基于SOAP协议和Ajax技术构建Web应用

杨 磊,王建斌,马光思,程永阳
(西安建筑科技大学 信控学院,陕西 西安 710055)

摘 要:Ajax技术从根本上改变了Web应用的交互模式,大幅度提高了访问速度。为了提高Web服务器的响应效率,改进Asp.net下Web程序的客户体验,研究Ajax技术在Asp.net环境下的具体应用。结合工程实践,详细分析了Ajax技术的基本原理,讨论了基于SOAP协议和Ajax技术在构建Web应用中进行通信的方法,给出了Asp.net环境下实现Web应用的具体实例、过程和技术要点。

关键词:XMLHttpRequest;文档对象模型;简单对象访问协议;Web服务
中图分类号:TP311.5 **文献标识码**:A **文章编号**:1673-629X(2008)01-0115-04

Use Ajax Application Ground on SOAP Protocol in Web Development

YANG Lei, WANG Jian-bin, MA Guang-si, CHENG Yong-yang
(Sch. of Information & Control, Xi'an Univ. of Architecture & Technology, Xi'an 710055, China)

Abstract: The applications of Ajax have changed the interact mode radically and improved the access speed mostly. For enhancing the server's efficiency, and improving the customer's experience in Asp. net Web application, implementing the Ajax technology in Asp. net, combine to the project practice, detailedly analyse the method of using Ajax to communicate to server ground on SOAP protocol, and show the processation of implementation in Asp. net.

Key words: XMLHttpRequest; DOM; SOAP; Web Service

0 前 言

Ajax技术的核心为Javascript,XMLHttpRequest,DOM和XML或其它的数据格式。

传统的Web交互方式,用户触发一个HTTP请求给服务器,然后空闲等待,无论用户请求的信息有多小,服务器都要对其分析处理然后生成并返回一个新的完整的HTML页,这种同步交互使用户每次都要浪费时间和带宽去重新面对新老相同的整个页面。

使用Ajax技术,可以和服务器实现异步交互,所有的操作都会及时响应,用户几乎感觉不到页面重载时的等待。

异步交互的能力主要依靠XMLHttpRequest对象来实现,此外还需要依靠消息交互协议,例如SOAP协议。

收稿日期:2007-04-02
基金项目:陕西省自然科学基金(2005F38);陕西省教育厅产业化培育项目(02JC39)
作者简介:杨 磊(1983-),男,陕西咸阳人,硕士研究生,研究方向为软件复用技术;马光思,教授,硕士研究生导师,主要研究方向为计算机科学与应用。

1 Ajax技术要点分析

1.1 XMLHttpRequest对象

Ajax的一个最大的特点是无需刷新页面便可向服务器传输或读写数据(又称无刷新更新页面),这一特点主要得益于XMLHTTP组件的XMLHttpRequest对象。XMLHttpRequest对象以一组特定的方法和属性与服务器进行数据层面的交互,而不用每次都生成传送及刷新界面。这样既减轻了服务器的负担,又加快了响应速度,缩短了用户等待时间。XMLHttpRequest对象的方法和属性各有6种[1]:

M1 abort() 停止当前请求
M2 getAllResponseHeaders()
作为字符串返回完整的headers
M3 getResponseHeader("headerLabel")
作为字符串返回单个的header标签
M4 open("method","URL"[,asyncFlag],
"userName"[,"password"]])
设置未决请求的目标URL、方法和其他参数
M5 send(content) 发送请求
M6 setRequestHeader("label","value")
设置header并和请求一起发送
A1 onreadystatechange 状态改变的事件触发器

A2 readyState

对象状态(integer): 0=未初始化;1=取中;2=已读取;3=交互中;4=完成

A3 responseText

服务器进程返回的数据文本

A4 responseXML

服务器进程返回数据的 XML 文档对象

A5 status

服务器返回的状态码,如:

404="文件未找到"、200 ="成功"

A6 statusText 服务器返回的状态文本信息

1.2 SOAP 协议与实例分析

SOAP的设计思想主要是为了独立于任何一种特定的编程模型和其他特定实现的语义而提供的一种轻量级协议,用于在分散型、分布式环境中交换结构化信息。SOAP利用 XML 技术定义一种可扩展的消息处理框架,它提供了一种可通过多种底层协议进行交换的消息结构。

