Lab 9 Ajax简单应用

Ajax是Asynchronous JavaScript and XML(以及DHTML等)的缩写，它是一种可以创建丰富的浏览器应用的技术。Ajax尝试建立桌面应用程序的功能和交互性与不断更新的Web应用程序之间的桥梁。本次实验通过一个简单的实例使大家对Ajax技术能有基本的了解和认识。

1. 预备知识

1.1 Ajax简介

Ajax这个短语是Adaptive Path的Jesse James Garrett发明的。（按Jesse的解释，它并不是首字母的缩写。）准确的说，Ajax并非什么最新技术，可以说，它是多种原有技术的综合，有人称它为“旧瓶装新酒”的客户端开发方法。

以下是基于Ajax的应用需要用到的基本技术：

- HTML/XHTML和CSS：用于建立Web表单并呈现标准化的Web页面。
- DOM（Document Object Model）：用于（通过JavaScript代码）实现动态内容显示和（在某些情况下）解析服务器返回的XML。
- XML和XSLT：作为标准的结构化数据存储格式，用于进行数据交换和处理
- JavaScript：运行Ajax程序的核心代码，特别是通过重要的XMLHttpRequest对象进行异步数据的读取，从而实现与服务器应用程序的通信。此外，Javascript还被用于绑定和处理所有的浏览器数据。

在Ajax出现以前，业界对于上述技术的使用基本停留在单独使用的阶段，没有综合性的使用。Ajax的出现充分整合了原有的客户端技术，同时又保持了与各种服务器端技术的良好交互性，形成了具有丰富用户体验的富客户端（Rich Client）技术。

1.2 Ajax的特别之处

与传统的Web应用不同，Ajax应用采用异步交互过程。这种传输模式分离了服务器相应和页面内容的刷新，浏览器不再直接与服务器端打交道，整个交互过程由作为中间媒介的Ajax引擎负责。这里说的Ajax引擎通常是以XMLHttpRequest对象为核心的JavaScript类库，它专门负责用户界面和服务器之间的交互。

由于Ajax引擎的介入，用户的请求不再直接由浏览器生成相应的HTTP请求直接发送给服务器，而是通过JavaScript调用Ajax引擎来代替产生一个HTTP用户请求，由于仅仅使用了客户端脚本JavaScript来发送请求，此时页面不会被刷新或清空。服务器端在处理完请求后（比如简单的数据校验，内存中的数据编辑甚至一些页面导航），将相应数据（通常是XML格式）返回给Ajax引擎，再由Ajax引擎对数据进行处理后，通过DOM和JavaScript在不刷新页面的情况下动态修改部分页面内容。通过这种响应与显示分离的传输模式，Ajax消除了客户端与服务器端交互过程中的处理-等待-处理-等待的缺点。

1.3 使用XMLHttpRequest对象

XMLHttpRequest对象是Ajax技术的核心。可以通过以下方法创建XMLHttpRequest对象。

```
var xmlHttp = false;
function createRequest()
{
    //Mozilla, Firefox等浏览器
    If ( window.XMLHttpRequest() )
        xmlHttp = new XMLHttpRequest();
    //IE浏览器
    else if ( window.ActiveXObject )
    {
        try
        {
            xmlHttp = new ActiveXObject ( “Msxml2.XMLHTTP” );
        }
        catch ( ex_otherms )
        {
             try
            {
                xmlHttp = new ActiveXObject ( “Microsoft.XMLHTTP” );
             }
             catch ( ex_failed )
             {
                xmlHttp = false;
```

```
            }
        }
    }
    //异常，创建对象实例失败
    If ( ! xmlHttp )
    {
        window.alert ( “无法创建XMLHttpRequest对象” );
        return false;
    }
}
```

