摘 要

学 科 名 称：计算机软件与理论

论 文 名 称：基于 XML 的安全通信—XML Engine

硕士研究生： 邓彦杰

导 师：周明天

随着 Web Service 技术的流行，作为 Web Service 基础协议的 XML 变得越来越重要。同时，XML 作为一种事实上的数据表示标准，越来越多的公司在通过网络传输结构化数据时采用 XML，XML 文档的安全性变得愈加重要。

XMLEngine（基于 XML 的安全通信平台）是出于对 XML 消息安全功能的设计考虑，为将来网络中基于 XML 信息的 Web 服务提供了安全而有效的信息保障。

本课题设计并实现了基于 XML 的信息安全通信平台。首先我们提出了 XMLEngine（中文名称 XML 引擎）这个基本概念，介绍了研发的技术背景，在 XML1.0 规范和 SOAP 规范 1.0.2 版本的基础上，总体描述了在开发过程中所涉及到的 XML、SOAP、Web Service、应用密码学及 XML 安全的基本知识，从而给出了 XMLEngine 体系结构的模型，并在 LINUX 平台下详细阐述了实现 XMLEngine 的方法和步骤。

在本项目中，本人承担了构建 Web Service 典型应用的任务，该应用是基于 J2EE 设计和实现的，应用完成后经过测试完全达到了设计要求，本文详细给出了该应用的设计和实现过程。

关键词 XML 安全 XML 引擎 加密/解密 签名/验证 Web Service 应用

Abstract

With the popularization of Web Services, XML, which is the basic protocol of Web Services， becomes more and more important. At the same time, XML as a de facto standard of data presentation, more and more company adopts XML when transmitting structure data through network, so the security of XML document becomes more important than ever before.

XMLEngine (XML-based Information Security platform) is designed to provide the secure and effective guarantee to the Web Services based on XML in the Internet.

The research subject accomplished the design and implement of the information security communication platform based on XML. The basic conception of XMLEngine and the technical background of the subject are presented firstly. Then, the basic principle of XML、SOAP、Web Services、applied cryptography and XML security involved in the subject are described. In addition, the thesis introduces the XMLEngine architecture and explaines the implement method and step of XMLEngine under LINUX platform .

In the project, I am responsible for the development of Web Services application which adopts J2EE architecture. Now, the application achieves the requirement of design and the thesis illustrates the process of the application design and implementation in detail.

Keywords

XMLSecurity XMLEngine Encrypt/Decrypt Signature/Verify Web Service

独 创 性 声 明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

签名：邓秀志　　日期：2004年12月15日

关于论文使用授权的说明

本学位论文作者完全了解电子科技大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权电子科技大学可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后应遵守此规定）

签名：邓秀志　　导师签名：

日期：2004年12月15日

第一章 引言

1.1. 课题背景

1.1.1 作为 Web Service 基础的 XML 的安全性越来越重要

构建大型应用时，通常要在不同计算机上进行并发工作，这样更加有效，甚至可能是必需的。强大的计算机和网络带来了分布式计算的现象，把计算分布在网络上，而不是只在中央计算机上进行。N 层应用程序把应用程序分布在多台计算机上。例如，三层应用程序可能把用户界面放在一台计算机上，把业务逻辑处理放在一台计算机上，数据库放在一台计算机上，它们在应用程序运行时进行交互。

为了让分布式系统正确工作，整个网络中不同计算机上执行的应用程序组件（通常包装成编程对象）要进行通信。许多公司和组织开发了自己的分布式组件间通信技术。对象管理组织 OMG 的公用请求代理体系结构（Common Object Request Broker Architecture，CORBA）、微软公司的分布式组件对象模型（Distributed Component Object Model，DCOM）、Sun 公司的 Internet 对象请求代理间协议上的远程方法调用（Remote Method Invocation/Internet Inter-ORB Protocol ,RMI/IIOP）和 IBM 公司的分布式对象模型（Distributed System Object Model ,DSOM）等，这些技术的每一个都能使不同语言、不同实现版本和不同地点运行的程序能够像在同一计算机上一样进行通信。

但是，这些技术的相互之间的互操作性（与不同厂家和平台的软件之间共享数据和相互通信的能力）非常有限。例如，两个最流行的技术 DCOM 和 CORBA 就很难相互通信。DCOM 和 CORBA 组件通常通过 COM/CORBA 桥进行通信. 如果 DCOM 与 CORBA 的基础协议发生改变，则编程人员必须修改这个桥，反映所作的这些改变。另外，这些分布式对象协议在 Intranet 环境下工作得很好，而将其用于公共的 Internet 就会有很多问题，为了解决安全问题，许多组织都在其公共的 Web 服务器和访问这些服务器的大众之间架设一道防火墙。由于防火墙的存在影响了分布式对象协议的使用。这些问题影响了分布式计算帮助业务过程集成与自动化的能力。

基于这些原因出现了 Web Service 技术，Web Service 技术解决了互操作性有限的问题，从而提高了分布式计算的功能。与 DCOM 和 CORBA 不同的是，Web

Service 用开放的标准（非专属标准）进行操作。理论上，Web Service 可以让任何两个软件组件相互通信，不管组件用什么技术生成，放在什么平台上。另外，基于 Web Service 的应用程序也更容易调试，因为 Web Service 用基于文本的通信协议而不是 DCOM 与 CORBA 采用的二进制通信协议。

随着 Web Service 技术的流行，作为 Web Service 基础协议的 XML 变得越来越重要。同时，由于 XML 是数据事实上的表示标准，越来越多的公司在通过网络传输结构化数据时采用 XML，XML 文档的安全性变得愈加重要。

1.1.2 传统的信息安全不能满足 XML 安全的需要

象其它任何文档一样，可以将 XML 文档整篇加密，然后安全地发送给一个或多个接收方。例如，这是 SSL 或 TLS 的常见功能，但是更令人感兴趣的是如何对同一文档的不同部分进行不同处理的情况。XML 的一个有价值的好处是可以将一整篇 XML 作为一个操作发送，然后在本地保存，从而减少了网络通信量。但是，这就带来了一个问题：如何控制对不同元素组的授权查看。商家可能需要知道客户的名称和地址，但是，无需知道任何正在使用的信用卡的各种详细信息，就像银行不需要知道购买货物的详细信息一样。可能需要防止研究人员看到有关个人医疗记录的详细信息，而管理人员可能正好需要那些详细信息，但是应该防止他们查看医疗历史；而医生或护士可能需要医疗详细信息和一些（但不是全部）个人资料。

XML 语言的强项之一是，搜索是明确的，无二义性的：DTD 或 Schema 提供了相关语法的信息。如果将包括标记在内的文档的一部分作为整体加密，就会丧失搜索与那些标记相关的数据的能力。此外，如果标记本身被加密，那么一旦泄漏，它们将被利用对采用的密码术进行纯文本攻击。

XML 是因特网以及近来 Web 服务持续增长和开发的主要支持者。但是,在实现 XML 语言的全部能力之前，还有许多与安全性相关的工作要做。目前，加密整个 XML 文档、测试其完整性和确认其发送方的可靠性是一个简单的过程，但是，越来越有必要对文档的某些部分也使用这些功能，以便以任意顺序加密信息和认证鉴别不同用户或发起方；同时与 HTML 相比，XML 的出现也使得数字签名的实施与安全传送能够得以实现。在与 XML 相关的安全性领域方面的开发规

范中最重要部分是 XML 加密、XML 签名、XACL 和 SAML 等。

1.2. 作者的主要工作

作者首先对 XML 引擎（XMLEngine）系统所涉及到的 XML 及 SOAP 规范、应用密码学知识和 W3C 有关 XML 安全的标准进行简要介绍，对 PKI 的基本知识进行概要说明，并在此基础上提出一个完整可行的 XMLEngine 模型，并在 LINUX 平台上实现这一模型，对具体的实现步骤和细节进行详细描述。作为一个新产品应该与具体的应用结合起来，为此我们专门设计和实现了一个网络数字社区－网上书店，将 XMLEngine 服务器明的部署到网络中，对其中诸如用户信用卡号和密码等敏感信息进行加密和签名，以体现充分体现 XMLEngine 的特点和优势。

在整个过程中作者主要从事其中的 XML 解析器、典型应用模块的需求分析、概要设计、详细设计、代码编码和测试工作，与小组成员一起参与 XMLEngine 整个项目的总体设计和规划，最后的总体测试和部署。应该来讲基于 XML 的应用越来越多，但针对 XML 的安全提出的时间不长，虽然国内外对其的研究已经不少，但针对其开发的应用产品还不多见。在 XML 安全的标准和一些关键问题上一方面仍有广阔的研究空间，另一方面 XMLEngine 作为一个大型系统，它的相关标准庞杂，涉及多方面的知识和技术，在有了第一阶段的产品后还有进一步完善和扩展。

1.3. 各章节安排

第一章，引言。主要介绍课题的来源和背景，作者的主要工作，文章的章节安排等。

第二章，XML 概述。介绍了 XML 概念、语法、格式、解析、显示等相关理论知识。

第三章，Web Service 概述。介绍了 Web Service 概念、SOAP、WSDL、UDDI、等相关理论知识。

第四章，XML 安全和 SOAP 安全基础。本章在第二章、第三章的基础之上，主要是针对 W3C 所提出的有关 XML 加密和解密、XML 签名和验证的规范，进行详细介绍。

第五章，XML 安全平台－－XML Engine 的设计。首先给出 XMLEgnine 相关的定义，然后介绍了 XMLEngine 服务器的结构、应用环境和处理流程，从系统架构角度分析了 XMLEngine 的应用和部署等。

第六章，构建 Web Service 典型应用－－数字书店，用以测试 XMLEngine 在实

际应用中的功能和效果。首先介绍了本应用使用的技术：MVC 设计模式概念、JAX—RPC 概念、SOAP 工具包 AXIS 概念等。然后介绍本应用在系统中的地位、作用、设计目标、功能等。最后从两个方面（一个方面是客户和服务器端；另一个方面是 MVC 的 M、V、C 三部分）详细给出设计和实现步骤，并通过抓包的方式测试应用的设计效果和 XMLEngine 服务器的工作状况。

第七章，今后的工作。本章中作者阐述了系统存在的不足以及系统未来的发展思路。

1.4. 小结

本章介绍课题的研究背景，作者在课题中所做的主要工作，以及全文的章节安排。下几章将对开发 XMLEngine 产品过程中所涉及到的相关基础概念体系、技术框架以及研究应用现状做一个综述。

第二章. XML 概述

2.1 什么是 XML

XML 是 Extensible Markup Language 的简写，是一种扩展性标记语言。XML 并不是标记语言。它只是用来创造标记语言（比如 HTML、化学标记语言 CML、数学标记语言 MathML）的元语言。

XML 来源于 SGML，SGML 全称是“Standard Generalized Markup Language”（通用标记语言标准）。它是标志语言的标准，也就是说所有标记语言都是依照 SGML 制定的，当然包括 HTML。SGML 的覆盖面很广，凡是有一定格式的文件都属于 SGML，比如报告，乐谱等等，HTML 是 SGML 在网络上最常见的文件格式。

而 XML 就是 SGML 的简化版，只不过省略了其中复杂和不常用的部分。那么有了 HTML，为什么还需要用 XML？XML 是可扩展的，而 HTML 是静态的有限的标记集无法满足日益增长的数据描述要求，XML 大大丰富了 HTML 的描述功能，可以描述非常复杂的 Web 页面，如复杂的数学表达式，化学方程式等，而 HTML 只能描述数据的显示样式，没有语义，因此在电子数据交换、查询数据库中的数据等方面存在极大不足。HTML 中数据与其显示样式完全混在一起的，数据的可重用性差，如果要换显示形式你不得不重新编码所有这样的 HTML 文件。XML 是结构化的，好处是使 XML 文件易于被程序处理，便于数据共享。

2.2 格式良好的 XML 文件—XML 语法

在 XML 中，“格式良好”有着明确的标准，即是要遵守 XML1.0 规范中的语法规则。XML 必须符合规范，才能被正确解释处理。

一个 XML 文件最基本的构成是：

- XML 声明（也是处理指示的一种）
- 处理指示（可选）
- XML 元素

2.2.1 XML 声明

一个 XML 文件，最好以一个 XML 声明作为开始。之所以说“最好”，是因为 XML 声明在文件中是可选内容，可加可不加，但 W3C 推荐加入这一行声明。

一个完整的 XML 声明是这样的：

<?xml version = "1.0" standalone = "no" encoding = "GB2312"?>

version 属性：指明所采用的 XML 的版本号

standalone 属性（缺省为 yes）：这个属性表明该 XML 文件是否和一个 DTD 文件文件配套使用。

encoding 属性（缺省为 UTF-8）：编码标准。如下是几种常见的编码：简体中文码：GB2312、繁体中文码：BIG5、西欧字符： UTF-8。如果标签是用中文来写的，就要用 encoding = "GB2312"。

2.2.2 XML 元素

元素是 XML 文件内容的基本单元。从语法上讲，一个元素包含一个起始标记、一个结束标记以及标记之间的数据内容。其形式是：

〈标记〉数据内容〈/标记〉

元素中还可以再嵌套别的元素。一个 XML 文档有且只有一个根元素。起始和结束标记之间出现的所有合法字符都是数据内容，都被忠实地传给 XML 处理程序。比方说 ：

<格式>一段文字</格式>	〈格式> 一段文字 </格式>
A	B

上表中左边和右边所列出的元素标签及内容均一致，但二者却是不同的，因为 B 中多了两个换行符，在屏幕上显示的时候分成了三行，所以象换行、回车这类不可见字符也是作为 XML 文档中的有效字符来处理。

1、字符引用

为了避免把数据内容和标记中需要用到的一些特殊符号相混淆，XML 还提

供了一些有用的字符引用。字符引用实际上也是实体引用（后面介绍）。常见的5 种字符引用如下表所示：

字符	实体引用
>	>
<	<
&	&
"	"
'	'

这样，如果我们需要在“示例”这个元素中出现文本“<姓名>张三</姓名>”，正确的写法应该是：

〈示例〉 <姓名>张三</姓名>〈/示例〉

2、标记

所有符号“〈”和符号“〉”之间的内容都称为标记。基本形式为：

〈标记名 （属性名=“属性取值” ）〉

XML 对于标记的语法规定要比 HTML 要严格得多。标记必不可少，任何一个形式良好的 XML 文件中至少要有一个元素。大小写有所区分，在 HTML 中，标记〈HELLO〉和〈hello〉是一回事，但在 XML 中，它们是两个截然不同的标记。 要有正确的结束标记，结束标记除了要和开始标记在拼写和大小写上完全相同，还必须在前面加上一个斜杠“/” 。为了简便起见，空标记直接在开始标记的最后惯以斜杠“/”来确认。