做为核心的 SOAP 消息处理框架定义了一整套 XML 元素,用以"封装"任意 XML 消息以便在系统之间传输。该框架包括以下核心 XML 元素:Envelope, Header,Body 和 Fault,所有这些都来自 SOAP1.1 中的 http://schemas. xmlsoap. org/soap/envelope/ 命名空间。SOAP 还有其它版本,目前最新版本是 SOAP1.2,由 http://www. w3. org/2002/12/soap - envelope 命名空间标识,它同样支持这些核心 XML 元素。Envelope 元素始终是 SOAP 消息的根元素,检查根元素名即可使应用程序识别 SOAP 消息。通过检查 Envelope 元素的命名空间,应用程序也可确定所使用的 SOAP 版本。以下消息模板展示了 SOAP Envelope 的结构[2]:

```
<soap: Envelope xmlns: soap = "http://schemas. xmlsoap. org/soap/envelope/" >
  <soap:Header> <! - optional→
    <! - header blocks go here··· →
  </soap:Header>
  <soap:Body>
    <! - payload or Fault element goes here··· →
  </soap:Body>
</soap:Envelope>
```

Envelope 元素包含一个可选的 Header 元素,后跟一个必要的 Body 元素。Body 元素代表了该消息的有效内容。作为一种通用容器,它可包含来自任何命名空间的任意数量的元素。例如,以下的 SOAP 消息代表在银行账户之间的一个转账请求:

```
<soap: Envelope xmlns: soap = "http://schemas. xmlsoap. org/soap/envelope/" >
  <soap:Body>
    <x:TransferFunds xmlns:x="urn:examples - org:banking" >
      <from>22 - 342439</from>
      <to>98 - 283843</to>
      <amount>100.00</amount>
    </x:TransferFunds>
  </soap:Body>
</soap:Envelope>
```

HTTP 协议绑定定义了在 HTTP 上使用 SOAP 的规则。SOAP 请求/响应自然地映射到 HTTP 请求/协议模型。图 1 展示了 SOAP HTTP 绑定的很多细节。

HTTP 请求和响应消息的 Content - Type 标头都必须设为 text/xml (在 SOAP 1.2 中是 application/soap + xml)。对于请求消息,它必须使用 POST 方法。SOAP 规范还定义了一个名为 SOAPAction 的新 HTTP 标头,所有 SOAP HTTP 请求(即使是空的)都必须包含该标头[3]。SOAPAction 标头旨在表明该消息的意图。

客户端请求封装格式

```
POST /path/bank. asmx HTTP/1.1
Content - Type: text/xml
SOAPAction:"urn:banking:transfer"
Content - Length:nnnn
<soap:Envelope···
```

服务器响应封装格式

```
HTTP/1.1 200 OK
Content - Type:text/xml
Content - Length:nnnn
<soap:Envelope···
```

```
HTTP/1.1 500 Internal Server Error
Content - Type:text/xml
Content - Length:nnnn
<soap:Envelope···
```

图 1 SOAP HTTP 绑定

2 Ajax 技术开发 Web 应用的过程分析

如下通过从页面异步调用目标服务器的一个 Web 方法,并且得到这个 Web 方法的返回值来介绍 Ajax 技术的实现,所用的开发环境是 Visual studio2003 中的 Asp. net。

STEP1:首先公开一个 Web 方法,以便在页面中对它进行调用,在 Solution Explorer 窗口中右键点击解决方案,在弹出的菜单中的 Add 大项中选择 Add Web Service 小项来添加一个 Web Service,将其命名为 MyWebService. asmx。在对应该 Web Service 的后缀名为 cs 的文件中添加的所有公有方法,假如方法的定义前面有[WebMethod]关键字,表明这是一个 Web 方法。如下所示:

```
using System;
```

```
……
namespace AjaxDemo
{
    public class MyWebService: System.Web.Services.WebSer
vice{
    public MyWebService()
    { InitializeComponent(); }
    ……
    [WebMethod]
    public string MyWebMethod(string param1,string param2)
    { return param1 + param2; }
    public string MyMethod(string param1,string param2)
    { return param1 + param2; }
    }
}
```

上述代码表示 MyWebMethod 既是公有方法,也是个 Web 方法,它有两个字符串型的参数 param1 和 param2,这个方法可以被外部程序调用,而 MyMethod 只是个公有方法,不能被外部调用。

