基于 SOAP 扩展的 Web Service 应用研究

杨 瑞 （淮海工学院电子工程系 江苏连云港 222005）
蔡 虹 （淮海工学院计算机科学系 江苏连云港 222005）

摘要：利用 SOAP 扩展机制，可以实现更改往返于 Web Service 的 SOAP 消息来增加 Web Service 的功能，弥补 SOAP 消息传输中的不足。在此基础上本文分别实现了 SOAP 消息的压缩和加密，达到扩充 Web Service 功能，改善传输效率和安全性的目的。

关键词：SOAP 扩展 Web Service 压缩 加密

1 引言

SOAP 是 Web Service 在分布式环境中交换信息的轻量级协议。SOAP 本身并没有定义程序语义，而是以 XML 的形式提供了一个简单且轻量，用于在分布式环境中交换结构化和类型化信息的机制。XML 的平台无关性使得 SOAP 消息可以用于任何平台和系统。在实际应用中，SOAP 消息的传输还存在着许多不足，如效率低，安全性不足等。利用 SOAP 扩展机制，可以更改往返于 Web Service 客户端/服务端的 SOAP 消息来增加 Web Service 的功能，以弥补 SOAP 消息传输中的不足。本文利用 SOAP 扩展机制实现了用户对线上传输的 XML/SOAP 消息的处理和控制。

2 SOAP 扩展原理

SOAP 扩展是. NET 提供的可扩展性机制。是一种在线上传送过程中获取 SOAP 消息的架构，对象转换为 XML 的过程被称为串行化，从 XML 中解构对象的过程被称为并行化。完成这一转换的模块称为 XMLSerializer。SOAP 扩展允许用户在对象与 XML 的转换过程中访问数据，这样对于输出请求/响应，可以在 SOAP 消息流串行化后，在线上发送给接收方的前后实现用户访问。对于接收的请求/响应，则在并行化 SOAP 消息流的前后访问它。利用这个原理，可以实现 SOAP 消息的用户级控制，达到扩充功能的目的。其工作原理如图 1 所示。

SOAP 扩展对 SOAP 消息内容的改变在不影响并行化操作的情况下可以单独出现。否则这种改变必须

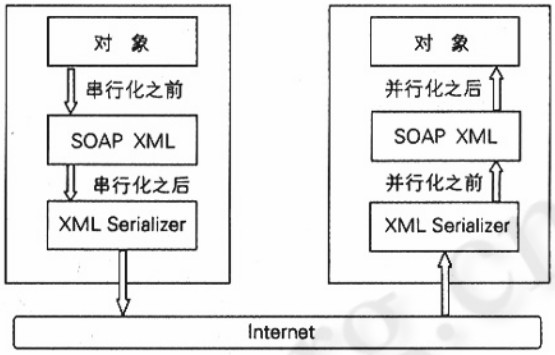

图 1 SOAP 扩展工作原理图

在 Client 和 Server 端同时进行，以免造成 SOAP 消息的并行化错误。实现 SOAP 扩展的步骤为：

（1）定义一个 SOAP 扩展类，该类继承自 SoapExtension 类。

（2）保存代表 SOAP 消息的 Stream 引用类型。

（3）初始化 SOAP 扩展的数据。

（4）在 SOAP 扩展执行阶段处理 SOAP 消息。

（5）配置 SOAP 扩展，使其与特定 Web Service 方法通信。

3 SOAP 扩展的具体实现

3.1 利用 SOAP 扩展实现数据压缩

当网络中传输数据的内容是文本的时候，通过压缩可以使文本的大小减少达 80% 。这就意味着在客户端和服务端之间带宽的需求也可以减少类似的百分

比。在 Web Service 应用中，客户端调用 Web service 的方法后，在网络上传输的 SOAP 响应可能是一个很大的数据集。特别是在数据库调用中，SOAP 响应可能会包含所有的数据，这可能是个很大的值。所以在传输之前，我们通过 SOAP 扩展压缩要传输的文本内容来减少数据流量，提高传输效率。

为了减少数据流量，要在发送端 SOAP 数据串行化之后对其进行压缩，而在接收端 SOAP 数据并行化之前对数据进行解压缩。这个过程必须在客户端和服务器上都进行。也就是说，如果要在客户端上运行 SOAP 扩展并压缩 SOAP 消息，则相应的 SOAP 扩展必须在服务器上对该 SOAP 消息进行解压缩。SOAP 扩展的实现利用创建带有派生自. NET Framework 中的两个抽象类 SoapExtensionAttribute 和 SoapExtension，这两个类中定义的一些方法需要用户自己实现。SoapExtension 中要重写的函数主要有：GetInitializer ( )、Initialize ( )、ProcessMessage ( )、ChainStream ( )。执行顺序为：

(1) 如果是第一次用 Web Service 执行 SOAP 扩展，则对 SOAP 扩展调用 GetInitializer ( ) 方法。GetInitializer ( ) 具有多态性参数分别为 Type 和 LogicalMethodInfo，SoapExtensionAttribute。

(2) 调用 public abstract void Initialize (object initializer) 方法，每次对 Web Service 方法发出 SOAP 请求时都会调用 Initialize，它传递在 GetInitializer 中初始化的 Object。

(3) 调用 public virtual Stream ChainStream (Stream stream) 方法，ChainStream 确保具有最高优先级的 SOAP 扩展可以修改与通过网络发送或返回的 SOAP 消息最接近的实际数据，SOAP 扩展应保存传入 ChainStream 的 Stream 和从 ChainStream 返回的 Stream 的引用。

(4) 调用 public abstract void ProcessMessage (SoapMessage message) 方法，SOAP 扩展的大多数工作在 ProcessMessage ( ) 方法中完成。SOAP 扩展在序列化的各个阶段都调用该方法，被定义的阶段是：