标记与标记之间要正确嵌套，比如：sample是错误的，正确

写法是：sample。标记命名要合法，一般标记应该以字母、下划线“_”或冒号“:”开头，后面跟字母、数字、句号“.”、冒号、下划线或连字符“-”，但是中间不能有空格，而且任何标记不能以“xml”起始。另外，最好不要在标记的开头使用冒号，尽管它是合法的，但可能会带来混淆。在 XML1.0 标准中允许使用任何长度的标记，不过，现实中的 XML 处理程序可能会要求标记的长度限制在一定范围内。

对于标记中的属性，是指对标记的进一步描述和说明，标记中可以包含任意多个属性。在标记中，属性以名称/取值对出现，属性名不能重复，名称与取值之间用等号“=”分隔，且取值用引号引起来。例如：

```
<author sex="male">Alex</author>
```

用属性“sex”来指明“author”的性别是“male”。

XML 中属性是自己定义的，虽然属性使用比较方便，但是有时候属性不易扩充和被程序操作，所以凡是用到属性的地方也可以用子元素来代替，例如上面的代码可以改成这样：

```
<author>Alex
    <sex>female</sex>
</author>
```

2.2.3 处理指示

处理指示是用来给处理 XML 文件的应用程序提供信息的。也就是说，XML 分析器可能对它并不感兴趣，而把这些信息原封不动地传给 XML 应用程序。然后，这个应用程序来解释这个指示，遵照它所提供的信息进行处理，或者再把它原封不动地传给下一个应用程序。正如我们前面看到的，XML 声明就是一个处理指示。

所有的处理指示应该遵循下面的格式：

〈？处理指示名 处理指示信息？〉

由于XML声明的处理指示名是“xml”，因此其它处理指示名不能再用“xml”。例如，我们使用一个处理指示来指定与这个 XML 文件配套使用的样式单的类型

及文件名：

<?xml-stylesheet type="text/xsl" href="mystyle.xsl"?>

2.2.4 实体

实体主要是用来代替字符数据的，它可以节省大量的录入工作。实体同样必须要格式良好。假如你为你的信件署名定义了一个实体 lettersign，它代表下面这一大段文本：

张三

某网络公司销售部门

北京市海淀区中关村 88 号，100000

那么以后当你的 XML 文件中出现“信件”元素时，就可以这样写：

〈信件〉

〈收件人〉李四〈/收件人〉

〈主题〉hello〈/主题〉

〈正文〉hello！ &lettersign ；〈/正文〉

〈/信件〉

1、实体的类型

实体包括两种类型：一般实体和参数实体。

- 一般实体

定义一般实体的格式如下：

<!ENTITY 实体名 "文本内容"〉

另外，你也可以指定一个实体代替一个外部文件的内容，此时要使用 SYSTEM 这个关键字。例如 <!ENTITY lettersign SYSTEM "http://www.mydomain.com/lettersign.xml"〉

- 参数实体

与一般实体相同，参数实体既可以是内部的也可以是外部的。不过，参数实体只用在 DTD 中。参数实体的格式与一般实体很类似，只不过中间要加上“%”符，例如：

<!ENTITY % 实体名 "文本内容">

2、实体的使用

实体的使用包括两部分：实体声明和实体引用。

- 实体声明

对于实体的声明应该放在文件类型 DOCTYPE 中。DOCTYPE 一般放在文件头（即 XML 声明和 DTD）之后，XML 元素之前。这样一来，XML 文件就变为下面的形式：

```
<?xml version="1.0"?>
<!DOCTYPE 文件根元素名 [实体声明部分]>
<文件根元素名>
    具体数据内容
</文件根元素名>
```

- 实体引用

实体引用指的是引用一个在实体声明中已经声明过的一个实体，实体引用的形式很简单：&实体名。实体引用一般有如下几点规则：

- 在引用 XML 实体之前，必须已经在 XML 文件中对此实体进行过声明；
- 在实体引用中不能出现空格。也就是说，& lettersign; 的用法会引起错误。
- 尽管在一个实体中可以再引用其它实体，但是不能出现循环引用。也就是说，一个实体不能引用它自己；同样，也不能出现实体 A 引用实体 B，然后实体 B 再反过来引用实体 A 的情况。
- 实体引用不能在 DOCTYPE 声明中出现。
- 实体引用的文本必须是形式良好的 XML。

完整的实体声明和实体引用的例子：

```
<?xml version="1.0"?>
<!DOCTYPE 联系人列表 [
   <!ENTITY A 公司地址 “北京市五街 1234 号”>
   <!ENTITY B 公司地址 “上海南京路 9876 号”>
]>
<联系人列表>
```

```
<联系人>
    <姓名>张三</姓名>
    <公司>A 公司</公司>
    <地址>&A 公司地址</地址>
</联系人>
<联系人>
    <姓名 gt;李四</姓名>
    <公司>B 公司</公司>
    <地址>&B 公司地址〈/地址〉
</联系人>
 <联系人>
    <姓名>王五</姓名>
    <公司>B 公司</公司>
    <地址>&B 公司地址〈/地址〉
 </联系人>
 </联系人列表>
```

从这个例子中可以看出，一旦哪个公司搬家了，只须改变实体声明中有关该公司的地址，所有这个公司的联系人的地址也就都改过来了。

2.3 有效的 XML 文档

格式良好的 XML 不一定是有效的 XML 文件，比如：

```
联系人列表>
  <联系人>
    <姓名>张三</姓名>
    <ID>001</ID>
    <ID>002</ID>
    <公司>A 公司</公司>
</联系人>

<联系人>
  <姓名>李四</姓名>
  <ID>002</ID>
  <公司>B 公司</公司>
```

</联系人>
</联系人列表>

这个 XML 文档是格式良好的，但是不是有效的，因为：ID 号被赋予两个公司、而且两个人的 ID 号是一样的，冲突了。这就是结构良好的 XML 文档没有语义的问题，DTD 和 Schema 就是解决这个问题的。

2.3.1 DTD 和 Schema

1、DTD

文件类型描述 DTD（Document Type Definition）是用来描述一个标记语言的语法和词汇表，也就是定义了文件的整体结构以及文件的语法；DTD 规定了一个解析器为了解释一个“有效的”XML 文件所需要知道的所有规则的细节；一个有效的 xml 文件不允许使用任意的标记。使用的任何标记都要在 DTD 内声明，而且必须以 DTD 允许的方式使用。所以 DTD 就是用来解决这类有效性问题的。

2、使用 DTD 的方法

①内部 DTD（standalone=“yes”）

在 XML 文件的序言部分加入一个 DTD 描述，加入的位置是紧接在 XML 处理指示之后。

```
<?xml version = "1.0" encoding="GB2312" standalone = "yes"?>
<!DOCTYPE 根元素名[
   元素描述
]>
文件体.....
```

②外部 DTD（standalone=“no”）

外部 DTD 的好处是：它可以方便高效地被多个 XML 文件所共享。你只要写一个 DTD 文件，就可以被多个 XML 文件所引用。

XML 声明中必须说明这个文件不是自成一体的，即 standalone 属性的属性值是 no。同时，在 DOCTYPE 声明中，应该加入 SYSTEM 属性：

```
<?xml version = "1.0" encoding="GB2312" standalone = "no"?>
<!DOCTYPE 根元素名 SYSTEM "外部 DTD 文件的 URL">
```

③公用 DTD

外部 DTD 一个系统内可共享的 DTD。目前，已经有数量众多的写好的 DTD

文件可以利用。针对不同的行业和应用，这些 DTD 文件已经建立了通用的元素和标签规则。你不需要自己重新创建，只要在他们的基础上加入你需要的新标识。这就需要公用 DTD。两个相同行业不同地区的人使用同一个 DTD 文件来作为文档创建规范，那么他们的数据就很容易交换和共享。引用公共 DTD 的形式为：

<!DOCTYPE 根元素 PUBLIC "DTD 名称" "外部 DTD 的 URL">

3、Schema

DTD 最大的局限是不能对元素内容提供足够的控制，例如，DTD 无法指定<today>09/01/2000</today>有效而<today>Eggs ，Toast，Coffee</today>无效。相对于 DTD，我们总结 Schema 的优势有如下几条：

- 一致性

实际上 DTD 是 XML 体系中的异类， 它的书写结构和 XML 文件的结构仿佛有着天壤之别，后者清晰直观，前者复杂晦涩。Schema 建立在 XML 之上，本身也是一种 XML，使用者不必再为了搞懂 DTD 而去重新学习；另一方面，它可以被现有的 XML 编辑制作工具所编辑、被 XML 语法分析器所解析、被 XML 应用系统所利用，既有投资得到了最大程度的保护。

- 扩展性

虽然 DTD 中也定义了一些数据类型，但那都是针对属性类型而定义的，而且类型非常有限。没有数据类型，无形之中大大增加了程序员的开发难度和工作量。何况电子交易过程中不可避免地会出现大量的数据转换，整型、实型、布尔型、日期型的数据层出不穷，DTD 显然招架不住。Schema 对 DTD 进行了扩充，引入了数据类型，很好地解决了这一问题。

- 易用性

XML Schema 取代 DTD 的另一个原因要归结于 DOM 和 SAX（关于 DOM 和 SAX 的概念将在后面章节中详细论述）。作为一种 XML API，DOM 和 SAX 只是对 XML 实例文档有效，对于 DTD 则无能为力，你不可能期望通过 DOM 或 SAX 来判定一个元素的属性类型或者某个元素的子元素允许出现的次数（当然，这都是 XML 分析器的本职工作）。但是，一旦有了 Schema，这个问题便不复存在，因为此时对于 XML 文档结构的描述已变成为 Schema--一种“格式良好的”XML 文档，用 DOM 和 SAX 去访问当然不在话下啦。

- 规范性

同 DTD 一样，Schema 也提供了一套完整的机制以约束 XML 文档中标记的使用，但相比之下，后者基于 XML，更具有规范性。Schema 利用元素的内容和属性来定义 XML 文档的整体结构，如哪些元素可以出现在文档中、元素间的关系是什么、每个元素有哪些内容和属性、以及元素出现的顺序和次数等等，一目了然

2.4 XML 文件的解析

处理 XML 文档的应用程序都需要一个 XML 解析器，XML 解析器的作用是：

- 文档中提取实际的数据，然后创建一系列的事件或者是新的数据结构
- 解析器也能够检查文档是否是格式良好的,也就是说是否严格的遵守了 XML 规范，而这是处理 XML 文档之前就必须进行的工作
- 解析器还应该能够判断一个文档是否是有效的（如果该文档有一个 DTD/Schema 的话）

当前比较常见解析器有：IBM XML4J、Apache Xerces、Sun Project X、Microsoft MSXML(IE5.0 以上版本已经集成了 MSXML 解析器)、Oracle XML Parser for Java 以及 James Clark XP 等。

2.5 XML 文件的显示

XML 文档本身是重内容而不重形式。XML 的显示一般用样式单来描述文档的显示效果。样式单是一种专门描述结构文档表现方式的文档，它既可以描述这些文档如何在屏幕上显示，也可以描述它们的打印效果，甚至声音效果。样式单一般不包含在 XML 文档内部，而以独立的文档方式存在。W3C 给出了两种样式单语言的推荐标准，一种是层叠样式单 CSS（Cascading Style Sheets），另一种是可扩展样式单语言 XSL（eXtensible Stylesheet Language）。如图 2.1 就是一个用 IE 打开的未有样式单简单的例子：

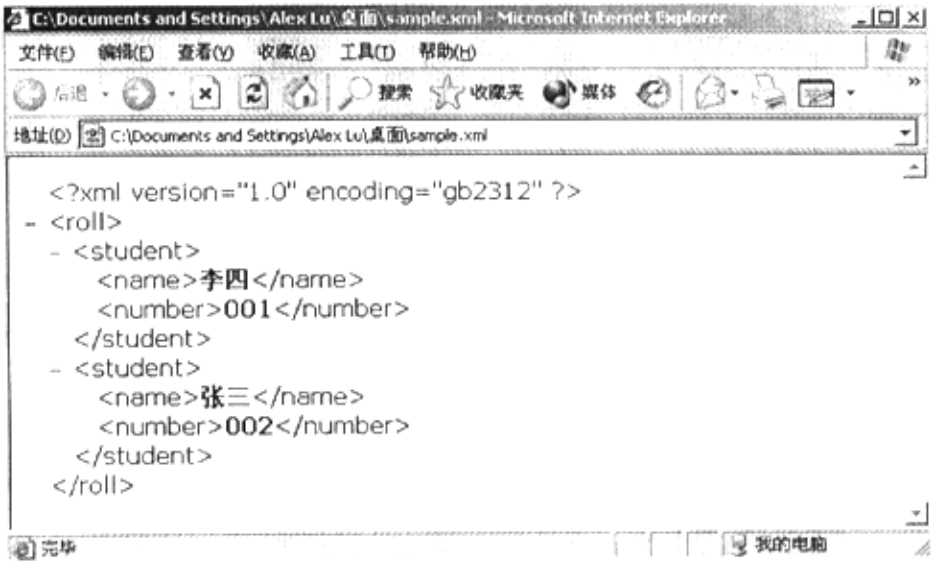


图 2.1 用 IE 打开的没有样式单的 XML 文档

2.6 应用程序接口

处理 XML 文档是由解析器来完成的，应用程序要创建、访问和操作一个 XML 文件就需要一个 XML 解析器的接口，W3C 和 XML_DEV 邮件列表成员分别提出的两个标准应用程序接口：DOM 和 SAX。

DOM 和 SAX 在应用程序开发过程中所处地位可以用下图 2.2 来表示：

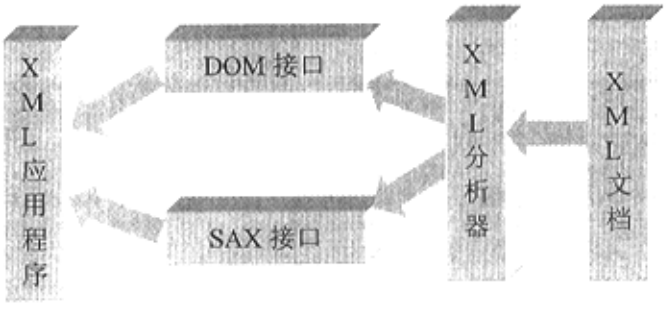


图 2.2 DOM 和 SAX

从图中可以看出，应用程序不是直接对 XML 文档进行操作的，而是首先由

XML 分析器对 XML 文档进行分析，然后，应用程序通过 XML 分析器所提供的 DOM 接口或 SAX 接口对分析结果进行操作，从而间接地实现了对 XML 文档的访问。