为了使对象更加容易使用，以下提供了一些辅助例程来简化XMLHttpRequest对象的使用过程。

以下三个函数是Ajax最基本的框架，其重要性这里就不赘述了，请大家参考教材上的说明。

```
function createRequest ( )
{
    var xmlHttp = null;
    //Mozilla, Firefox
    if ( window.XMLHTTPRequest )
        xmlHttp = new XMLHttpRequest();
    //IE
    else if ( window.ActiveXObject )
    {
        try
        {
            xmlHttp = new ActiveXObject ( “Msxml2.XMLHTTP” );
        }
        catch ( anyEx )
        {
            try
            {
                xmlHttp = new ActiveXObject ( “Microsoft.XMLHTTP” );
            }
            catch ( anyEx )
```

```
            {
                xmlHttp = null;
            }
        }
    }
    else
        alert ( “Fail to Create XMLHttp – Ajax Disabled!” );
}

function callback ( )
{
    if ( xmlHttp.readyState == 4 )
    {
        If ( xmlHttp.status == 200 )
        {
            …
        }
        //204响应码指示服务器没有发回任何信息
        else if ( xmlHttp.status == 204 )
        {
            …
        }
    }
}

function sendRequest ( )
{
    …
    var url = “…Servlet?names=” + escape (inputField.value) ;
    xmlHttp.open ( “GET”, url, true );
    xmlHttp.onreadystatechange = callback;
    xmlHttp.send ( null );
    …
}
```

1.4 W3C DOM接口的使用

本次实验中服务器按XML格式发送响应，W3C DOM指定了一组很丰富的API，可用于搜索和处理XML文档，DOM兼容的浏览器必须实现这些API，而且不允许有自己定义的行为，这样就能尽可能地改善脚本在不同浏览器之间的可移植性，DOM可以被任何脚本语言访问，这里我们采用JavaScript访问DOM。下面列出一些有用的API供大家在实验中参考。其中表1和表2用于处理服务器的响应，表3用于处理响应之后的浏览器端显示。

属性名称	描述
childNodes	返回当前元素所有子元素的数组
firstChild	返回当前元素的第一个下级子元素
lastChild	返回当前元素的最后一个子元素
nextSibling	返回紧跟在当前元素后面的元素
nodeValue	指定表示元素值得读/写属性
parentNode	返回元素的父节点
previousSibling	返回紧邻当前元素之前的元素

表 1　用于处理XML的DOM元素的属性

方法名	描述
document.getElementById (id)	获取有指定唯一ID属性值文档中的元素
document.getElementsByTagName (name)	返回当前元素中有指定标记名的子元素的数组
hasChildNodes( )	返回一个布尔值，指示元素是否有子元素
getAttribute (name)	返回元素的属性值，属性由name指定

表 2　用于遍历XML的DOM元素的方法

方法名	描述
document.createElement (tagName)	创建由tagName指定的元素
document.createTestNode (text)	创建一个包含静态文本的节点
appendChild (childNode)	将制定节点增加到当前元素的子节点列表
setAttribute (name, value)	设置name属性的值为value
insertBefore (newNode, targetNode)	将newNode作为当前元素子节点插到targetNode前面
removeAttribute (name)	从元素中删除name属性
removeChild (childNode)	从元素中删除子元素childNode
replaceChild (newNode, oldNode)	节点替换