BeforeSerialize：在 SOAP 请求或者响应被串行化到 XML 之前。

AfterSerialize：在 SOAP 请求或者响应被串行化到 XML 并发送之后。

BeforeDeserialize：在 SOAP 请求或者响应从 XML 并行化之前。

AfterDeserialize：在 SOAP 请求或者响应从 XML 并行化之后。

为了实现数据压缩，需使用 AfterSerialize 和 BeforeDeserialize 两个阶段，在 AfterSerialize 阶段实现对数据的压缩，而在 BeforeDeserialize 阶段实现对数据的解压缩，工作流程图如图 2 所示。

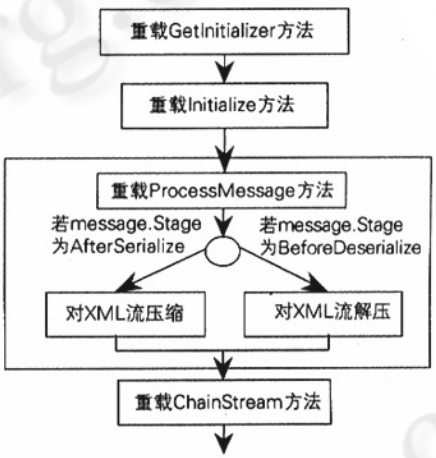

图 2　利用 SOAP 扩展实现数据压缩的程序流程

实现的关键代码如下：

```
...
public override void ProcessMessage ( SoapMessage message)
    {
        switch ( message. Stage)
        {
          case SoapMessageStage. BeforeSerialize:
            break;
          case SoapMessageStage. AfterSerialize:
            zipped( message) ;
            break;
          case SoapMessageStage. BeforeDeserialize:
            unzipped( message) ;
            break;
          case SoapMessageStage. AfterDeserialize:
```

```
            break;
        default:
            throw new Exception("invalid stage");
        }
    }
...
public override Stream ChainStream( Stream stream )
    {
        oldStream = stream;
        newStream = new MemoryStream();
        return newStream;
    }
...
```

使用压缩的方法虽然可以提高数据传输的效率，但也会带来相应的问题，如果使用的压缩算法平台间不通用，则会牺牲 Web Service 的跨平台性。

3.2 利用 SOAP 扩展实现加密

当 SOAP 消息绑定在 HTTP 上进行传输时，为了提高数据传输的安全性，可以利用类似 HTTPS 的补充协议进行加密，整个 HTTP 信息包括头部和 HTTP 协议的正文，都使用了公开不对称加密算法加以加密。但这种加密依赖于 HTTP 协议，而且会对整个消息进行加密。此外 SOAP 是一个独立的传输协议，还可以使用 SMTP、TCP、UDP 等进行传送。所以，当 SOAP 消息需要在公共网络上路由，并且只加密部分重要的数据时，HTTPS 的补充协议就无法实现，而利用 SOAP 扩展来实现用户自己的 SOAP 数据加密可以完成这一功能。

利用 SOAP 扩展实现数据加密的工作原理和上述数据压缩的工作原理类似，所不同的就是在 AfterSerialize 阶段调用相应的加密算法对 SOAP 信息进行加密，而在 BeforeDeserialize 阶段调用解密算法对 SOAP 信息解密。加密和解密算法可以使用公开的 DES 算法也可以使用不对称加密算法，即客户端使用服务端的公开密钥对传送的 SOAP 信息进行有选择的加密，而服务端使用私有密钥对收到的加密数据进行解密。

4 SOAP 扩展与 Web Service 的结合使用

SOAP 扩展需要配置才能够和 Web Service 结合使用，利用配置可以实现多个 SOAP 扩展按不同优先级运行，配置可以通过属性值或配置文件进行。使用属性值进行配置时，需要在每个应用 SOAP 扩展的 Web Service 方法中添加一个属性值，该属性值从 SoapExtensionAttribute 类继承而来。SoapExtensionAttribute 具有两个属性：ExtensionType 和 Priority。SOAP 扩展在 ExtensionType 属性中返回该 SOAP 扩展的类型。Priority 属性表示该 SOAP 扩展的相对优先级。

利用配置文件配置 SOAP 扩展，则 Web Service 的所有方法在配置文件定义的范围内调用 SOAP 扩展。这时要将名为 soapExtensionTypes 的 XML 元素添加到 App. config 或 Web. config 配置文件的 webServices 部分。在 soapExtensionTypes 元素中，需要添加名为 add 的 XML 元素，add 元素具有的属性如表 1 所示。

表 1 add 元素属性说明表

属 性	含 义
type	SOAP 扩展的类型以及它所驻留的程序集
priority	SOAP 扩展在其组中的相对优先级
group	SOAP 扩展所在的组

5 结束语

SOAP 扩展提供了一种强大的方法，使用户可以在 Web Service 中进行自定义操作。利用 SOAP 扩展，除了可以实现 SOAP 信息流的压缩、加密，还可以建立一些如路由、日志记录和计费处理等用户功能，利用 SOAP 扩展可以弥补 SOAP 消息传输的不足，达到扩充 Web Service 功能，改善传输效率和安全性的目的。

参考文献

1 郑小平编著，.NET 精髓，Web 服务原理与开发，人民邮电出版社，2002 -1。

2 Russ Basiura，Mike Batongbacal 等著，康博译，ASP. NET WEB 服务高级编程，清华大学出版社，2002 -6。

3 [美]Scott Seely 著，杨涛 、杨晓云等译，SOAP：XML 跨平台 Web Service 开发技术，机械工业出版社，2002 -4。

4 柴晓路著，Web Services：技术，架构和应用，电子工业出版社，2003 -1。