1、DOM

DOM 的全称是 Document Object Model，也即文档对象模型。在应用程序中，基于 DOM 的 XML 分析器将一个 XML 文档转换成一个对象模型的集合（通常称 DOM 树），应用程序正是通过对这个对象模型的操作，来实现对 XML 文档数据的操作。通过 DOM 接口，应用程序可以在任何时候访问 XML 文档中的任何一部分数据，因此，这种利用 DOM 接口的机制也被称作随机访问机制。

DOM 接口提供了一种通过分层对象模型来访问 XML 文档信息的方式，这些分层对象模型依据 XML 的文档结构形成了一棵节点树。无论 XML 文档中所描述的是什么类型的信息，即便是制表数据、项目列表或一个文档，利用 DOM 所生成的模型都是节点树的形式。也就是说，DOM 强制使用树模型来访问 XML 文档中的信息。由于 XML 本质上就是一种分层结构，所以这种描述方法是相当有效的。

2、SAX

SAX 的全称是 Simple APIs for XML，也即 XML 简单应用程序接口。与 DOM 不同，SAX 提供的访问模式是一种顺序模式，这是一种快速读写 XML 数据的方式。当使用 SAX 分析器对 XML 文档进行分析时，会触发一系列事件，并激活相应的事件处理函数，应用程序通过这些事件处理函数实现对 XML 文档的访问，因而 SAX 接口也被称作事件驱动接口。

3、DOM 和 SAX 的比较

DOM 树所提供的随机访问方式给应用程序的开发带来了很大的灵活性，它可以任意地控制整个 XML 文档中的内容。然而，由于 DOM 分析器把整个 XML 文档转化成 DOM 树放在了内存中，因此，当文档比较大或者结构比较复杂时，对内存的需求就比较高。而且，对于结构复杂的树的遍历也是一项耗时的操作。所以，DOM 分析器对机器性能的要求比较高，实现效率不十分理想。不过，由于 DOM 分析器所采用的树结构的思想与 XML 文档的结构相吻合，同时鉴于随机访问所带来的方便，因此，DOM 分析器还是有很广泛的使用价值的。

SAX分析器在对XML文档进行分析时，触发了一系列的事件，由于事件触发本身是有时序性的，因此，SAX提供的是一种顺序访问机制，对于已经分析过的部分，不能再倒回去重新处理。SAX之所以被叫做"简单"应用程序接口，是因为SAX分析器只做了一些简单的工作，大部分工作还要由应用程序自己去做。也就是说，SAX分析器在实现时，它只是顺序地检查XML文档中的字节流，判断当前字节是XML语法中的哪一部分、是否符合XML语法，然后再触发相应的事件，而事件处理函数本身则要由应用程序自己来实现。同DOM分析器相比，SAX分析器缺乏灵活性。然而，由于SAX分析器实现简单，对内存要求比较低，因此实现效率比较高，对于那些只需要访问XML文档中的数据而不对文档进行更改的应用程序来说，SAX分析器更为合适。

第三章. Web Service 概述

3.1 基本概念

Web Service 是能够基于网络，尤其是基于万维网（World Wide Web）直接调用的能够处理离散任务或连续任务的软件模型。目前较为流行的应用是，由一家公司对其专有数据进行封装，提供 Web Service，然后其它公司就可以通过 Internet 来动态使用这些在线服务。这为未来全球的电子商务发展提供了新的标准和架构。

Web Service 是独立的、模块化的应用，能够通过因特网来描述、发布、定位以及调用。在 Web Service 的体系架构中包括三个角色：服务提供者(Service Provider)、服务请求者(Service Requestor)、服务注册器(Service Registry)。角色间主要有三个操作：发布(Publish)、查找(Find)、绑定(Bind)。
图 3.1 清楚的描述了三种角色，以及角色之间的作用关系。

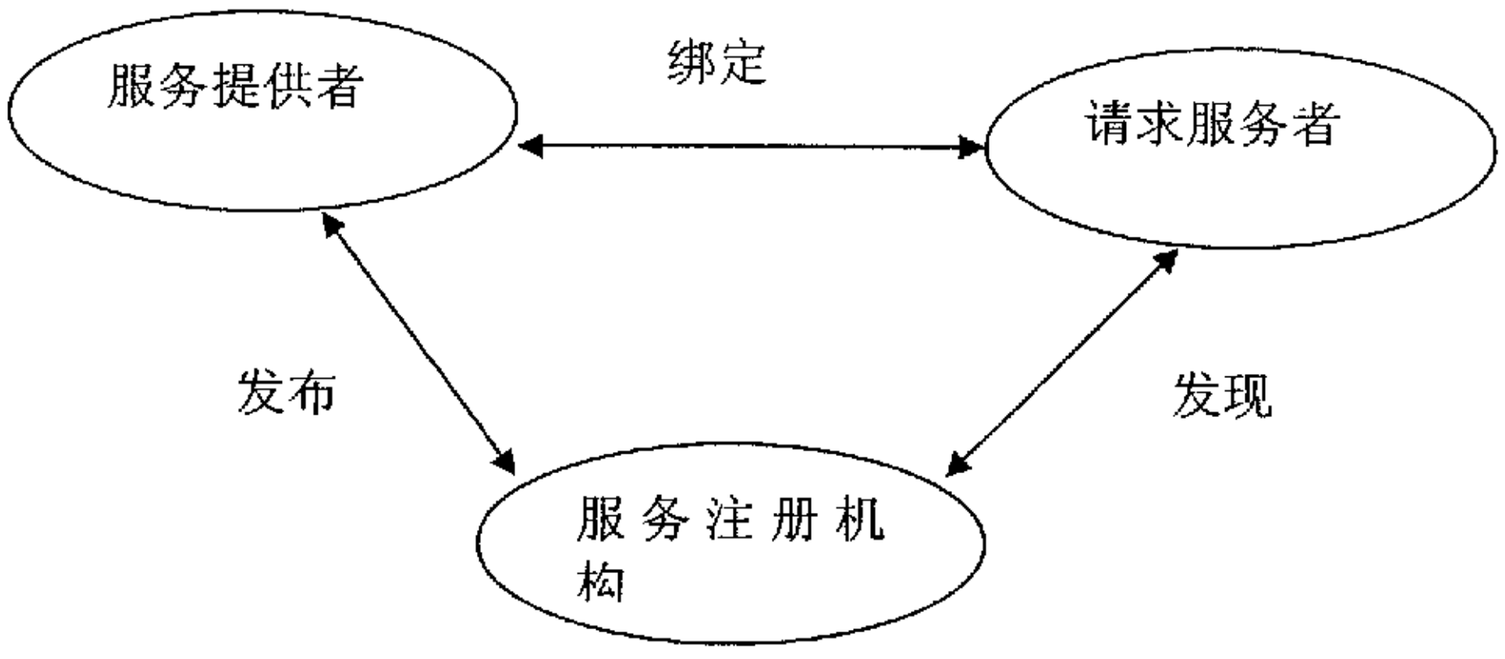


图 3.1Web Service 关系图

Web 服务是位于应用程序代码和应用程序之间的一个接口。它的作用相当于一个抽象层，将应用平台与编程语言相关的细节（如怎样调用应用程序代码）等分隔开。这个标准化的抽象层意味着任何支持 Web 的服务的编程语言都可以访问应用程序提供的功能。Web 服务是以一种与平台无关的方式提供了这种跨平台的互操作性。

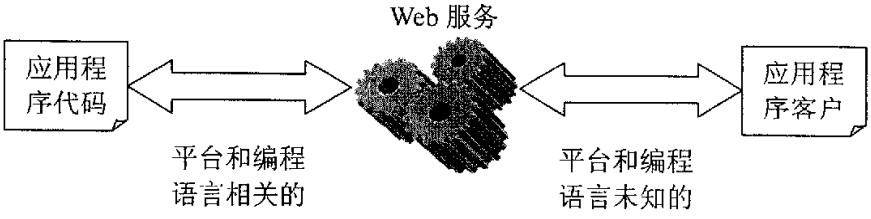


图 3.2 Web 服务在应用程序客户和应用程序代码之间提供了一个抽象层

3.2 Web Service 协议标准

3.2.1 简单对象访问协议（SOAP）

SOAP 是 Simple Object Access Protocol 的缩写，是一种基于 XML 的不依赖传输协议的表示层协议，用来在分散或分布式的应用程序之间方便地以对象的形式交换数据。在 SOAP 的下层，可以是 HTTP/HTTP，也可以是 SMTP/POP3，还可以是为一些应用而专门设计的特殊的通信协议。

SOAP 是序列化调用位于远程系统上的服务所需信息的标准方法，这些信息可以使用一种远程系统能够读懂的格式通过网络发送到远程系统，而不必关心远程系统运行于何种平台或者使用何种语言编写。SOAP 以 XML 格式提供了一个简单、轻量的用于在分散或分布环境中交换结构化和类型信息的机制。SOAP 本身并没有定义任何应用程序语义，如编程模型或特定语义的实现；实际上它通过提供一个有标准组件的包模型和在模块中编码数据的机制，定义了一个简单的表示应用程序语义的机制。这使 SOAP 可用于联合各种现有的网络协议和格式，包括 HTTP、SMTP 和 MIME，并可被用于消息传递到 RPC 的各种系统。

SOAP 解决了通过防火墙传送往返于远程应用程序的消息的问题。除了通过某些预先设定的作为特定用途的端口，防火墙通常禁止通过其它端口进行远程通讯。这就出现了一个问题，大部分分布式协议不使用分配的端口，而是动态地选择端口。微软 SOAP 技术实现的解决方案是通过 HTTP 的 80 端口传送对远程进程的调用。这个远程调用使用 XML 定义消息请求或响应的格式，把调用附加到 HTTP 协议的顶部。这个技术的优点之一就是降低通过防火墙传送消息的复杂性。但是 80 端口通常还用来作为 Web 通信之用，所以可能会降低其效率。

SOAP 可以用来解决因特网应用程序的交互性问题。你可以使用一种平台无

关性方式在远程（或本地）服务器上访问对象和服务。现在的互联网世界由不同的操作系统、不同的防火墙、不同的产生远程过程调用的方法和平台组成。为了跨因特网交互，客户机和服务器都需要了解彼此的安全类型和信任、服务部署模式和实现细节以及平台语言。使用 SOAP，这种平台特定性的混乱局面就会结束。基于已被业界广泛接受的 HTTP 标准和 XML 标准，SOAP 也可与其竞争对象 RPC 技术连通，并提供用于任何操作系统、程序语言和平台的轻量级消息格式。

在 SOAP 体系结构有四个主要的部分：

SOAP 信封（envelope），用于描述消息内容和处理方法。

一个 SOAP 消息包含一个信封（envelope），信封可以包含一个可选的标题（header）和一个必需的正文（body），如图 3.3 所示。

图 3.3 SOAP 消息结构

SOAP 编码规则：定义了一个编码机制用于交换应用程序定义的数据类型的实例。

SOAP RPC 表示，定义了一个用于表示远程过程调用和响应的约定。

SOAP 绑定，定义了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。

简单的说，SOAP 提供了使用完全独立于平台的访问服务、对象和服务器的

技术。通过 SOAP，你将能够查询服务、调用服务、与服务通讯并处理服务，而不用去关心远程系统的位置、所在的操作系统或平台到底是什么样的。

3.2.2 Web Service 描述语言（WSDL）

WSDL 是 Web Service Description Language 的缩写，该语言将网络服务定义成一个能交换消息的通信端点集，为分布式系统提供了帮助文档，同时也可作为自动实现应用间通信的解决方案。

Web 服务定义语言(Web Services Definition Language，WSDL)是一个建议性标准，用于描 述 Web 服务的技术调用语法。WSDL 定义了一套基于 XML 的语法，将 Web 服务描述为能够进行消息交换的服务访问点的集合，从而满足了这种需求。WSDL 服务定义为分布式系统提供了可机器识别的 SDK 文档，并且可用于描述自动执行应用程序通信中所涉及的细节。WSDL 的当前版本是 1.1，规范可以从 http://www.w3.org/TR/wsdl 获得。

WSDL 就是描述 XMLWeb 服务的标准 XML 格式，WSDL 由 Ariba、Intel、IBM 和微软等开发商提出。它用一种和具体语言无关的抽象方式定义了给定 Web 服务收发的有关操作和消息。就其定义来说，你还不能把 WSDL 当作一种对象接 口定义语言，例如，CORBA 或 COM 等应用程序体系结构就会用到对象接口定义语言。 WSDL 保持协议中立，但它确实内建了绑定 SOAP 的支持，从而同 SOAP 建立了不可分割的联系。

WSDL 服务描述是一个 XML 文档，它与 WSDL 模式(schema)的定义一致。WSDL 文档并不是完整的服务描述，而只包括了服务描述任务的较低层次，即：服务接口的原始技术描述。WSDL 是 Web 服务的接口定义语言 IDL（Interface Definition Language，），本质上，WSDL 描述说明的是 Web 服务的以下三个基本属性：

服务做些什么--服务所提供的操作(方法)。

如何访问服务--数据格式详情以及访问服务操作的必要协议。

服务位于何处--由特定协议决定的网络地址，如 URL。

3.2.3 统一描述、发现和集成协议（UDDI）

UDDI(统一描述、发现和整合)是一套基于 Web 的、分布式的、为 Web Service 提供的、信息注册中心的实现标准规范，同时也包含一组使企业能将自身提供的 Web Service 注册，以使别的企业能够发现的访问协议的实现标准。

通过使用 UDDI 的发现服务，企业可以单独注册那些希望被别的企业发现的自身提供的 Web 服务。企业可以通过 UDDI 商业注册中心的 Web 界面，或是使用实现了"UDDI Programmer's API 标准"所描述的编程接口的工具，来将信息加入到 UDDI 的商业注册中心。UDDI 商业注册中心在逻辑上是集中的，在物理上是分布式的，由多个根节点组成，相互之间按一定规则进行数据同步。当一个企业在 UDDI 商业注册中心的一个实例中实施注册后，其注册信息会被自动复制到其它 UDDI 根节点，于是就能被任何希望发现这些 Web 服务的人所发现。

第四章 XML 安全和 SOAP 安全基础

XML 是保证数据的可移植性的一种有效可行的技术，XML 安全是将“应用安全”运用到 XML 结构上去的应用，是把数据保密和认证应用到 XML 结构上去的应用。

SOAP 是用于数据交换的轻量级的以 XML 为基础的协议。它促进了由远程过程调用和响应产生的数据传送。它设计成用于分布式和远程应用程序中，并且它是 Web 服务的主要组件。SOAP 提供了包含消息及其处理信息的信封。由于这个信封内容很机密，所以安全性就是您必须解决的问题。XML 加密对此问题提供了无缝的解决方案。

