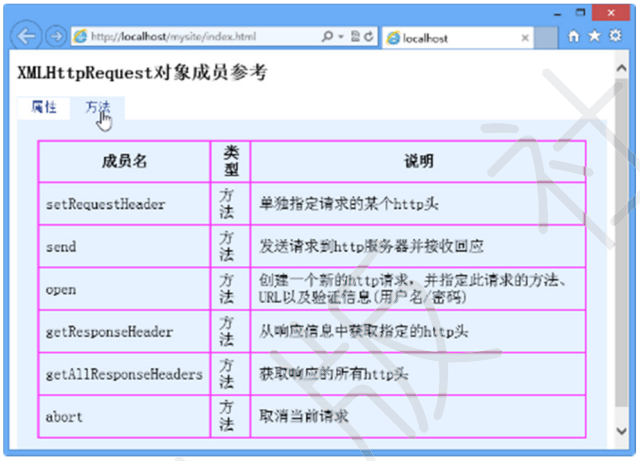
21.3.3 异步更新 Tab 面板内容

本节示例设计一个能够异步更新的 Tab 面板。当用户切换 Tab 面板选项卡时，面板会自动从后台数据库中读取数据并动态显示出来，而不是在页面加载时已经全部下载并隐藏起来。这种局部面板区域的数据更新是 Ajax 应用的一种形式，它真正缓解了带宽给数据传输带来的压力，更为重要的是它能够随时更新页面局部信息，而不是需要完全更新整个页面。整个页面的演示效果如图 E21.3 所示。



显示第一个选项卡



显示第二个选项卡

图 E21.3 异步更新的 Tab 选项卡

本示例将在前面示例的基础上展开讲解，其中后台数据库共享上节 data.mdb 文件中的数据，后台服务器处理脚本依然采用上节 test.asp 中的记录集读写脚本代码，并适当进行调整。

设计思路：

当鼠标指针移到 Tab 标题栏中时，将触发 check()函数，并传递给它一个参数值，告诉当前的 Tab 菜单的编号。check()函数会打开与后台的异步连接，并发送请求，同时把该参数值发送给后台，后台服务器根据这个 Tab 菜单编号，从数据库中查询对应的数据，并转换为 XML 格式的数据响应给客户端。前台浏览器接收到响应数据之后，会自动调用回调函数，由回调函数来处理响应的数据，把它转换为表格结构，最后根据 Tab 菜单编号把这些数据插入到对应 Tab 面板中。

【操作步骤】

第 1 步，由于数据库共享上节示例数据库，这里就不再重复介绍。后台脚本也共享了上节示例 test.asp 文件，复制该文件，然后准备修改部分代码。

第 2 步，上节示例根据客户端传递的记录数来查询记录集，而本节是根据不同的分类来定义记录集。本示例共分两个类（如属性和方法），其中属性使用 1 来表示，2 表示方法，所以只要简单修改查询字符串即可，其他有关记录集的查询、生成 XML 格式数据都可以不同。具体修改的代码如下：

```
<%
coun=CInt(Request("coun"))
'获取客户端传递过来的数据，并转换为数值类型
if  coun = 1 then
'如果值为 1，说明要查询属性类型的成员
        str = "属性"
else
'如果值为 2，说明要查询方法类型的成员
        str = "方法"
end if
%>
<%
set rs = Server.CreateObject("adodb.recordset")   '定义记录集
'重新修改 SQL 查询字符串，其中增加 where 语句，用来筛选记录集
sql ="select * from xmlhttp where class = '"&str&"' order by id desc"
rs.open sql,conn,1,1                                '打开记录集
```

```
%>
```

第 3 步，找到 Javascript 事件处理一章的示例“设计 Tab 面板”，复制其中的 index.html。在页面结构中绑定异步请求的事件处理函数，代码如下，其他代码均保留不动。

```
<h1>XMLHttpRequest 对象成员参考</h1>
<div class="tab_wrap">
    <ul class="tab" id="tab">
        <li id="tab_1" onmouseover="check(1)" class="hover">属性</li>
        <li id="tab_2" onmouseover="check(2)" class="normal">方法</li>
    </ul>
    <div class="content" id="content">
        <div id="content_1" class="show">暂无属性</div>
        <div id="content_2" class="none">暂无方法</div>
    </div>
</div>
```

第 4 步，在页面初始化中也绑定异步处理函数，以实现加载页面时能够自动显示一个 Tab 面板中的数据：

```
window.onload = function(){
    hover();                        //这是原来示例中的 Tab 菜单切换函数
        check(1);                   //调用异步处理函数
}
```

第 5 步，定义异步处理函数。

```
function check(n){
    request.open( "GET", "test.asp?coun=" + n, true );
   request.onreadystatechange = function(){
//使用闭包调用带有参数的回调函数
            updatePage(n);
        };
    request.send( null );
}
```

在异步请求处理函数中，首先打开与后台服务器的连接，并把 Tab 菜单的参数传递过去，然后使用 send()发送请求。在 onreadystatechange 属性中绑定回调函数。

【提示】

本示例需要向回调函数传递 Tab 菜单参数值，但是 onreadystatechange 属性绑定的函数是不能接受参数值，同时 Ajax 也不会向回调函数传递参数。因此，采用了闭包的形式，把参数传递给 updatePage(n)函数，然后使用闭包结构把它包装起来，这样其中的参数值就被保存在回调函数中从服务器端带回到客户端。通俗说，闭包就是一个函数体结构，但是它能够存储数据，并不受外界的影响长期存在，直到函数生命周期结束。

第 6 步，设计回调函数。该回调函数与上节示例回调函数代码结构基本相似，但是增加了一个参数，并对参数进行条件判断，根据不同的参数设置要响应信息插入的对象。增加代码如下：

```
function updatePage(n){
//定义带有参数的回调函数的
        if(n==1){
//如果参数为 1，则设置插入的对象 id 为 content_1
      var info = document.getElementById( "content_1" );
        }else{
//如果参数为 2，则设置插入的对象 id 为 content_2
            var info = document.getElementById( "content_2" );
        }
    if( request.readyState == 1 ){
        info.innerHTML = "<img src='images/loading.gif’ />，连接中.....";
    }
    else if( request.readyState == 2 || request.readystate == 3 ){
        info.innerHTML = "<img src='images/loading.gif’ />，读数据.....";
    }
```

```
    else if( request.readyState == 4 ){
        if( request.status == 200 ){
            xml = request.responseXML;
            info.innerHTML = showXml( xml );
        }
        else
        alert( request.status );
    }
}
```

有关 XML 格式数据读取、转换为表格，并显示出来的函数 showXml(xml)与上节实例中该函数代码完全一样，这里就不再讲解了。