表 3　动态创建内容时所用的W3C DOM方法

2. 实验：使用Ajax提供自动完成功能

实验目的：

1） 熟悉Ajax客户端程序的编写。
2） 熟悉JavaScript中W3C DOM接口的使用

实验任务：

实现类似于Google Suggest的功能。

实验环境：

Web Browser：IE6+
IDE：MyEclipse

实验交付物：

将nameSuggest.html上传到10.132.141.33 用户名：学号 密码：java

实验步骤：

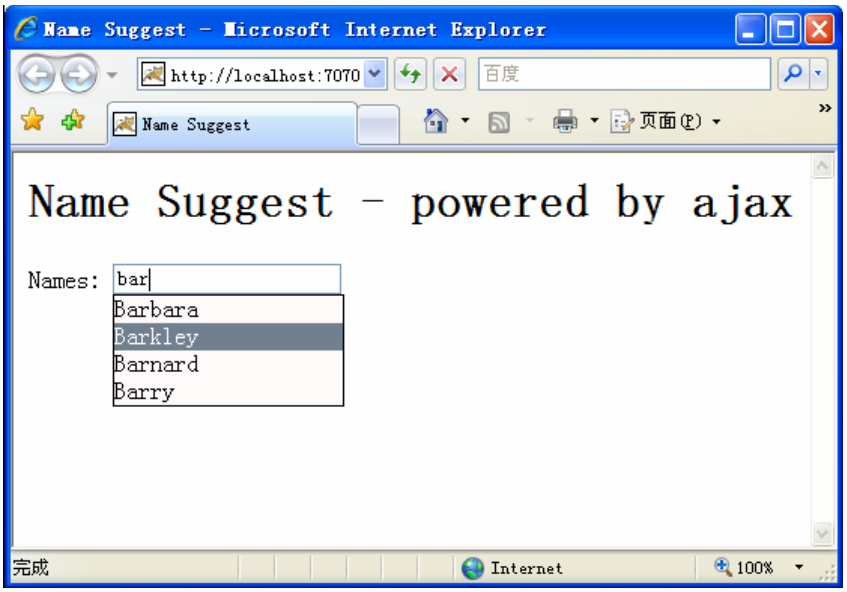
1）在MyEclipse中新建Web项目，将所给的两个Java文件导入。（建议大家自己给项目新建一个servlet文件，然后将代码拷入，否则需要自己配置）
2）实现nameSuggest.html页面，页面请求经Ajax引擎交由给出的Servlet处理
3）将项目部署到tomcat服务器测试运行

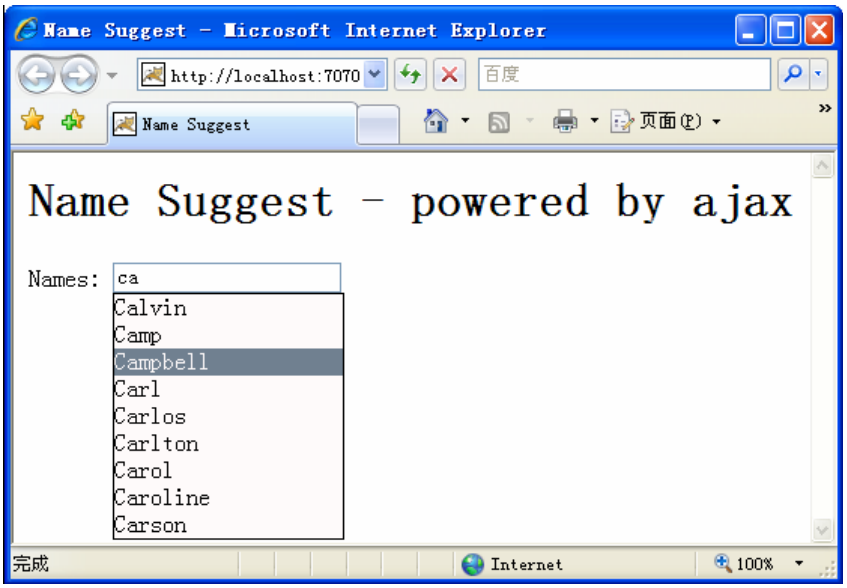
实验提示：

参考1.3和1.4实现nameSuggest.html

实验效果：

类似于 Google Suggest，如下图所示：


Name Suggest - Microsoft Internet Explorer
http://localhost:7070
百度
Name Suggest
页面(P)
Name Suggest - powered by ajax
Names: bar
Barbara
Barkley
Barnard
Barry
完成
Internet
100%


Name Suggest - Microsoft Internet Explorer
http://localhost:7070
百度
Name Suggest
页面(P)
Name Suggest - powered by ajax
Names: ca
Calvin
Camp
Campbell
Carl
Carlos
Carlton
Carol
Caroline
Carson
完成
Internet
100%