SOAP 本身是 XML，它让您使用 XML 加密以任何合适的方式随意处理加密问题。例如，您可以决定对整个 SOAP 主体加密，也可以对部分主体加密。

本论文中所有有关 XML 的技术都源自由 W3C 所提出的标准，W3C 提出了多种互操作技术（许多规范、指南、软件和工具等等）来引导 Web 充分发挥它的潜能，即作为 W3C 一种用于信息、商务、交流和共同理解的论坛。目前 W3C 中与 XML 安全直接有关的主要包括 XML 签名、XML 加密和 XML 密钥管理规范（XKMS），本项目也主要是围绕前两个方面进行研发的。

4.1 XML 签名和验证过程简介

依照 RFC，设计的 XML 签名带有多个目标，可提供“对任何数据类型的完整性、消息认证、或签名者认证服务，无论是在包括该签名的 XML 内部还是在别处。”

4.1.1 什么是 XML 签名

XML 签名其实是特定的语法，用于表示对任意的数字内容的数字签名，XML 签名本身就是一个 XML 文档，它具有规范化 XML 文档所具有的一切特性。XML 签名的高层生成过程是生成一个签名清单的规范化形式的 HASH 值，并通过一个强单向变换把签名清单的内容与一个钥联系在一起。

1、XML 签名与一般的数字签名的区别

XML 签名是具有结构化的、特定于上下文的和可扩展的特性。而一般的数字签名是分段的、与上下文无关的和严格的。

2、XML 签名类型

XML 的签名类型一般包括以下三种结构类型：

- 被封装（Enveloped Signature）的签名：

```
<original_document>
    <Signature>...    </Signature>
</original_document>
```

这一类型的特点是:原始文档是<Signature>的父元素

- 封装式（Enveloping Signature）的签名：

```
<Signature>
  <original_document>...</original_document>
</Signature>
```

这一类型的特点是：原始文档是<Signature>的子元素

- 分离式（Detached Signature）的签名：

```
<Signature>...    </Signature>
<original_document>...</original_document>
```

这一类型的特点是：原始文档和<Signature>之间没有父子关系

4.1.2 签名过程简介

生成和验证 XML 签名需要两步，签名生成的过程是核心生成，签名验证的过程是核心验证

1、XML 签名的结构

```
<Signature>
  <Signedinfo>
    <SignatureMethod/>
      (<Reference (URI=)?>
        (<Transforms>)?
        <DigestMethod>
```

```
            <DigestValue>
          </Reference>)+721
    </Signedinfo>
    <SignatureValue>
    (<KeyInfo>)?
    (<Object>)*
</Signature>
```

注：*：表示零个或者多个出现

+：表示一个或者多个出现

?：表示零个或者一个出现

2、元素描述

<Signature>：该元素标识了一个特定上下文环境中完整 XML 签名。在单个文件上下文环境中存在多个<Signature>实例的情况下，可以增加可选的 Id 属性来作为一个标识进行区分了。它一般包括两个实体：一个原始的文档或者原始文档的集合（<Signedinfo>）和一个实际的签名值（<SignatureValue>）

<Signedinfo>：包括实际签名的的所有信息，即签过名的信息

<Reference>：是 XML 签名的核心部分，是描述如何获得资源，如何在可能的情况下变换资源以生成被摘要的数据，随后又把数据签发成<SignedInfo>元素的一部分

<Transforms>：是一个或多个<Transform>元素的容器，<Transform>说明实施变换的类型，是以抽象的方式描述怎样获得被摘要的数据。它的工作原理是层叠的，是从一种变换输出，然后再输入到下一种变换，每下降一步就要应用一种新的变换，直到数据到达接收器

<DigestMethod>：是一个必须的元素用来指明将被用于签名对象的摘要算法

<DigestValue>：存放摘要的编码值，一般摘要总是使用 Base64 编码

<SignatureValue>：用于放置经过 Base64 编码后的数字签名实际值

<KeyInfo>：包括用来验证 XML 签名的具体信息，它能使接收者验证签名，而没有必要直接查找验证密钥

<Object>：是一个通用容器，包含其它有用的元素

2、核心生成过程

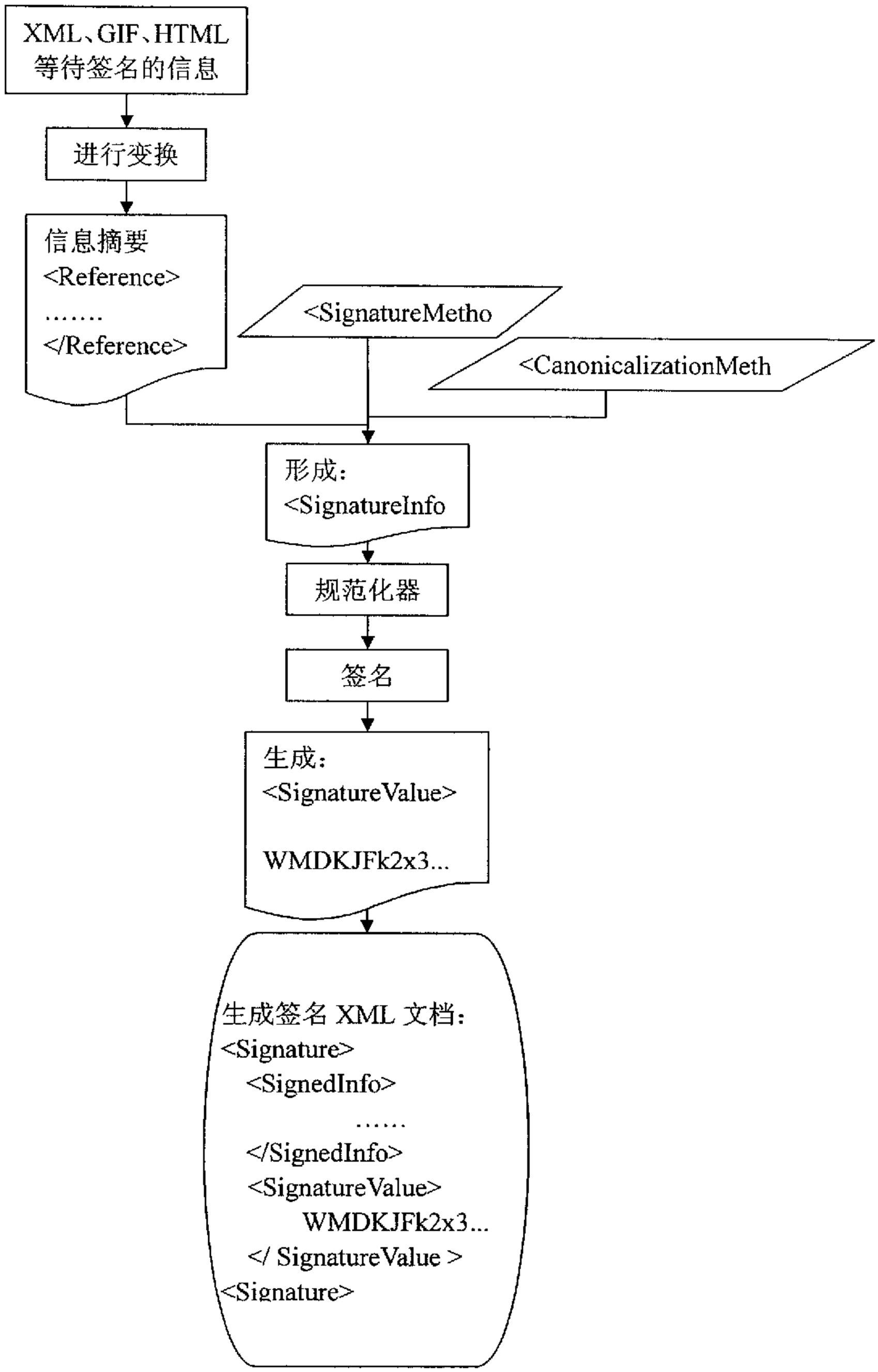


图 4.1 XML 签名的过程

流程图 4.1 大致地描述了一个 XML 签名的过程，对于核心生成包括两个方面：引用元素的生成和对 SignedInfo 签名后 SignatureValue 值的生成，它们的步骤分别描述如下：

①引用生成

对于每一种要被签名的资源：

引用变换（如果存在）

计算变换后资源的摘要值

创建<Refernece>元素。这包括可选属性、变换及表示摘要算法的标识符和实际摘要值

②签名生成

使用在引用生成中形成的引用元素，和指定的签名方法和规范化方法一起产生 SignedInfo 元素

对 SignedInfo 元素进行规范化操作，然后对 SignedInfo 元素按照 SignedInfo 中指明的算法进行计算等到 SignatureValue 元素的值

完成上述工作后，将得到的包括 SignedInfo、Object(s)、KeyInfo(如果存在)和 SignatureValueg 等构造成 Signature 元素

4.1.3 核心验证步骤

与签名相对应，验证也有类似的几个方法和步骤：

1、引用验证

首先，必须对<SignedInfo>元素进行规范化。对于每一个要验证的<Reference>，都有如下步骤：

- 通过对每一个<Reference>元素的 URI 属性引用解析来获得要待计算摘要的数据流。如果不存在任何 URI 属性，那么应用就应该知道数据源的位置。最后要待计算摘要的数据是可先选的层叠变换的结果。
- 对第 1 个步骤中得到的数据流计算摘要，此过程使用<DigestMethod>元素中指定的 HASH 函数来处理当前的<Reference>元素
- 将第 2 步骤中计算出来的摘要值和当前正处理的<Reference>元素中的

<DigestValue>元素的内容作比较，如果这些值不匹配，则引用验证失败。

2、签名验证

从<KeyInfo>元素或具体应用的密钥源中得到验证密钥。使用<SignatureMethod>的规范化确定正使用的签名算法，并对规范化形式的<SignedInfo>元素计算签名值。把签名值和<SignedInfoValue>元素内的值作比较。如果这些值不匹配，则签名失败。

4.2 XML 的加密和解密

4.2.1 基本概念及特点

XML 加密（XML Encryption）是使用 XML 句法定义的一种过程，用于加密数据和结果的表示。XML 加密为需要结构化数据安全交换的应用程序提供了一种端到端安全性。

XML 加密与 传统的 SSL/TLS 的不同之处在于，它涉及到了两个重要领域，即：加密交换数据的一部分和多方（不止两方）之间的安全会话。象其它任何文档一样，可以将 XML 文档整篇加密，然后安全地发送给一个或多个接收方。例如，这是 SSL 或 TLS 的常见功能，但是更令人感兴趣的是如何对同一文档的不同部分进行不同处理的情况。XML 的一个有价值的好处是可以将一整篇 XML 作为一个操作发送，然后在本地保存，从而减少了网络通信量。

与传统的加密方式相比，XML 可对远程 WEB 资源、本地文档，以及 XML 中的元素及元素内容分别进行加密；另外 XML 可以对同一文档中的不同部分采用不同的密钥加密，而普通的加密方法只能用单个密钥对整个明文进行加密。

4.2.2 XML 加密的方式

根据 XML 加密原理和规范，总结 XML 加密的方式有如下几种情况：

XML 使用 XML 加密对整个文档加密

加密前的原始数据是 XML 类型的文件， 并且 IANA 对 XML 的正式类型

定义是 ： http://www.isi.edu/in-notes/iana/assignments/media-types/text/xml。

```
<?xml version='1.0' ?>
<EncryptedData
   xmlns='http://www.w3.org/2001/04/xmlenc#'
Type='http://www.isi.edu/in-notes/iana/assignments/media-types/text
/xml>
...
```

- 对单个元素加密

此时 Type 属性的值应该是 http://www.w3.org/2001/04/xmlenc#Element，这里不再使用 IANA 类型了，相反，我们使用 XML 加密已指定这个类型

```
<?xml version='1.0' ?>
<Root_Element>
    <original_document>
    ...
    </original_document>
    <EncryptedData
       xmlns=' http://www.w3.org/2001/04/xmlenc#'
       Type=' http://www.w3.org/2001/04/xmlenc#Element' >
    ...
    </EncryptedData>
</EncryptedData>
```

说明：用<EncryptedData>来替换好加密的单个元素

- 加密元素的内容

Type 属性的值应该是：http://www.w3.org/2001/04/xmlenc#Content

```
<?xml version='1.0' ?>
<Root_Element>
    <original_document>
    ...
    </original_document>
    <encrypted_document>
        <EncryptedData
            xmlns='http://www.w3.org/2001/04/xmlenc#'
            Type=' http://www.w3.org/2001/04/xmlenc#Content' >
        ...
        </EncryptedData>
    </encrypted_document>
</EncryptedData>
```

说明：<encrypted_document>是待加密的元素标签，用<EncryptedData>替换元素的内容。

- 加密非 XML 数据

Type 属性值应该是:Type='http://www.isi.edu/in-notes/iana/assignments/*/*'。比如加密一个 JPEG 图片文件，此时 Type 属性的值就应该设置为：

Type='http://www.isi.edu/in-notes/iana/assignments/media-types/jpeg，其中加密方式是将按字节序列对整个 JPEG 文件进行加密，并将作为 CipherValue 元素的内容出现。

```
<?xml version='1.0' ?>
<EncryptedData
    xmlns='http://www.w3.org/2001/04/xmlenc#'

Type='http://www.isi.edu/in-notes/iana/assignments/media-types/jpeg
' >
    <CipherData>
        <CipherValue>...</CipherValue>
    </CipherData>
```

4.2.3 加解密过程简介

1、XML 加密的结构

```
<EncryptedData Id? Type?>
    <EncryptionMethod/>?
    <ds:KeyInfo>
        <EncryptedKey>?
        <AgreementMethod>?
        <ds:KeyName>?
        <ds:RetrievalMethod>?
            <ds:*>?
    </ds:KeyInfo>?
    <CipherData>
        <CipherValue>?
        <CipherReference URI?>?
    </CipherData>
    <EncryptionProperties>?
  </EncryptedData>
```

注：*：表示零个或者多个出现

+：表示一个或者多个出现

?：表示零个或者一个出现

2、元素描述：

EncryptedData：是 XML 加密句法中的基础元素之一

EncryptionMethod：主要功能是识别加密算法和可能的辅助参数，比如密钥长度、填允模式（针对非对称密码），或是加密模式。加密算法一般用 URI 标识符来表示

KeyInfo：描述了如何获得密钥从而对<CipjerData>元素的内容解密

EncryptedKey：可以被用来从一个发送者和一个已知的接收者之间传递加密密钥。它可以作为一个单独的 XML 文档使用，也可以加在应用程序中，或者是包含在元素<EncryptedData>中，作为 ds:KeyInfo 的一个子元素。

AgreementMethod：用来在数据加密时进行共享密钥的协商

KeyName：表示一些类型，它可能包含有接收者的名称等内容，目的是认证解密密钥的公开持有者（无论解密密钥是对称的，还是非对称的）

RetrievalMethod：是为了用特定的<ds:KeyInfo>元素把远程的验证密钥关联起来，

它指向解密密钥，尤其是指向<EncryptedKey>元素，为这一特定的<EncryptedKey>元素提供一种功能上的关联

CipherData：主要功能是以某种形式存储加密后的数据，如果是充当封装作用，那么就会有<CipherValue>子元素，它包含 Base-64 编码的密文数据；如果是充当引用作用，就会有<CipherReference>元素，它引用的是密文数据（加过密的）

