XML与CORBA的集成

侯孟书* 赫俊民 胡卫东 杨 帆 唐雪飞

(电子科技大学计算机科学与工程学院 成都 610054)

【摘要】 阐述了CORBA与扩展标记语言集成的背景和特点，分析了集成的字符串、DOM/Value映射、SOAP、Web服务的四种实现方式，对各种方式的优缺点进行比较后，指出Web服务方式是实现CORBA与扩展标记语言集成的理想方式，并提出了一种紧耦合的CORBA应用与松耦合的Web应用相结合的方法。

关 键 词 扩展标记语言； 公共对象请求代理； 简单对象访问协议； Web服务

中图分类号 TP311.133.1

Integrating XML and CORBA

Hou Mengshu He Junmin Hu Weidong Yang Fan Tang Xuefei

(College of Computer Science and Engineering, UEST of China Chengdu 610054)

Abstract This paper studies the reason of integrating XML and CORBA. The characteristics of XML and CORBA specification are described. There are four modes of integrating XML and CORBA: XML as Strings, DOM/Value mapping, SOAP and Web services, then analyses four implement methods and their weakness, point out the Web services is the best methods which provides the method to integrate the loose-couple Web application with the tight-couple CORBA application.

Key words extensible markup language; common object request broker architecture; simple object access protocol; Web services

Internet/Intranet的兴起及蓬勃发展极大地改变了人们的生活、学习和工作方式，网络提供给人们的不仅是大量的信息和娱乐活动，而且也带来了无限商机。随着电子商务的深入发展，企业信息系统比以往任何时候更加依赖分布式计算架构。扩展标记语言(XML)SGML的子集，具有良好的自描述性、灵活性、可扩展性，非常适合Web上的数据交换和发布，而CORBA定义了分布式对象之间通信所需要的完整体系结构，如何将企业已有的CORBA应用与面向消息的、松散耦合的Web技术结合起来,使企业更好地适应新经济的发展,是很多企业分布式应用面临的一个重要课题,XML与CORBA集成为解决该课题提供了一个有益的思路。

1 CORBA的特点

CORBA是对象管理组织OMG在其对象模型体系结构OMA框架下以对象请求代理ORB为核心制定的分布式对象处理标准，它定义了对象之间通过ORB透明发送请求接收响应的机制，保证在分布异质环境下对象之间的互操作性[1]。

2002年4月25日收稿

* 男 31岁 硕士生

OMG 的目标是推动对象技术的理论和实践在软件行业中的应用，特别是在分布式应用系统方面，为实现这一目标，OMG 为面向对象的应用提供一个公共框架，如果符合这一框架，就可以在多种硬件平台和操作系统上建立一个异质的分布式应用环境。

OMA 包括两部分：对象模型和参考模型，对象模型定义如何描述分布异质环境中的对象，参考模型描述对象之间的交互，参考模型由以下几部分组成：应用界面、域截面、公共设施、对象服务和对象请求代理 ORB，其参考模型如图1所示。图中 ORB 独立于语言、平台，相当于软件总线，连接另外几个部分。ORB 之间通过 IIOP 协议连接[2]。

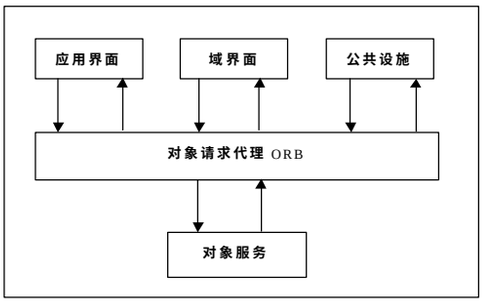

图1 OMA 的参考模型

2 XML 的特点

可扩展标记语言 XML 包含三要素：文档类型定义 DTD、可扩展样式语言 XSL 和可扩展链接语言 Xlink[3]。DTD 定义了 XML 文件中的元素、元素的属性以及元素和元素属性之间的关系；名字空间实现统一的 XML 文档数据表 示以及数据的相互集成；XSL 是用于规定 XML 文档呈现样式的语言，它使得数据与其表现 形式相互独立；而 XLink 将进一步扩展目前 Web 上已有的简单链接。

XML 文档本身只描述数据内容，它的显示功能由 XML 样式单来完成。W3C 正式推荐的样式单标准有两种：一种是层叠样式单 CSS；另一种是可扩展样式单语言 XSL。

处理 XML 文档需要 XML 解析器，解析器读入 XML 文档检查其合法性并进行处理。许多解析器既提供 SAX 接口又提供 DOM 接口。SAX 是一个为基于事件的 XML 解析器定义的 API，它允许程序和脚本动态地访问和更新 XML 的内容、结构和文档风格。DOM 是一个为基于树型的 XML 解析器定义的 API，它允许程序将 XML 数据构建成对象并且允许对象间相互结合、访问、操纵。

