基于 WebService Behavior 的 Web 页面数据实时显示技术研究与实现

Research and Implementation the Real – time Display of Data in Web Pages Based on WebService Behavior Technology

杨启亮　邢建春　王 平
（解放军理工大学工程兵工程学院国防工程智能化技术研究中心 江苏南京 210007）

摘要：指出了现有 Web 页面数据实时显示实现技术存在的不足，提出了利用 WebService Behavior 技术实现 Web 页面数据实时显示的新方法。研究了.NET 环境下利用 WebService Behavior 技术实现 Web 页面数据实时显示的关键过程。

关键词：Web Services　WebService Behavior　.NET　实时显示

1　前言

DHTML（动态 HTML）和 Web Services（Web 服务）的出现，为 Web 页面的开发和实现提供了新的技术手段，本文利用 WebService Behavior 解决了 Web 页面数据的实时刷新问题，为实现 Web 页面数据实时显示提供了一种新方法。

2　Web 页面实时显示技术比较

当前实现 Web 页面数据实时刷新的技术主要有两种：

其一，通过 ActiveX 控件或 JavaApplet 插件实现，这种方法在目前应用的较为流行。其实现原理为：将定时器和数据通讯（访问远程数据库、Socket 通讯等）和显示逻辑封装在控件或插件中，作为 Web 页面的一部分下载到客户端，实现数据在控件或插件区域的实时刷新。这种方法的缺点是浏览器与控件或插件的交互性不强，且控件和插件必须要用单独的开发工具开发，调试困难等；

其二，通过将 Web 页面分割的方法实现，将浏览器显示页面分割成多个 Frame，需要实时数据显示的页面文件单独放在一个 Frame 中，通过定时刷新该 Frame 中的页面实现数据的刷新，该方法实质上仍是对整个页面的刷新，刷新效果和性能较差。

本论文中，采用了 Microsoft. NET 平台最新的 WebService Behavior 技术，较好地解决了网页上数据的实时显示问题。

3　Web Services 与 WebService Behavior

3.1　Web Services

Web Services 是基于网络的、分布式的模块化组件，它执行特定的任务，遵守具体的技术规范，这些规范使得 Web Services 能与其他兼容的组件进行互操作。

Web Services 的主要目标就是在现有的各种异种平台的基础上构筑一个通用的平台无关、语言无关的技术层，各种不同平台上的应用依靠这个技术层来实施彼此的连接和集成。

作为基于 Web 的技术的重要发展，Web Services 类似于常见 Web 站点的分布式服务器端应用程序组件。但是，与基本 Web 的应用程序不同的是：XML Web Services 组件不具有 UI（User Inerface，用户界面）。Web Services 以 XML 为基础，包括 SOAP，WSDL

和 UDDI。Web Services 基本协议栈如表 1 所示。

表 1 Web Services 基本的协议栈

服务发布/发现	UDDI
服务描述	WSDL
XML 消息	SOAP
传输网络	HTTP,SMTP,FTP,HTTPs Over TCP/IP

Web Services 的核心技术是 SOAP,SOAP 是一种简单、轻量级的协议,用于在 Web 上传输、交换 XML 数据。客户应用程序正是通过 SOAP 协议来访问 Internet 上的 Web 服务的。

WSDL 是一种描述 Web 服务的 XML 语言,它定义了描述 Web 服务接口规范的标准格式。有了 WSDL,服务请求者就可以真正以一种语言无关和平台无关的方式自动产生 Web 服务的代理。从概念上讲,就像在 CORBA 或 DCOM 中使用 IDL 文件一样,WSDL 是用来描述 Web 服务的编程接口的,WSDL 文件是客户与服务器之间的一个协约。

UDDI 是一套基于 Web 的、分布式的、为 Web 服务提供信息注册中心的发现标准规范。简言之,UDDI 标准定义了一个 Web 服务发布与发现的方法。有了 UDDI,就可以建立一个全球化的、平台无关的、开放式的架构。

值得注意的是,XML Web Services 不是 DCOM 的替换,而是用于在使用行业标准的平台间进行通讯的一种消息处理基础结构。

3.2 WebService Behavior

XML Web Services 可被多种客户端应用程序使用。XML Web Services 的客户端可以是 Windows 应用程序、 Web 窗体和其他 XML Web Services。但 Microsoft 当初开发 XML Web Services 时并没有考虑将浏览器作为客户端来访问 XML Web Services。为了弥补这一缺陷,Microsoft 推出了最新的 WebService Behavior(行为)技术。