CipherValue：存储加密后的数据

CipherReference：引用密文数据（加过密的）

EncryptionProperties：它包括一个或者多个<EncryptionProperty>子元素，这些子元素可以包括任何命名空间中的任意元素。标识符：Type=http://www.w3.org/2001/04/xmlenc#EncryptionProperties，可以被元素 ds:reference 中使用来识别引用的类型。有关元素 EncryptedData 和 EncryptedKey 产生时的附加信息项也可以放在一个 EncryptionProperty 元素中，如：时间/日期戳等。

4.2.4 加密器处理过程流程图

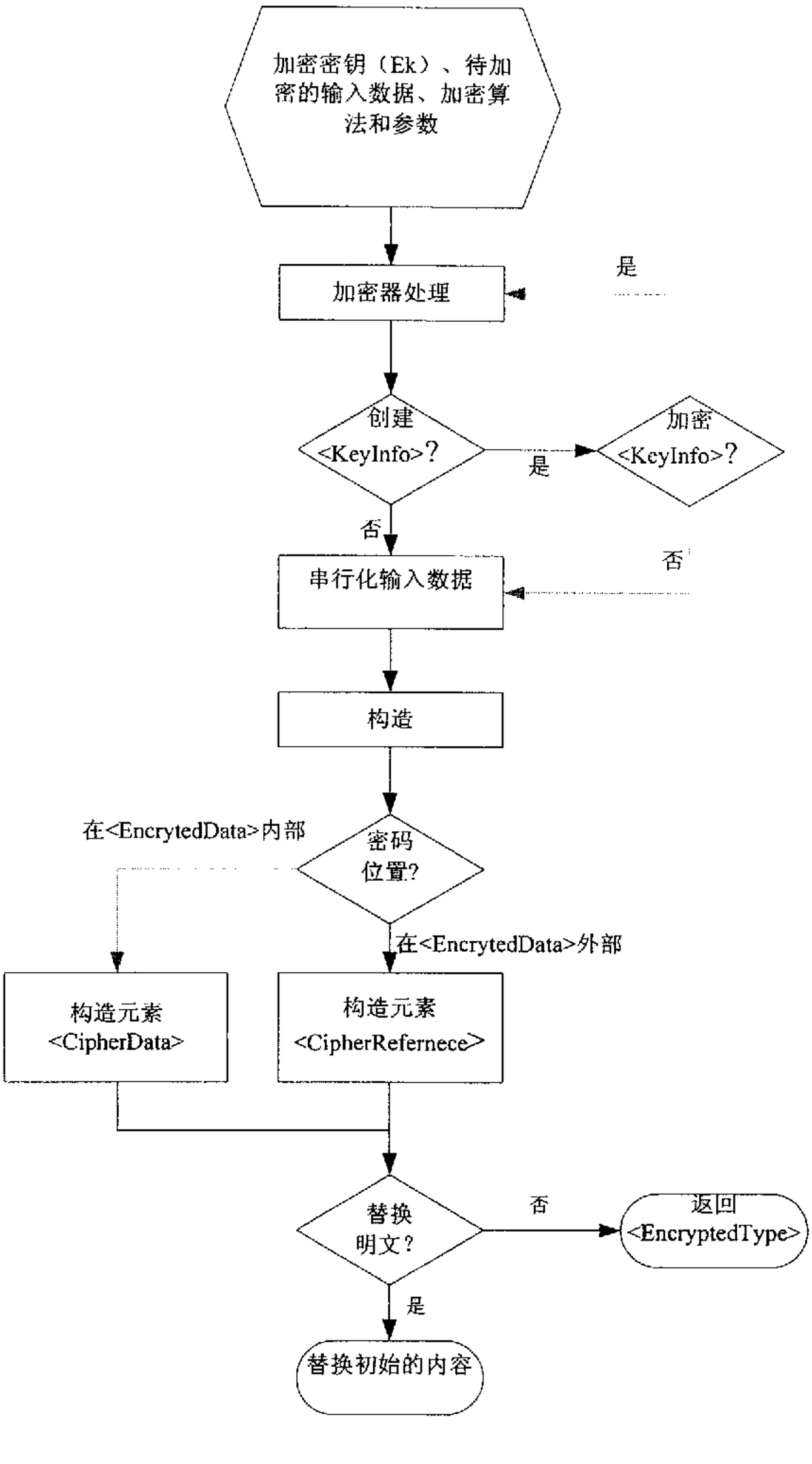


图 4.2 加密器处理过程流程图

4.2.5 解密器处理过程流程图

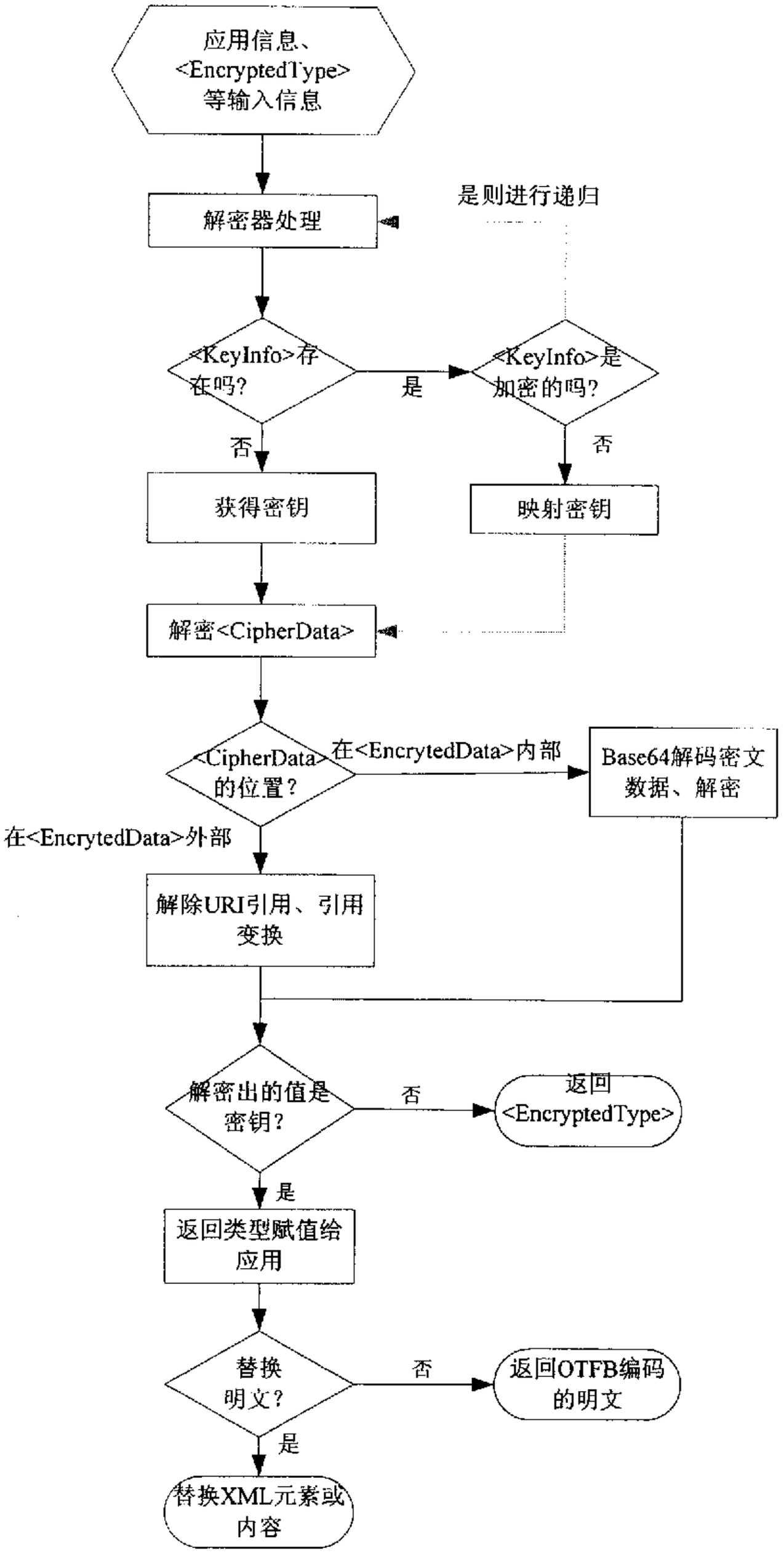


图 4.3 解密器处理过程流程图

第五章 XML 安全平台——XML Engine 的设计

简单一段安全程序或代码是无法与实际的应用相结合的，在如今应用为王的时代，即使国家将继续支持国内安全产业，但是链路级产品的市场仍然非常有限，并且受制于用户、运营商和集成商等，仅仅提供独立的应用模块是非常危险的布局，也不能最大限度的发挥作为一个安全产品所应起到的安全作用。

因此我们确立了最终要立足于为用户提供系统安全服务和解决方案的目标和内容，只有这样才能使公司实现较大的利润增长点，因此我们的产品应该是与应用系统整合在一起的、对用户是完全透明的、应用程序几乎不用作什么改动就能实现预定目标的整体解决方案。

在此我们提出了“IT 安全化”和规模化的方案，对于企业来讲，盈利的方式表现为面向最终用户，才能做到规模化，但必须提供低成本的产品，这点比较困难；面向面对大量最终用户的行业客户，如电信、银行，以点带面，提供“规模化”产品，最终用户可以由行业客户“赠送”，通过“服务”和行业客户的服务端产品获得收入；面向运营商，这类似于有大量最终用户的行业客户；面向有良好客户关系的集成商，它能将公司产品销售到这些客户身上。但是，应用安全所面临的问题是：应用千差万别，最终用户自身的要求也千差万别，很难一个产品满足所有这些用户的要求，这势必增加产品的成本，而且面临没有好评的结果；行业客户能提出自己的确定需求，但不会很多，需要公司先有方案，让其认可，再实施；但这样面临方案被抄袭给其他关系户的可能，带来不公平竞争和方案、产品的复制；应用掌握在集成商手里，用户也没办法左右，安全很难增加到现有的应用中。

我们提出的开发以 Web Services 应用提供消息级安全支持的 XMLEngine 产品，该产品研发成功后，能够应用于所有以 XML 为基础的应用环境，为这些环境提供可定制、可动态插拔的消息级安全服务。

5.1 定义

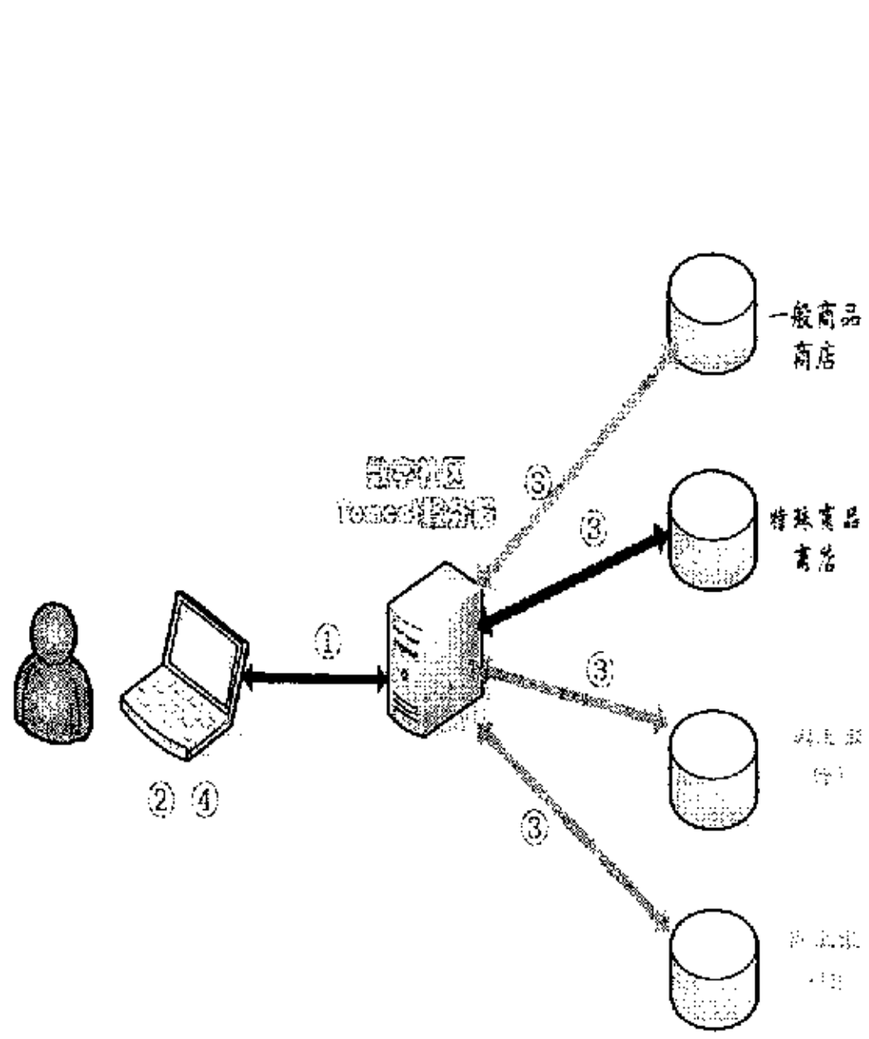
EC: XML Engine Client

ES: XML Engine Server

XML Parser：XML 文档解析器

5.2 典型应用环境

为体现 XML Engine 的典型应用和应用价值，我们设计并实现当前典型应用结构的 Web Services 应用环境“基于 Web Services 的数字社区应用”，并在应用环境中实现购物系统和网上银行系统。目前，Web Services 应用环境有两种不同的拓扑结构，如图 5.1 所示，图 5.1a 是目前常见的结构，所有 Web Services 请求由 WWW 服务器代理发出；图 5.1b 是 Web Services 今后的主要应用情况，Web Services 请求由浏览器直接发出。



a．应用服务器发 SOAP 包　　b．浏览器直接发 SOAP 包

图 5.1 基于 Web Service 的数字社区应用

如图 5.1 所示，在“基于 Web Service 的数字社区应用”中，用户访问的大致流程是：

- 用户访问“数字社区 WWW 服务器”，该服务器的作用类似于门户（Portal），返回用户下一步操作需要的网页，这些网页包含访问不同 Web Services 服务（如一般商品购买、网上银行等）所需的逻辑；
- 用户根据自己的需要填写网页上的内容，比如购买一般商品则填写购买商品的名称、数量，以及用户的银行帐号信息等等，然后通过点击“确认”按钮触发相应的 Web Services 调用；
- 图 5.1a：WWW 服务器（tomcat）根据户输入的相关信息，构建 SOAP 消息，向合适的 Web Services 服务器发送请求，如果是购买一般商品则向“一般商品”服务发送 SOAP 请求；Web Services 服务器经过处理，发回结果数据；WWW 服务器（tomcat）将结果形成网页文档发回用户浏览器；
- 图 5.1b：浏览器根据网页上所包含的逻辑，以及用户输入的相关信息，构建