3 集成方式

3.1 字符串方式

OMG IDL 的一个缺点就是对 CORBA 应用的版本支持不够。假设写一个错误跟踪系统，在 IDL 中定义一个 Bug 类型如下：

```
struct Bug {
    long bugnum;
    string synopsis;
    string owner;
};
```

错误跟踪系统的服务器通过上述结构可以查看客户端提交的错误信息，随着用户的增加，系统可能想知道错误信息的来源，于是更改 Bug 结构如下：

```
struct Bug {
    long bugnum;
    string synopsis;
    string owner;
    string reported_by;    // added field
};
```

当发生上述变化时，服务器和客户端的应用程序都需要重新编译、安装，以适应这种结构的变化。当然可以通过版本声明、定义新的结构来解决，但仍存在很多问题。

而在 XML 中可以定义 Bug 数据如下：

```
<bug>
    <bugnum>49938</bugnum>
    <synopsis>DynStruct broken</synopsis>
    <owner>vinoski</owner>
    <reported_by>schmidt</reported_by>
</bug>
```

当在 XML 中扩展 Bug 数据时，不会造成任何问题，因此用 XML 定义数据提供了更好的灵活性。由于 XML 是文本的，显然可以将 XML 作为字符串在 CORBA 系统中传送。定义如下：

```
interface BugTracker {
    exception NoSuchBug { long bugnum; };
    typedef string Bug; // assume XML string contents
    Bug get_details(in long bugnum) raises(NoSuchBug);
    // ...
};
```

使用这种方式可以解决结构变化带来的 CORBA 应用版本问题，但仍有以下两方面不足：

1) 不能保证 get_details 返回有效的 XML，ORB 只能保证返回的值是字符串类型，而这个值是不是有效的 XML 字符串，ORB 却无能为力。

2) 效率低，服务器通过在内存创建 DOM 树实现 get_details 对数据的访问，需要在 DOM 树和 XML 字符串之间进行转换，其应用承担了编码和解码工作。

3.2　DOM/Value 映射方式

在 CORBA 系统中用字符串方式传输 XML 数据是脆弱低效的，可以采用另外一种格式来表示 CORBA 系统中的 XML 数据。XML 本身就是结构化的数据，例如，上面用 XML 定义的 Bug 数据中，Bug 元素有四个子结点 bugnum、synopsis、owner 和 reported_by。因此，在 CORBA 系统中，可以用 XML 树来表示 XML 数据。在 IDL 中用结构(structs)和序列(sequences)表示树结构，如在 IDL 中可以这样表示树结点：

```
struct XMLElement;
  typedef sequence<XMLElement> XMLElementSeq;
  struct XMLAttr {
      wstring name;
      wstring value;
   };
   typedef sequence<XMLAttr> XMLAttrSeq;
   struct XMLElement {
      wstring name;
      wstring value;
      XMLAttrSeq attributes;
      XMLElementSeq children;
      // ...
   };
```

这种在应用中构造 XML 树是可行的，但显得臃肿且易于产生错误。DOM/Value 映射方式解决了上述问题。2001年4月 OMG 发布了一个新规范“XMLDOM：DOM/Value Mapping Specification”。

和 DOM 类似，该规范使用户通过解析器提供的 API 在内存中创建和遍历 XML 解析树，不同的是，DOM/Value 用 IDL 值类型(valuetypes)表示 XML 解析树中的结点，而 DOM 是用 IDL 界面(interfaces)表示 XML 解析树中的结点。由于 IDL 值类型支持方法实现，结点拥有自己的方法实现，因此不需要将结点的数据成员公布给所有的应用。更重要的是 IDL 值类型传值而不是传地址，应用能直接将 XML 数据作为树结构发送和接受，而不需要转化为字符串格式。这对应用来说，避免了低效的字符串格式转换，提高了应用性能。

3.3 SOAP 方式

SOAP 是基于 XML 的协议，通常建立在 HTTP 之上，用于应用间消息传递。与 IIOP 用二进制表示消息数据不同，SOAP 用 XML 表示消息内容。SOAP 为在一个松散的、分布的环境中使用 XML 对等地交换结构化信息提供了一种简单且轻量级的机制。

SOAP 由四部分组成：

1) SOAP 信封：一个整体的表示框架，用于描述消息的内容，以及如何处理。

2) SOAP 编码规则：一个编码机制，用于交换应用程序定义的数据类型的实例。

3) SOAP RPC 表示：一个用于表示远端过程调用和响应的约定。

4) SOAP 绑定：一个使用底层传输协议来完成在结点间交换 SOAP 信封的约定。

OMG 提出的“扩展传输框架”就是希望用标准的方法在 ORB 中插入不同的传输协议，使 CORBA 具有包容多种协议的能力。显然通过该框架可以将 SOAP 插入 ORB。事实上，已经有了对 SOAP 支持的 ORB。SOAP 作为可插入协议具有以下优点：