所谓 WebService Behavior 是指能通过 HTTP、SOAP 协议访问 Web Services 的可复用的 DHTML 组件 。这种利用 Behavior 访问 Web Services 的技术的主要优势在于在不需要加载整个页面的情况下更新页面,本文正是利用这一特性来实现客户端数据的实时显示。

只有 Internet Explorer 5.0 版或更高版本,才能从浏览器直接访问 Web 服务,即可以通过客户脚本直接访问 Web 服务。

当前在使用 Web Services 行为构建页面时,存在两方面的限制:①WebService Behavior 只能访问与包含这个行为的网页在同一个域中 Web 服务(这是 DHTML 内置的安全限制所造成的)。如果希望访问远程 Web 服务,那么必须通过相同域中的代理 Web 服务来访问它。②Web Services 行为访问的数据类型是有限的。Web Services 行为支持所有基本的.NET 类型,比如 String、Integer、DateTime、数组等。但 Web Services 行为不支持更复杂的类型,如 DataSet,DataTable,Collection 或自定义的对象等,在使用 Web Services 行为访问数据库时,必须克服这个限制。

3.3 几个概念的区别

本文提到了几个容易混淆的概念:Web Services,Web Service,WebService。Web Services 不能直观地理解为 Web Service 的复数形式。Web Services 首先是指用于构架 Web Service 的整体技术框架,也用作对而 Web Service 的统称;Web Service 则是使用 Web Services 技术而创建的应用实例即特定的服务。Web Service 和 WebService 含义基本相同,只是 WebService 常做定语,本文提及的 WebService Behavior 就是这种用法。

4 WebService Behavior 调用原理

使用 WebService Behavior 时,首先需要获取 webservice.htc 文件,这个文件可以从 Microsoft 的网站上下载(当前为 http://msdn.microsoft.com/workshop/author/webservice/ webservice.htc),为了避免 DHTML 所产生的 Behavior 相关的异域所不能访问的安全限制,应该将 webservice.htc 文件和 Web 页面文件位于同一目录下。

整个 WebService Behavior 的调用过程可分为四个部分:①配属 WebService Behavior,②定位 Web Services,③调用 Web Services 方法,④处理返回结果。下面分别进行讨论。

(1) 配属 WebService Behavior 。配属 WebService Behavior 是指通过某个 HTML 元素的 Sytle 属性将 WebService Behavior 文件即 webservice.htc 配属到该元素中,从而使该 Behavior 能为整个 HTML 文档所使

用。同时为了使这个元素能便于被客户端脚本所引用，该元素的 ID 属性也需赋予有意义的名字。配属过程代码如图 1 表示。

```
<body>
<divid = " service"  style = " behavior: url( webservice. htc) "  > </
div >
</body
```

图 1　配属 WebService Behavior

（2）定位 Web Services。所有给定的 Web Service 都位于一个特定的 URL（Uniform Resource Locator，统一资源定位符）中，因此 WebService Behavior 提供 useService 方法来实现将特定的 Web Service URL 映射为某个友好的命名，从而能提高客户端脚本的可读性和简化脚本编写过程。UseService 方法的基本语法为：id. useService（"ServiceURL"，"FriendlyName"），这里的 id 是指 Behavior 所配属到的那个元素的 ID 属性值（如上文的 service），方法带有两个参数，第一个参数是指特定 Web Service 的 URL，第二个参数是指 URL 映射后的命名。图 2 为 useService 方法调用的 Java Script 例子代码：

```
<script language = " JavaScript" >
function init( )
{
  service. useService( " dataservice. asmx? WSDL" ," mydataser-
vice" ) ;
}
</script > <body onload = " init( ) " >
( div id = " service"  style = " behavior: url( webservice. htc) " >
</div >
</body >
```

图 2　调用 useService 方法的 Java Script 例子代码

代码中的 useService 方法第一个参数的 URL 后面，有一个带查询字符串"？WSDL"的约定，意为采用一个 Web Services 定义语言（Web Services Definition Language）文件来描述这个 Web Service 即 dataservice. asmx。这个 WSDL 文件是一个带. xsd 扩展名的 XML 文件，其包含了构建方法调用的所有信息，Behavior 在 useService 方法被调用时从服务器端下载该描述文件。