SOAP 消息，向合适的 Web Services 服务器发送请求，如果是购买一般商品则向“一般商品”服务发送 SOAP 请求；Web Services 服务器经过处理，发回结果数据；

- 用户通过浏览器可以看到 Web Services 调用的操作结果。
- 其中购物系统主要用于过滤模块的演示，包括商品查询、商品列表、购物车、广告等功能；网上银行系统主要用于加密和签名模块的演示，包括用户余额查询、用户明细账查询、用户转账等功能以及管理员余额查询、管理员明细账查询等功能。

为了全面测试 XML Engine，该应用设计注意以下几点：

- 正常的进行 SOAP 请求和应答。以便进行整个系统流程的测试。
- 包含大量数据的 SOAP 包的传送。以便测试系统处理大数据的性能。
- 包含复杂结构数据的 SOAP 包的传送。以便测试系统处理复杂逻辑的性能。
- SOAP 包中的标签和内容都应该包含需要过滤的字符串，以便测试系统过滤功能
- 典型应用环境作为 XML Engine 项目的演示和测试附加功能，完全独立于项目。

5.3 基本设计概念和处理流程

在图 5.1 中，步骤①使用常规的浏览器和 WWW 服务器之间的数据交换，使用 HTML 表达数据，并且其内容是任何人都可以查看的，不涉及任何用户的私有数据，这部分连接不需要 XML Engine 的保护。

而步骤③中则涉及了用户的交易数据，包括用户购买商品的种类、数量，银行账号等信息，这些需要保护的数据在缺乏安全措施的情况下，很容易被窃取。我们的 XML Engine 正好可以在这里大显身手。

图 5.1 中用户浏览器和网上商店等 Web Services 服务可以抽象成 SOAP 的客户端和服务端（图 5.2a），使用了 XML Engine 的部署示意图（图 5.2b）：

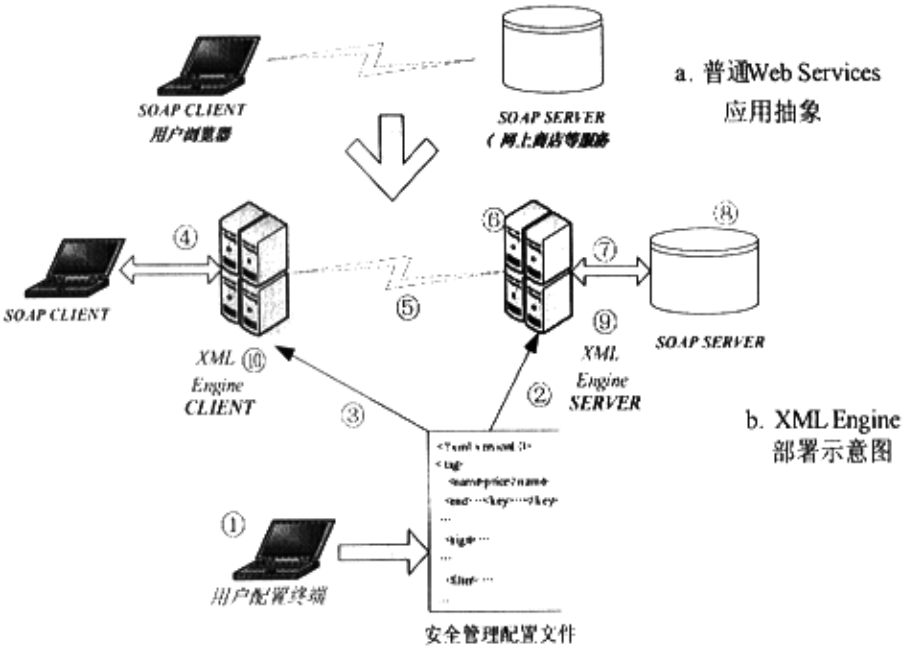


图 5.2 XML Engine 部署

由于位于 SOAP 客户端和服务端的 XML Engine 处理不同，因此我们将 XML Engine 也分为 XML Engine Client（简称 EC）和 XML Engine Server（简称 ES），分别和 SOAP 客户端和服务端部署在一起。

5.4 XML Engine 部署和使用的大体流程

用户根据已有的 XML Engine 组件情况（加密、签名、过滤等）和安全策略，形成“安全管理配置”。

安全管理配置文件被存放在 ES 中，并通过使用配置管理员签名保护配置文件的完整性。

在每一个 session 开始的时候，EC 下载对应 ES 的配置文件，并检查其签名，确定安全管理配置文件的完整性，得到“安全管理配置”。

SOAP CLIENT 发出的 XML 消息被 EC 截获，EC 分析该 XML 消息，并根据“安全管理配置”进行处理（例如对指定标签加密、签名，对指定内容进行过滤），形成新的经过安全处理的 XML 消息。

EC 将经过安全处理的 XML 消息通过公共信道发往 ES，如果这时的数据已

滤），形成新的经过安全处理的 XML 消息。

EC 将经过安全处理的 XML 消息通过公共信道发往 ES，如果这时的数据已经根据用户需求进行加密操作，就可以保证消息的机密性。

ES 收到 EC 发来的 XML 消息，对其中的内容进行逆变换（比如，对加密的部分解密，对签名的部分进行验证），如果这时验证签名的结果正确，则能够确定用户所发的消息是完整的。

ES 将已经经过还原处理的 XML 消息（在有的情况下，这时的消息并不完全等同于 SOAP Client 所发出的原始消息，比如经过过滤的消息）发往 SOAP Server。

SOAP Server 按照其原有方式处理 SOAP 中所包含的应用请求，形成“结果消息”，发往 ES。

ES 这时执行步骤 4 中 EC 所进行的操作（比如加密、签名、过滤等），对结果消息进行安全处理。并通过公共信道发往 EC。

EC 收到 ES 发来的结果消息后，执行步骤 6 种 ES 的操作（比如解密、验证等），如果 EC 验证通过，则可保证消息确实是希望的 ES 发出的。最终还原结果返回给用户端的 SOAP Client，这样，SOAP Client 所看到的结果消息和 XML Engine 未部署时相同（在有的安全组件存在时，可能会有不同，比如过滤组件存在时，可能对结果消息进行了过滤）。

在 XML Engine 的设计中，注重了对原有应用的保护：在部署时几乎完全不用修改应用，用户只需要安装 XML Engine，配置安全管理组件，就可以对应用进行很好的安全保护。有一个细节必须指出来，就是原有应用还是需要做细微的改变，即将 SOAP Client 发出的 SOAP 请求通过代理的形式发往 EC。在浏览器作为 SOAP Client 的情况下，配置浏览器代理是非常容易的。

需要说明的是，EC 和 ES 并不是一一对应的关系，每一个 EC 都可能和多个 ES 连接，同样，每一个 ES 都可能接收多个 EC 的连接，这种情形，和浏览器与 Web 服务器的关系是相似的（如图 5.3 所示）。

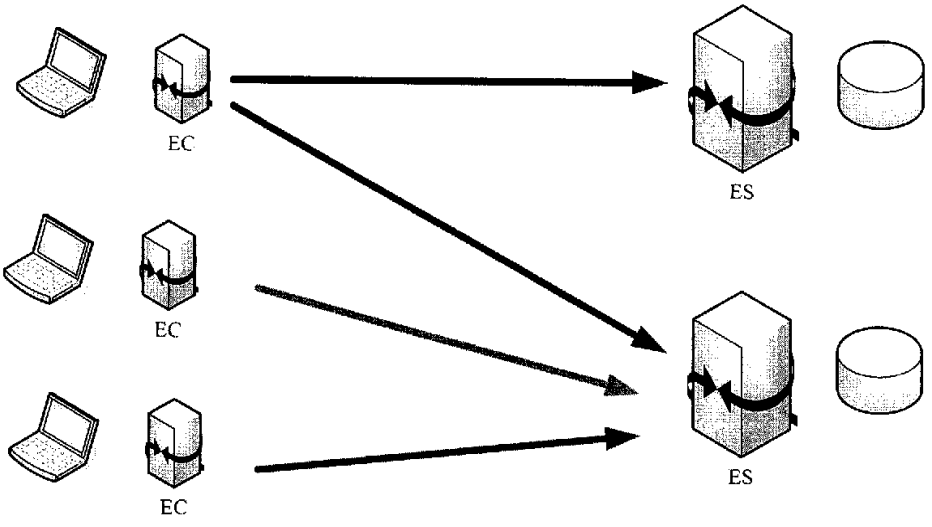


图 5.3 多个 EC 和 ES 的连接示意图

5.5 XMLEngine 的结构

5.5.1 XML Engine 总体结构

XML Engine 总结构如图 5.4 所示：

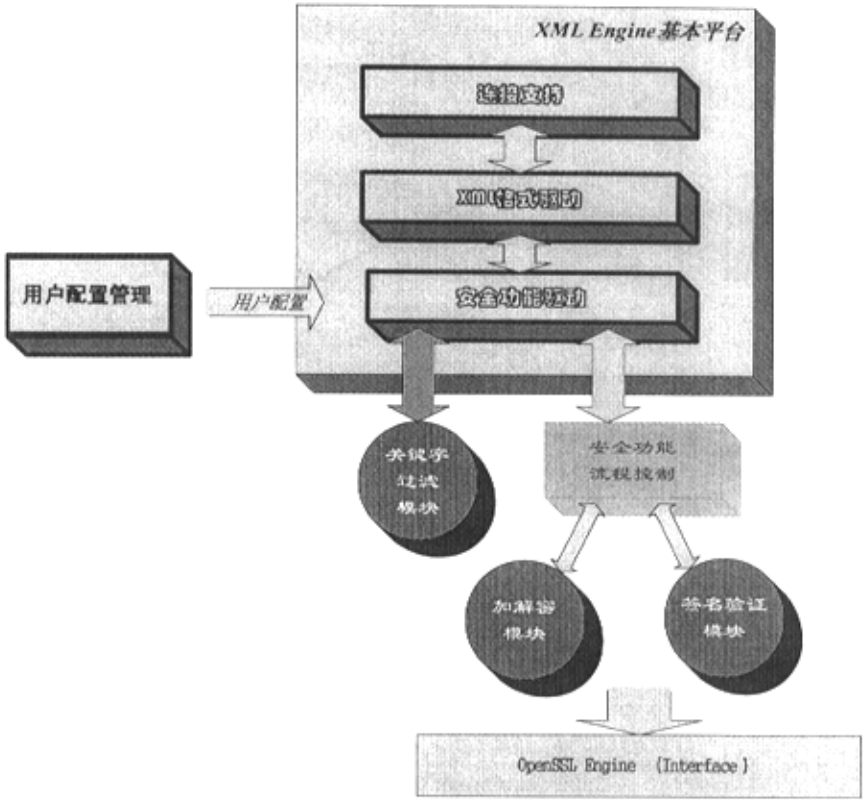


图 5.4 XML Engine 总结构图

安全管理配置

用户配置部分起着将用户安全策略转换为 XML Engine 配置的重要作用，其主要任务是：

- 对配置管理员的身份认证；
- 友好的用户配置界面；
- 对新安装的系统，配置并生成新的配置文件；
- 对已经配置的系统，装入并修改配置，形成新的配置文件；
- 对配置文件完整性的保护；
- 将配置文件明文上传到 ES。

需要说明的是安全配置管理是比较独立的一个模块，它和其它模块的联系就

是安全策略配置文件：Config.xml，图 5.5 是安全管理配置模块的部署示意图。

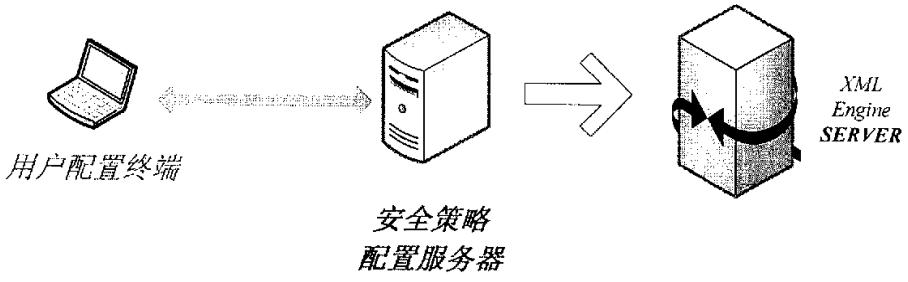


图 5.5 安全配置管理部署示意图

图 5.6 表示了“安全管理配置”的流程示意图，其中有用户标记的表示需要用户输入。配置界面的内容根据用户安装的安全组件确定。

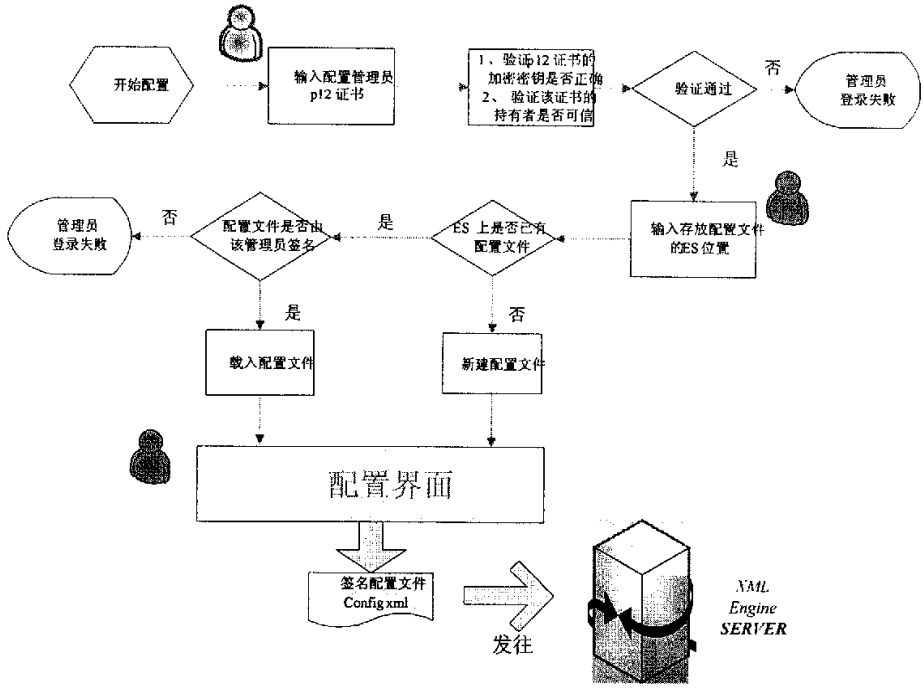


图 5.6 安全管理配置流程图

当经过签名的配置文件送达 ES 之后，如果 ES 正在运行，需要通知 ES 重新读取新的配置文件（一般来说，需要 ES 重新初始化一次）。

5.5.2 XML Engine 运行结构

当 XML Engine 接收到数据后，内部处理过程如图 5.6 所示：

理，否则直接将数据送到出连接；

- 基础平台使用“XML 格式驱动”解析 XML 文档（即 SOAP 报文），并将结果保留在内存中，以便今后安全模块的处理；
- “安全功能驱动”检查已经装入的安全模块，逐一调用这些模块，并将已经解析的 XML 文档传给这些模块；
- 在需要“流程控制”的模块，由流程控制负责模块中安全功能的使用；
- 不需要流程控制时，由安全功能驱动直接调用相应的安全模块；