STEP2:接着按照和上面类似的方法,在 Add 菜单中选择 Add WebForm 项来添加一个页面,将其命名为 AjaxTestPage.aspx。在页面对应的后缀名为 aspx 文件的脚本部分可以用 Javascript 来编写所需要的代码,Ajax 的主要实现就可以写在这个文件中。在这里创建一个 XMLHttpRequest 对象,要获得 XMLHttpRequest 对象在不同的浏览器中需要不同的方法,下面的代码是在 Internet Explorer 浏览器和非 Internet Explorer 浏览器都能得到 XMLHttpRequest 对象的一段代码:

```
try{
    xmlHttp=new
      ActiveXObject("Msxml2.XMLHTTP");
    }
catch (e){
    Try{ xmlHttp=new
    ActiveXObject("Microsoft.XMLHTTP"); }
    catch (e2)
        { xmlHttp = false;}
    }
if (! xmlHttp && typeof XMLHttpRequest ! = 'undefined')
{ xmlHttp=new XMLHttpRequest();}
```

生成 XMLHttpRequest 对象后,即可利用该对象向 WebService 发异步请求去调用它的 Web 方法,以获得该方法的返回结果。

STEP3:为了使 WebService 能理解所发送的参数,需要利用 SOAP 协议将参数封装成固定的 XML 文件。封装的 XML 文件的对应结点的名称和被请求的 Web 方法的参数名称必须一致,以保证相应结点的数据会被做为参数传入到这个 Web 方法中。接着即可发起调用 MyWebMethod 方法的请求,具体核心代码如下:

```
function AjaxRequest(){
xmlHttp.onreadystatechange=
    AjaxCallBackResp(xmlHttp); *
xmlHttp.open("POST","MyWebService.asmx",true);
xmlHttp.setRequestHeader("Cache-Control","no-cache");
xmlHttp.setRequestHeader("Content-Type","text/xml;charset=
utf-8");
xmlHttp. setRequestHeader ("SOAPAction", "http://tempuri.
org/"+"MyWebMethod");
soapMessage="<? xml version=      \"1.0\" encoding=\"
utf-8\"? >";
soapMessage+="<soap:Envelope"
+"xms:xsi=\"http://www.w3.org/2001/
    XMLSchema-instance\""
+"xmlns:xsd=\http://www.w3.org/2001/XMLSchema\"
+"xmlns:soap=\"http://schemas.xmlsoap.org /soap/envelope/
\">";
soapMessage += "<soap:Body>";
soapData ="<MyWebMethod xmlns=
    \"http://tempuri.org/">";
soapData += "<param1>data1</param1>";
soapData += "<param2>data2</param2>";
soapData += "";
soapMessage = soapMessage + soapData +"</soap:Body>";
soapMessage =soapMessage +"</soap:Envelope>";
xmlHttp.send(soapMessage);}
```

代码中 Open 方法的第二个参数表明了请求的是"MyWebService.asmx"这个 WebService,这就是最初建立的 WebService。对 SOAPAction 标头的设置表明去调用名为"MyWebMethod"的 Web 方法。soapMessage 变量存储的这个字符串是发送到目标服务器的数据主体,通过这种方式,可以将 data1 和 data2 传送到 MyWebService.asmx 的 MyWebMethod 方法中进行处理。

```
* xmlHttp.onreadystatechange=
    AjaxCallBackResp(xmlObj);
```

这行代码用来监视目标服务器的处理动作,每当 xmlHttp 的 onreadystatechange 属性发生变化时,就会调用 AjaxCallBackResp 函数。该函数的实现细节如下:

```
function AjaxCallBackResp(xmlHttp){
  switch(xmlHttp.readyState){
    case 4://对服务端的请求完成
    switch(xmlHttp.status)
    { case 200://服务端的应答正确
    break;
    }
    break;
```

```
    case 0://未初始化!
    break;
    case 1://读取中!
    break;
    case 2://已读取!
    break;
    case 3://交互中!
    break;
  }
}
```

xmlHttp 对象的 readyState 属性反映了目标服务器对请求的处理状态,可以在不同的状态写上自己的与状态对应的处理代码。当 readyState 为 4 时,表明处理结束,可以在这时候去取 Web 方法的返回值[4]。如果返回值是字符串,直接通过 responseText 属性去取,如果返回值是 xml 文件,则通过 responseXML 属性取到返回值,然后通过 DOM 对它进行操作,代码如下:

```
var xmlDoc=new
    ActiveXObject("Msxml2.DOMDocument.3.0");
xmlDoc.async=false;
xmlDoc.loadXML(xmlObj.responseXML.text);
```