一旦 useService 方法被调用，相应的 Web Service 就被定位和映射成为一个友好的名称（本例为 mydataservice）。

（3）调用 Web Services 方法。完成 Behavior 配属和定位并映射 Web Service 后，下一步的工作就是如何访问 Web Services 中的各种方法以获取需要的服务。WebService Behavior 提供 callService 方法函数来实现对 Web Services 中方法的调用。CallService 方法的调用语法为：iCallID = id. FriendlyName. callService（[CallbackHandler,] " MethodName" , Param1, Param2, . . . ）。iCallID 是调用该方法后的特定的返回值，用于标示和定位每次对方法的调用。方法函数中的第一个可选参数是用于处理返回结果的回调函数句柄，如果没有指定该参数，则返回结果的处理则在 event 对象句柄的 onresult 事件过程中进行，onresult 事件在对 callService 完成远程调用后触发；第二个参数为调用的 Web Service 的方法名称；剩下的部分为传递给方法的参数，这些参数不需要指明参数的数据类型。图 3 为 callService 的调用示例。

```
<script language = " JavaScript" >
var iCallID;
var x = new Array( ) ;
iCallID = service. dataservice. callService( " data" ,x) ;
</script >
```

图 3　callService 的调用示例

（4）处理返回结果。callService 方法在 WebService Behavior 和 Web Service 之间产生的是一个异步通讯过程，也就是对 Web Services 中方法的调用和结果的返回是两个独立的过程。Behavior 技术采用了 event 对象句柄和回调函数两种方式来处理返回的结果。

当 WebService Behavior 接收到调用结果的 XML 数据包时，将触发 onresult 事件（当且仅当 callService 中没有指定回调函数）。Onresult 事件是 event 对象的一个事件。图 4 为采用 event 对象的 onresult 事件处理返回结果的客户端代码。

```
<SCRIPT language = " Javacript" >
var iCALLID = 0;
function init( )
{
    service. useService ( " /services/math. asmx? WSDL " ," MyMath" ) ;
iCallID = service. MyMath. callService( " add" ,5,6) ;
}
function on WSresult( )
{
 event. result. value) ;
}
</SCRIPT >
<body onload = " init( )" >
<div id = " service" style = " behavior: url( webservice. htc)" onresult = " on WSresult( )" >
</div >
</body >
```

图 4 基于 Event 对象事件方式处理返回结果的客户端代码

从上面代码可以看出，在调用 onresult 事件时，利用绑定的 HTML 元素的 onresult 属性来指明处理返回结果的函数过程名称。

利用回调函数(callback)时，直接在 callService 的第一个参数中指定处理返回结果的函数过程名称，而不需在 HTML 元素中指定，示例代码参见图 6。

应该说，这两种方法都是很好的返回结果处理方法。二者之间最主要的区别是事件采用了匿名广播的机制，能为所有侦听它的函数所接收和处理。而回调方式的代码性更强一些，更易于编程处理。本文采用了回调的方式来处理返回结果。

5 用 WebService Behavior 实现 Web 页面的实时数据显示

本文以一个简单实例来说明整个 Web 页面数据实时显示的实现过程。

某工程设备监控系统中采用 SQL Server 存储现场设备的运行数据，这些数据通过一定机制在 SQL Server 中实时更新的，系统采用 IE 网络浏览器作为客户端来实时浏览和监视现场设备运行状态和参数。该系统采用 WebService Behavior 技术实现了数据在 IE 浏览器中实时显示。

基本实现思想是：采用. NET 平台作为开发工具，将对 SQL Server 数据库访问部分封装为 Web Service 方法，将从 SQL Server 查询来的数据作为该方法的返回值，IE 浏览器通过 Behavior 定时访问这个 Web Service 方法获取返回数据，并在 HTML 元素中显示，从而实现了客户端 Web 页面的数据刷新。下面分别从服务器端 Web Service 方法定义，客户端 Behavior 对 Web Service 方法的访问两个部分来详细讨论 Web 页面的数据实时显示的实现过程。

(1) 服务器端 Web Service 定义。. NET 开发平台对 Web Services 有着固有的集成能力，能非常简单地使用 ASP. NET 创建 Web Services。

使用 ASP. NET 生成 XML Web Services 时，它自动支持使用 SOAP、HTTP – GET 和 HTTP – POST 协议的客户端通讯。ASP. NET 提供用于 XML Web Services 内部工作的基础结构，所以开发人员可以致力于实现其特定 XML Web Services 的功能。使用 ASP. NET 开发 XML Web Services 一般按照如下三个步骤进行：①创建一个具有 . asmx 文件扩展名的文件。②在该文件中使用指令声明 XML Web Services。③定义组成 XML Web Services 功能的 XML Web Services 方法。