所有的安全模块执行完毕后，基础平台再次使用“XML 格式驱动”，这一次将处理过后的 XML 文档重组起来，并通过出连接发送出去。

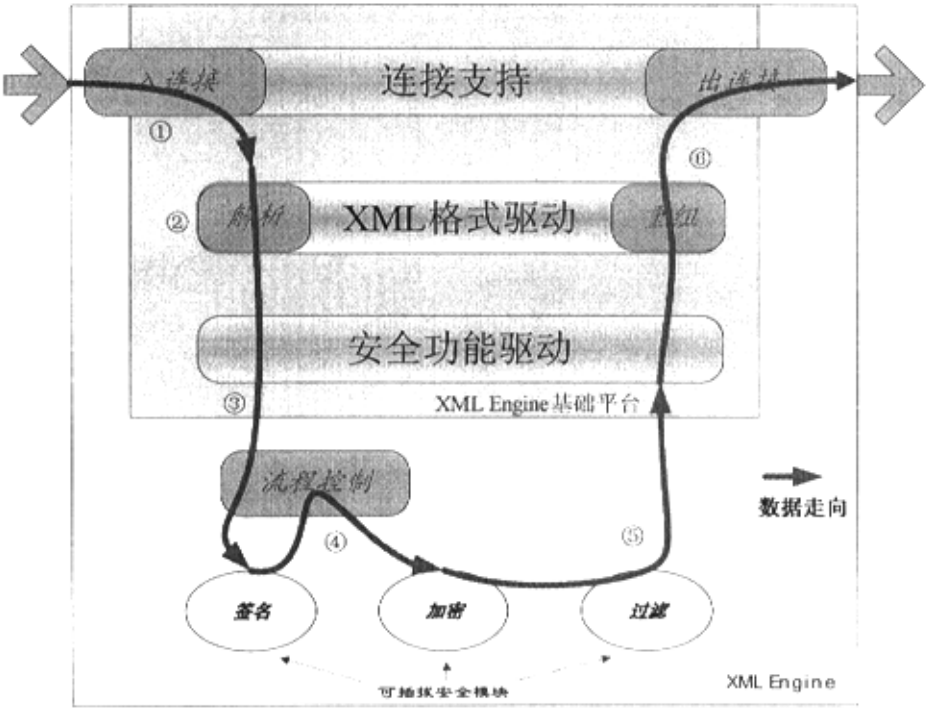


图 5.7 XML Engine 内部处理过程

5.5.3 基本平台

整个项目通过引擎平台组织在一起，平台将各个组成部分结合起来，协调其运行，并提供良好的架构。具体来说，平台的任务有：

5.5.3 基本平台

整个项目通过引擎平台组织在一起，平台将各个组成部分结合起来，协调其运行，并提供良好的架构。具体来说，平台的任务有：

- 构建良好的可升级的架构，构成一个完整的安全系统，并提供模块化的快速升级能力；
- 提供可插拔的扩展模块机制，使安全模块可以方便的加入和卸出系统；
- 处理多进程、线程运行的管理，为系统各部分分配任务；
- 提供 XML Engine 公共功能支持：连接支持；XML 文档的处理，包括解析和重组；配置文件的获取和完整性验证。
- 完成系统平台其它底层支撑功能，如统一的日志管理、出错管理等等。

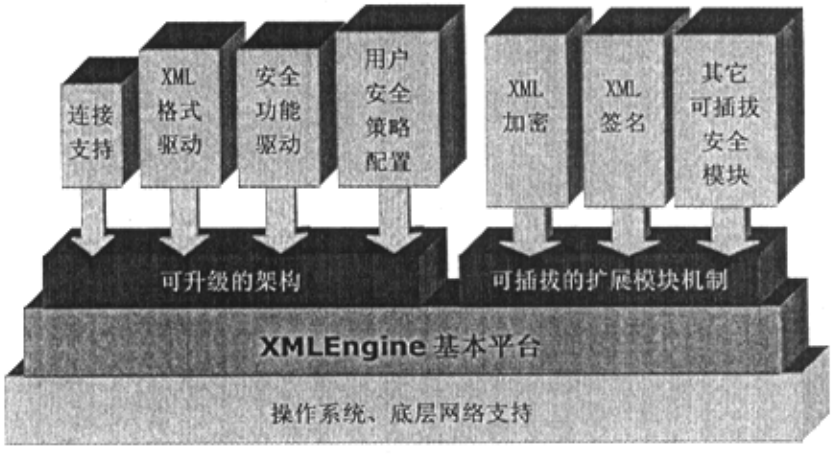


图 5.8 基本平台与 XMLEngine 其它部分的关系

1、连接支持

正是由于处于 Web Services 客户端和服务端 XML Engine 连接支持部分的不同，导致了 XML Engine 分为 EC（客户端）和 ES（服务端）。除此之外，EC 和 ES 几乎没有区别。

连接支持部分负责 XMLEngine 最外层的接口，换句话说，产品实现之后，应用从外部看到的就是这部分。具体地，连接实现分为客户端和服务器端两部分。

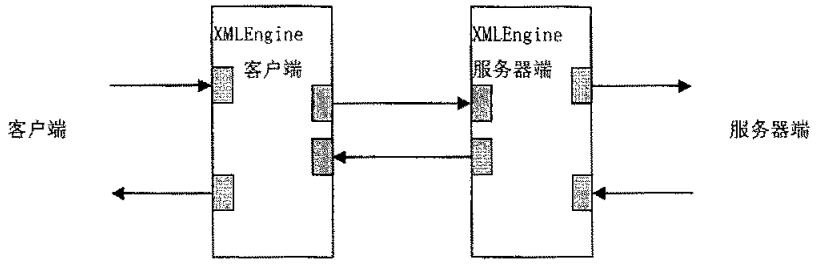


图 5.9 基本平台连接支持示意图

按连接方向划分，连接有两种：监听（负责监听远端的连接请求）和发送请求（负责向远端发送连接请求）。

按与外部应用的关系划分，连接也是两种：应用连接（连接应用的客户端和服务器端）和安全连接（连接 XML Engine，在这种连接下，信息交流是安全的）。

由于客户端和服务器端各自不同的处理流程，连接实际上有八种不同的形式，这八种形式的连接实现有各自的特性，共同完成 XML Engine 之间和与应用之间的连接交互。

2、基本平台对配置文件的处理

在 ES 中，基本平台对配置文件的处理比较简单：在平台初始化的时候，从指定位置读出配置文件，验证其有效性，经过处理放入内存，以便安全功能驱动模块能高效地进行处理。

在 EC 中，基本平台对配置文件的处理要复杂一些：当新的 Web Services session 到来的时候，EC 需要从这个 session 中得到需要连接的 ES 的位置，并从该位置下载配置文件。然后验证其有效性，经过处理放入内存。

设计中存在 ES 和 EC 中配置文件不一致的情况：当 ES 重新配置后，EC 只有等到下一个 session 才能更新其配置文件。但是如果每一次连接都要求 EC 到 ES 重新下载配置文件，这会产生严重的效率问题，考虑到配置文件使用前都需要进行验证，每一次连接都要求重新下载配置文件是不能接受的。由于每个 session 持续时间都在分钟级别，这种不一致应该是可以接受的。