1) 穿越防火墙(Firewall traversal)：防火墙总是尽可能少地打开端口，一般情况下，80 端口(HTTP 的保留端口)是唯一能够使用的端口，这也是 Web 使用的端口。IIOP 没有保留端口，不易穿越防火墙，所以很多 ORB 产品采用自己的方法来处理防火墙问题，如：VisiBroker 的 GateKeeper 和 IONA 的 WonderWall。SOAP 通常建立在 HTTP 之上，故可用 HTTP 隧道技术(HTTP tunneling)使 SOAP 消息穿越防火墙。当然，网管人员仍可以通过配置防火墙阻止恶意的数据和请求通过 SOAP 进入系统。

2) 与非 CORBA 系统的互操作：与 DCE、CORBA、COM、J2EE 不同，SOAP 得到了更普遍的支持，其在互连网上的优势是其他传输协议所不能比拟的。基于 SOAP 的系统也越来越多，这些系统与基于 CORBA 的系统进行互操作日益重要。但是由于对象模型和应用语义的不同，两个系统间的低层次互操作效率很低。

3.4 Web 服务方式

Web 服务是通过 SOAP 消息调用的，通过 WSDL 进行界面描述的，以及通过 UDDI 进行公共注册发布的。Web 服务发送和接受的是 XML 编码消息，通过 Web 服务与 CORBA 的集成实现 XML 与 CORBA 的集成。

3.4.1 Web 服务具有的特征

Web 服务具有以下特征：

1) 能通过 HTTP、SMTP 协议访问。

2) 支持松耦合的面向服务架构 SOA(Service-Oriented Architectures)，在这种架构中，服务方在目录服务器中发布自己提供的服务和联系方式，客户方在目录服务器中查找所需服务对象，并与服务对象建立绑定。SOA 一般隐藏了双方的实现细节，诸如编程语言和操作系统等。

3) 发送和接受的是 XML 编码消息

3.4.2 Web 服务代表了三个领域的发展和集成方向

三个领域的发展和集成方向如下：

WWW：Web 服务代表了 Web 的发展趋势，即从面向浏览的互动系统到大规模 A2A(application

-to-application)集成系统。

传统中间件：虽然 Web 服务代表了传统中间件的下一步发展方向，但是 Web 服务的基本概念来自成熟的传统中间件。

电子数据交换 EDI：B2B 集成使贸易双方能在完全标准化的商业文档格式和商业交易上进行商务活动，这些商务活动并不需要人们全程干预。

3.4.3　Web 服务与 CORBA 集成方式

1) 用 Web 服务实现 CORBA 对象

在这种方式下，CORBA 对象完全隐藏了它的实现细节。客户通过服务器端的对象伺服(servant)调用一个或多个 Web 服务，对客户而言就象调用一个 CORBA 对象。对象伺服将客户请求的数据转换为适合访问 Web 服务的数据，然后又将 Web 服务返回的数据转换成客户识别的响应数据。将一个或多个 Web 服务封装为 CORBA 对象，其主要工作是实现 CORBA 应用模式(CORBA application choreographies)和 Web 服务应用模式(Web application choreographies)之间的映射。在面向集成的中间件(integration-oriented middleware)中，封装和映射是非常普遍的。

2) 用 CORBA 对象实现 Web 服务

这种方式和上面的方式类似，但更通用。由于可以通过 Web 访问 Web 服务，因此 Web 服务技术被看做比 CORBA 技术更通用。一个 Web 服务(通常作为一个 J2EE servlet)通过“后台”(back-end)的 CORBA 对象来实现其功能。而客户端并不知道如何封装 CORBA 对象，也不知道如何映射不同应用间的数据。

4　结 束 语

在四种实现方式中，字符串方式解决了 CORBA 应用中的版本问题，由于应用负担了 DOM 与 XML 字符串之间的转换，效率较低，目前已有实现该方法的产品。DOM/Value 映射方式在内存建立和遍历 XML 解析树，采用 IDL 值类型定义 XML 树结点，避免了 XML 解析树转化为 XML 字符串给应用所带来的负担。SOAP 作为 CORBA 的可插入协议，实现 XML 与 CORBA 的集成。Web 服务方式是 XML 与 CORBA 集成较好的方式，这种方式将紧密耦合的、高效的 *N* 层计算技术与面向消息的、松散耦合的 Web 技术相结合，为 CORBA 技术和 XML 技术提供了广阔的发展前景。

参 考 文 献

1　Michi H, Steve V 著. 基于 C++CORBA 高级编程. 徐金梧译. 北京: 清华大学出版社, 2000

2　R.otte, Patrick P, Roy M 著. CORBA 教程. 李师贤译. 北京: 清华大学出版社, 2000

3　Steven H 著. XML 完全探索. 师夷工作室译. 北京: 中国青年出版社, 2001