接着就可以利用 DOM 的方法对这个 XML 文件进行读取。当读到返回值时,就可以根据这些返回值通过 DOM 和 Javascript 来控制页面,使页面完成无缝重构,从而使页面内容实现无刷新的更新。

3 结束语

使用 Ajax 技术,由于在实现的过程中要用到大量的脚本语言,这无疑给开发工作加大了很多工作量,也增加了开发的难度[5]。但它使客户端和服务器之间的通讯量大大减少,极大地减轻了服务器负担;而且可以异步操作,同时进行多个数据更新操作,实现了真正的多任务处理。

参考文献:

[1] Dynamic HTML and XML: The XMLHttpRequest Object [EB/OL]. 2005-06. http://developer.apple.com/internet/webcontent/xmlhttpreq.html.

[2] Skonnard A. 理解 SOAP[EB/OL]. 2003-03. http://www.microsoft.com/china/MSDN/library/WebServices/WebServices/UnderstandingSOAP.mspx? mfr=true.

[3] Ewald T. 关于 SOAP 编码的论点[EB/OL]. 2004-04. http://www.microsoft.com/china/MSDN/library/WebServices/WebServices/UnderstandingSOAP.mspx? mfr=true.

[4] Teare D. An Introduction To Ajax[EB/OL]. 2005-08. http://dev2dev.bea.com/pub/a/2005/08/ajax-introduction.html.

[5] McLaughlin B. Make asynchronous requests with Javascript and Ajax[EB/OL]. 2006-03. http://www-128.ibm.com/developerworks/xml/library/wa-ajaxintro2/.

(上接第 114 页)

[J]. Fuzzy Sets and Systems, 2001(124):97-107.

[5] Roy A, Pa Sankar K. Fuzzy Discretization of Feature Space for a Rough Set Classifier[J]. Pattern Recognition Letters, 2003(24):895-902.

[6] Swiniarski Roman W, Skowron A. Rough Set Methods in Feature Selection Letters and Recognition[J]. Pattern Recognition Letters, 2003(24):833-849.

[7] Beaubouf T, Petry Frederick E, Arora G. Information Theoretic Measures of Uncertainty for Rough Sets and Rough Relational Database[J]. Journal of Information Sciences, 1998(109):185-195.

[8] Guan J W, Bell D A. Rough Computational Methods for Information Systems[J]. Artificial Intelligence, 1998(105):77-103.

[9] Pawlak Z. Rough Set: Theorencal Aspects of Reasoning About Data[M]. Dordrecht: Kluwer Academic Publishers, 1991.

[10] Yao Y Y. Granular Computing: basic issues and possible solutions[C]// Proc. of Fifth Joint Conference on Information Sciences. Atlantic City, New Jersey, USA: [s.n.], 2000:186-189.

[11] Yao Y Y, Li X. Comparison of rough-set and interval-srt models for uncertain reasoning[J]. Fundamental Informatics, 1996,27:289-298.

[12] 史忠植. 知识发现[M]. 北京:清华大学出版社,2002.

[13] 张文修,吴伟志. 粗糙集理论与方法[M]. 北京:科学出版社,2001.

[14] 刘 清. Rough 集及 Rough 推理[M]. 北京:科学出版社,2001.

[15] Yao Y Y, Wong S K M, Wang L S. A nonnumeric approach to uncertain reasoning[J]. International Journal of General Systems, 1995,23(2):343-359.

[16] Yao Y Y, Zhong Ning. Granular Computing Using Information Table[M]// in Lin T Y, Yao Y Y, Zadeh L A. Data Miming, Rough Sets and Granular Computing. [s.l.]: Physica-Verlag, 2000:102-124.

[17] 张燕平,张 铃,夏 莹. 商空间理论与粗糙集的比较[J]. 微机发展,2004,14(10)::21-24.

[18] 张燕平,张 铃,吴 涛. 不同粒度世界的描述法——商空间法[J]. 计算机学报,2004,27(3):328-333.

[19] Allen J F. Towards a General Theory of Action and Time[J]. Artificial Intelligence, 1984,23:123-154.

[20] Allen J F. Planning Using a Temporal World Model[C]//IJCAI-83. [s.l.]: [s.n.], 1983:741-747.