作为示例，实现了一个名为 dataservice 的 Web Service，在 dataservice 中提供 data 方法，来供客户端 Behavior 调用。在 data 方法内部封装了对 SQL Server 的数据访问逻辑，数据访问对象采用了 ADO. NET。ADO. NET 一般以 DataSet 或 DataTable 返回数据库数据，但由于 WebService Behabior 对数据类型有着固有的限制，不能访问 DataSet 或 DataTable 等复杂数据类型，为了克服这个限制，在 data 中，将 DataTable 转换为一个数组并作为方法的返回值。该方法的输入参数为字符串数组，存储客户端数据，并将这些数据作为查询数据库用的条件值，将查询结果以字符串数组返回给客户端。在设备监控系统中，输入参数指测点编号(例如 C0001)，输出结果代表相应测点的实时数据。Web Service 中 data 方法的 VB. NET 实现代码(略)。

(2) 客户端 Behavior 对 Web Service 方法的访问。该部分主要是按照 WebService Behabavior 的调用原理编写客户端脚本程序，实现对已定义的 Web Service 即 dataservice 中的 data 方法的调用，获得返回数据并在 Web 页面中实时显示。

在一个名为 WebForm1. aspx 的 ASP. NET 网页上有四个 HTML label 控件，分别取名为 L1，L2，L3，L4，用于显示四个测点的实时数据。在设计态时，每个 label 控

件的内容(InnerText 属性值)为测点的编号,在页面加载时(运行态),获取相应编号并将其存储在一个数组中,Behabavior 将该数组作为 callService 的参数传递给服务器端的 data 方法,data 方法以这组测点编号作为数据库查询条件从 SQL Server 数据库中获取相应测点的实时数据并返回给客户端处理并在这四个 HTML label 显示。以下为基于 JavaScript 的客户端的主要实现代码。

```
<script language = " JavaScript" >
<! - -
    var j = 0
    var x = new Array( ) ;
    var iCallID = 0;
    var a = new Array( ) ;
    function init( )
    {
      x[0] = L1. innerText;
      x[1] = L2. innerText;
      x[2] = L3. innerText;
      x[3] = L4. innerText;
      window. setInterval( " dataAccess( )" ,3000) ;
    }
    function dataAccess( )
    {
        service. useService( " dataservice. asmx? wsdl" ," mydataservice" ) ;
        iCallID = service. mydataservice. callService (dataResults," data" ,x) ;
    }
    function dataResults( result)
    {
      a = result. value;
      L1. innerHTML = a[0] ;
      L2. innerHTML = a[1] ;
      L3. innerHTML = a[2] ;
      L4. innerHTML = a[3] ;
    }
    - - > </script >
<body onload = " init( )" >
<div id = " service" style = " behavior: url( webservice. htc)" >
</div >
</body >
```

代码中,共定义了三个函数即 init( )、dataAccess( )和 dataResults( )和两个数组变量 x 和 a。Init( )在页面加载时执行,用于从 Label 控件的 innerText 属性中获取测点编号并将编号存储到 x 数组中,同时设定调用 dataAccess( )函数的定时间隔为 3 秒。

dataAccess( )函数首先采用 useService 定位服务 dataservice 并将其映射为 mydataservice,然后利用 callService 调用服务的 data 方法,该调用方式采用了回调技术。callService 的第一个参数为回调函数名称 dataResults,第二个参数为服务 dataservice 中的方法名 data,第三个参数为 data 方法需要的参数 x。

dataResults 是 useService 的回调函数,用以接收从服务 dataservice 传来的数据,回调函数通过 result 对象获取 dataservice 中 data 方法返回的数据。通过数组 a 取出数据后赋予 Label 控件的 innerText,将服务器端 SQL Server 数据在 Web 页面中显示。

由于 dataAccess( )函数定时执行,所以 callService 能定时向服务 dataservice 请求和返回数据,从而实现了 Web 页面中的数据的实时显示功能。

6 结论

本文利用 WebService Behavior 技术实现服务器端数据库数据在 Web 页面的实时显示,无闪烁感,视觉效果很好。该方法克服了传统方法的不足,具有调试方便、开放性强、运行可靠等优点。该技术已在某特大型洞库群基于 B/S 架构的智能化监控系统项目中获得成功应用,工程运行维护人员可方便地通过 Web 页面实时监视整个工程设备运行的状况。

参考文献

1 Microsoft Corporation. Using the WebService Behavior[EB/OL] Microsoft Developer Network, 2003.

2 [美]Stephen Walther 著,汤涛译. ASP. NET 揭秘(第二版)[M]. 北京:中国电力出版社, 2004.

3 蔡晓路,梁宇奇. Web Services 技术、架构和应用[M],北京:电子工业出版社,2003.

4 李安渝等, Web Services 技术与实现[M],北京:国防工业出版社,2003.