3、安全功能驱动

安全功能驱动提供对安全模块的插拔支持，并在安全模块插入后，自动探知它的存在，为其调入配置文件，并在对 XML 文档的处理流程中使用该安全模块进行处理。

4、XML 格式驱动

XML 格式驱动承担着对 XML 文档处理的第一道处理工序，它将 XML 文档中的数据提取出来，组织成树状结构，并且提供 DOM 接口，以便后续模块操作。

XML 格式驱动的主要任务是：

- 解析 XML 文档在内存生成 DOM 树
- 对 XML 树上的每个节点进行操作
- 生成 XML 文档（XML 文档生成器）

对于每种任务，对应的大致处理流程如图 5.10 所示：

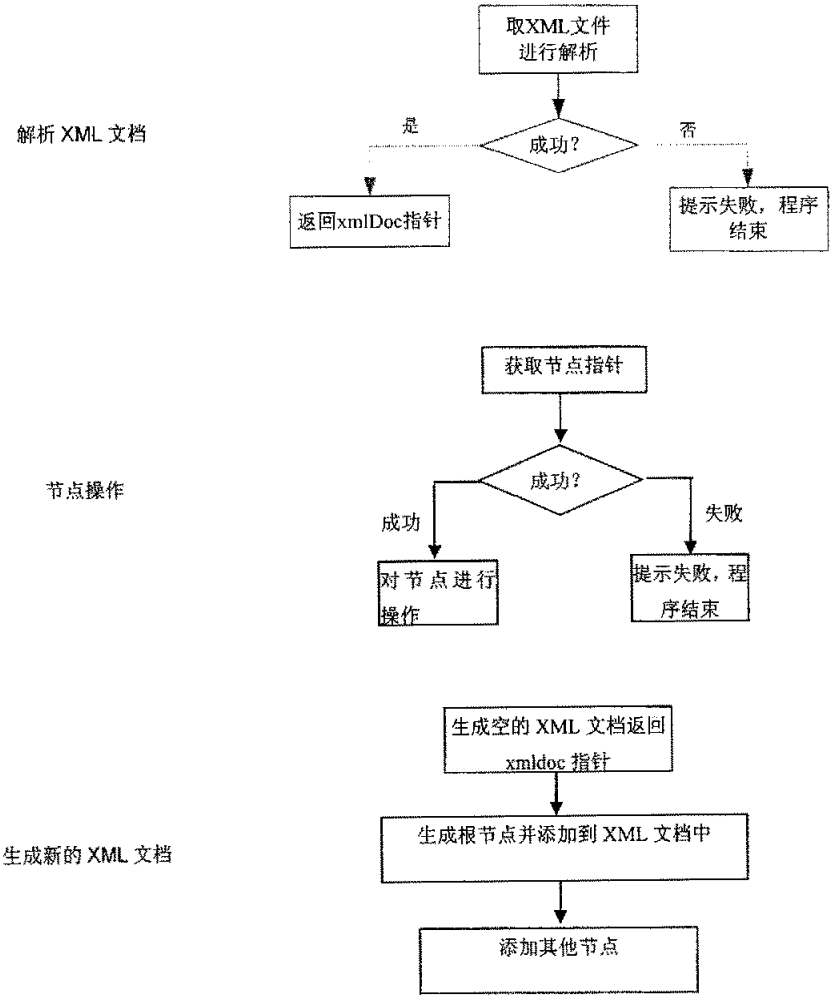


图 5.10 XML 格式驱动基本操作

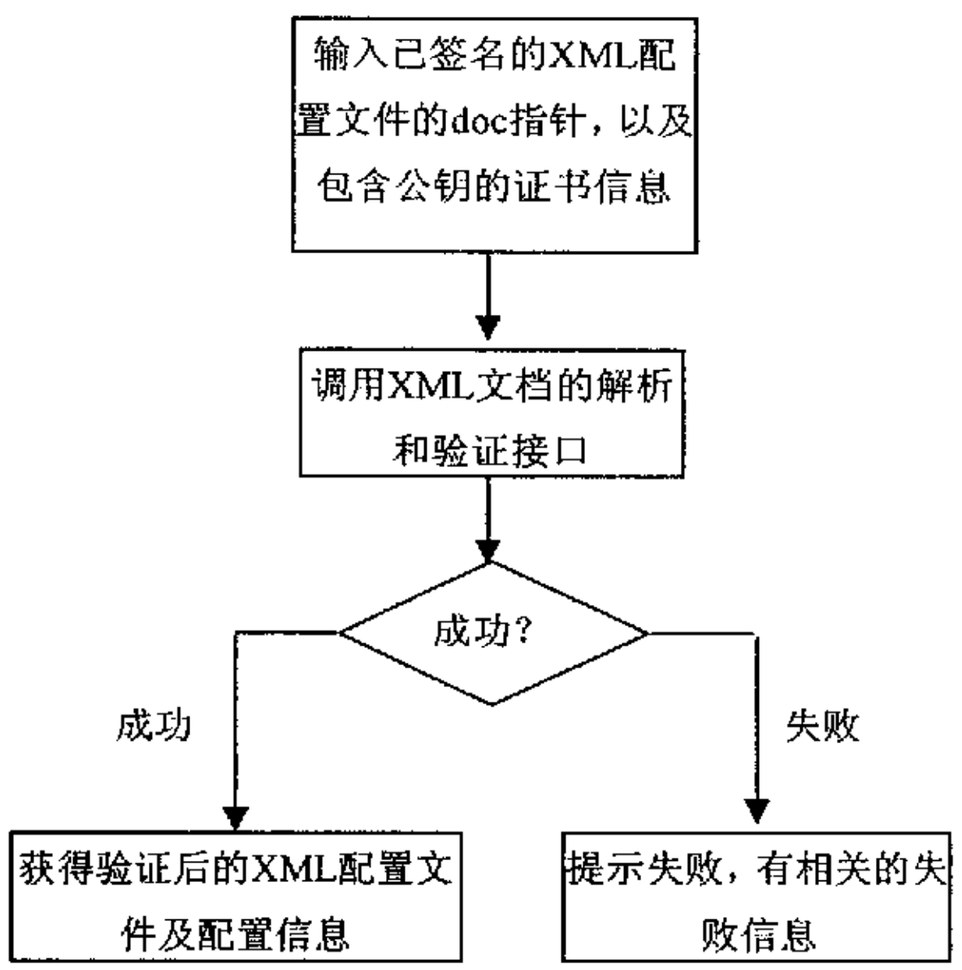
5.5.4 流程控制

XML 操作顺序流程控制模块的主要任务是：

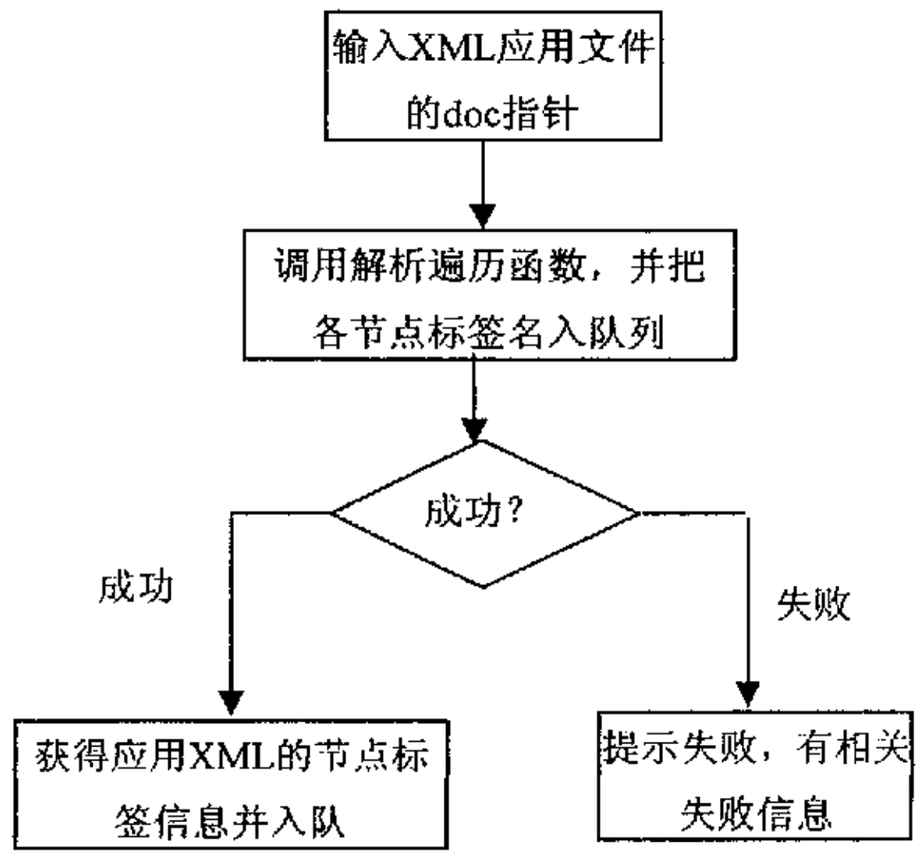
- 将配置 XML 文档中的配置信息 config.xml 提取出来
- 与实际应用的 XML 文档进行比较，以得到实际应用的 XML 文档中需要进行加密、签名操作的标签以及内容
- 根据已经配置好的对实际应用的 XML 文档进行加密、解密和签名、验证操作的处理工序；
- 调用相应的加解密、签名验证接口函数，完成指定的操作。

整个模块的设计遵循“先子后父、先签后密、签名内嵌、加密替换”的算法原则。其主要任务可由以下的流程处理示意图简要说明：

验证并读取 XML 配置文件：



解析遍历 XML 应用文件：



根据配置信息操作 XML 应用文件：

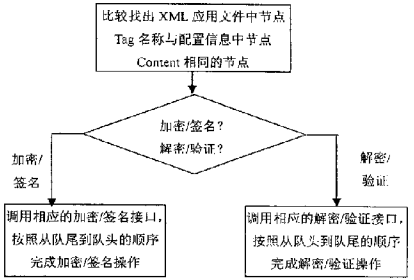


图 5.11 XML 操作顺序流程控制

5.5.5 XML 加/解密和签名/验证

XML 加/解密和签名/验证是不同的两个模块，用户可以根据需要选择其中一个或者两个进行使用。在设计角度看，这两个模块有很强的联系，事实上，两个模块在最初设计时属于同一个模块。其处理过程有许多相似之处。

XML 加/解密和签名/验证总体的设计结构框架如图 5.12 所示：

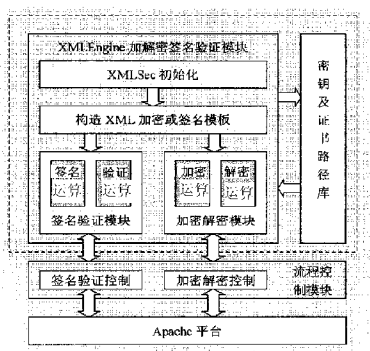


图 5.12 XML 加/解密和签名/验证模块示意图

XML 加/解密和签名/验证模块处理过程是：

- XMLSec 初始化：对加密签名运算所需要的算法、内存分配、数据结构及全局变量等进行初始化。
- 构造 XML 加密或签名模板基于 XML 的元素加密和签名，按照 W3C 中相关规范，是在原有的 XML 文档内容中嵌入相应的加密或签名元素节点，例如：对于签名，是以<Signature>为元素名称及特定的名字空间为父元素的一系列元素节点集，其内容包括指明所使用的密钥类型、所采用的规范性算法、摘要算法，当然还包括签名的密文以及用户证书（可选）等等。在进行具体运算前首先构造这样一个除摘要值和签名密文的节点作为模板，提供给下一步进行操作。
- 加解密和签名验证运算：这一步将按照上一步所产生和模板中的指定算法进行具体运算，并将运算结果写入到模板中相应的位置。

1、XML 加/解密模块

加解密包括普通用对称密钥直接对元素进行加解密、用 Session 密钥的方式加解密和用数字信封的方式加解密三种：

①对称加解密

- 加密：根据用户在配置文件中的需求，按指定的加密密钥和加密算法对 XML 文档中的元素进行加密，采用明文替换的方式
- 解密：使用用户在加密时所使用的密钥对已经加密的元素进行解密

②用 Session 密钥的方式加解密

- 加密：根据用户在配置文件中的需求，按指定方式产生一对称 Session 密钥，用该密钥对对 XML 文档中的元素进行加密，然后再用用户指定的对称密钥对该 Session 密钥进行加密后，封装在已经加密的 XML 文档中，采用明文替换的方式
- 解密：使用用户在加密时对 Session 密钥进行加密的密钥解密封装在 XML 文档中的 Session 密钥，然后再用该 Session 密钥对 XML 文档进行解密

③用数据信封的方式加解密

- 加密：根据用户在配置文件中的需求，按指定方式产生一对称 Session 密钥，用该密钥对对 XML 文档中的元素进行加密，然后再用用户 PKI 证书中的公钥或者公钥文件对该 Session 密钥进行加密后，封装在已经加密的 XML 文档

中，采用明文替换的方式

- 解密：使用加密 Session 密钥相对应的用户私钥解密封装在 XML 文档中的 Session 密钥，然后再用该 Session 密钥对 XML 文档进行解密

2、XML 签名/验证模块

- 签名：根据用户在配置文件中的需求，使用用户私钥采用给定的规范化算法以及签名和摘要算法对指定的元素进行签名，签名内容内嵌到待签名元素内部，可以对一个元素进行多次签名，同时也要求用户提供私钥相对应的 PKI 证书，并任务签名中的一个元素一起发送出去
- 验证：需要提供一证书路径来验证收到的 XML 文档中的证书是否合法有效，然后再验证签名是否正确，如果是则删除该签名元素节点，恢复 XML 文档的原始信息

3、内容过滤

本模块实现基本的内容过滤功能，提供基于标签名和文本关键字过滤的能力。

- 基于标签名的过滤：主要用于对应用功能方面的控制。比如，在网上购物应用中，如果我们不希望用户访问诸如“成人用品”的内容，可以通过设置过滤 adultProduct 标签，来控制是否允许成人用品内容的通过。
- 基于文本关键字过滤：主要用于对内容方面的控制。比如，我们如果不希望出现“法轮功”的字眼，可以通过设置，过滤该关键字。

当过滤功能找到目标，处理的方式有多种选择，目前提供两种处理方式：

- 替换：对目标数据进行替换，说明已经被过滤。被替换的内容可能是整个标签的内容，也可能是整个文档。
- 删除：删除目标数据所在的标签或者整个文档。

第六章. 构建 Web Service 典型应用——数字书店

6.1 典型应用介绍

6.1.1 典型应用要达到的目标

该应用是一个集成部分网上银行功能的网上书店的例子,开发这个应用目的是在本系统商业化过程中的测试和演示系统,因此本应用是XMLEngine项目能否商业化的关键,该应用必须达到以下要求:

- 网上比较流行的基于 Web Service 的应用系统模式
- 正常的进行 SOAP 请求和应答以便进行整个系统流程的测试;
- 包含大量数据的 SOAP 包的传送，以便测试系统处理大数据的性能
- 包含复杂结构数据的 SOAP 包的传送，以便测试系统处理复杂逻辑的性能;
- SOAP 包中的标签和内容都应该包含需要过滤的字符串（在含有过滤模块的情况下，由于本论文主要是针对 XML 的安全方面的，所以凡是与过滤模块的相关的本论文都不予讨论），以便测试系统过滤功能。
- 系统技术应该先进，架构应该合理，可扩展性要强，如果需要作为商业应用，能够在该系统的基础下快速扩展。

基于以上要求，典型应用采用 J2EE 技术开发，Web 容器用 Tomcat，表示层框架用 struts，Web Service 工具包用 Apache Axis。

6.1.2 系统功能

本数字社区系统包含两部分：购物系统和网上银行系统。

购物系统主要用于过滤模块的测试，包括商品查询、商品列表、购物车、广告等功能；网上银行系统主要用于加密和签名模块的测试，包括用户余额查询、用户明细帐查询、用户转帐等功能以及管理员余额查询、管理员明细帐查询等功能。

6.1.3 技术实现

6.1.3.1 物理部署图

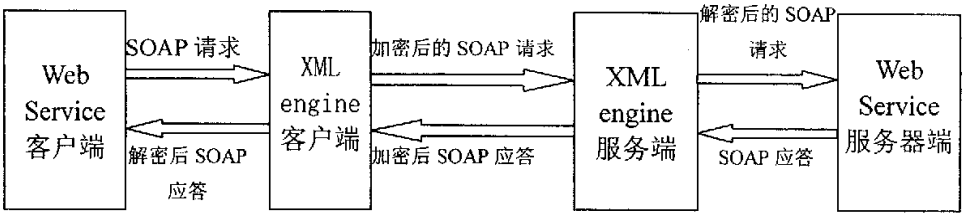


图 6.1 SOAP 物理部署图

本测试系统构建在 J2EE 基础上，整个系统的体系结构分为四层，各层主要功能为：客户端主要采用浏览器（如 IE）；应用服务器层（TomCat、Servlet 等）主要完成整个系统的业务逻辑处理、接收浏览器的请求、向 Web Service 服务器请求服务、接收 Web Service 服务器返回的数据等，最后向浏览器返回处理后的数据。Web Service 服务层（TomCat）提供计算、数据存取等各种服务。数据库层（数据库用 MySQL）实现数据的存储。

6.1.3.2 详细设计

- 网上银行流程图：图 6.2 是个简单的网上银行模拟流程：

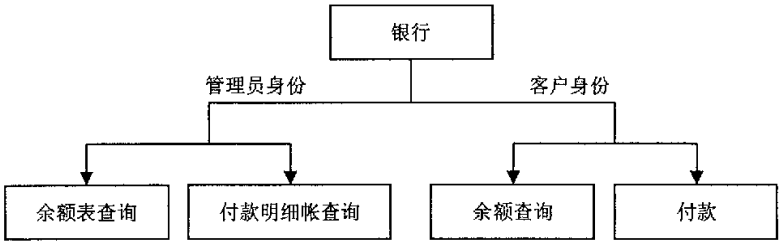


图 6.2 网上银行数据处理流程图

- 顾客网上商店购物顺序图如图 6.3 所示：

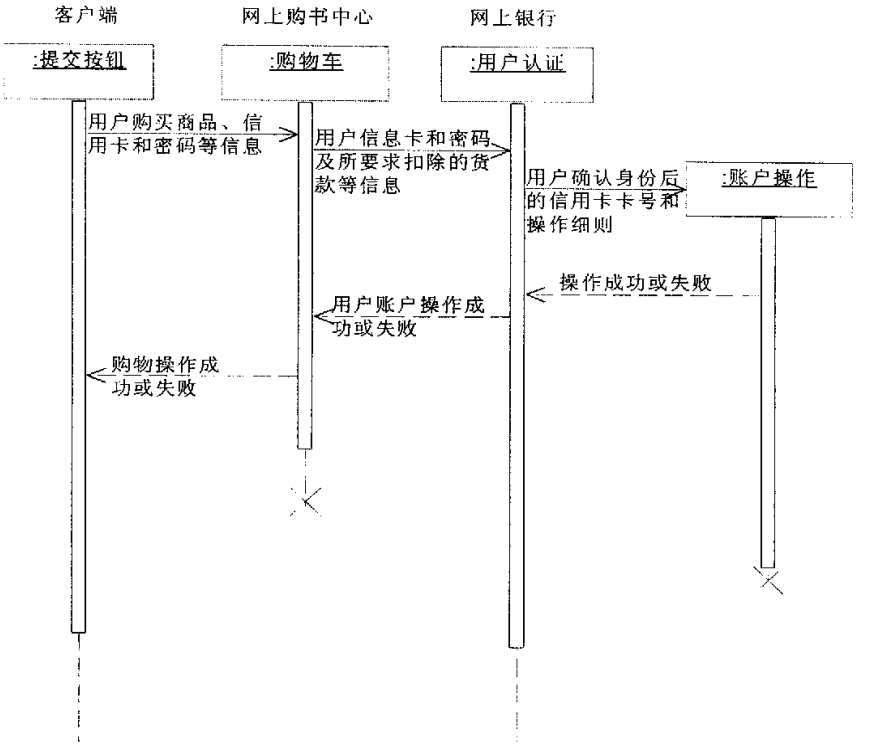


图 6.3 顾客网上商店购物顺序图

- 功能介绍

①会员注册

为了便于系统对网上商店购买者的管理，本系统采用免费的注册会员制度。如果首次来访，建议您注册为会员，点击页面导航条上的“会员注册”，根据提示填写完整的注册表单后，您就成为此网上商店的一名会员了。另外，也可在选购好商品后去收银台时，在会员区再注册。

会员注册申请表单包括三部分：

会员资料录入部分。可根据提示输入会员代号、会员密码，以及查询密码问题、查询密码答案等，作为取回密码的条件。其中会员代号及会员密码和会员登录时的登录名及密码是一致的。

个人资料部分。为了确保用户得到更好的服务，此部分务必详细真实地填写如姓名、电话、电子邮件地址、邮编、地址等项，作为保证能联系到你的主要依据。

特殊服务信息区。在此要填写用户身份证件号码及详细地址，便于核实发款人及收货人的真实可信性，并且明确送货的地址。

注册成功系统会提示您注册完成的时间，并提示您通过返回首页来选购商品。

②订单查询

订单查询必须通过会员登录，所以必须是会员才可能查询订单。订单查询功能提供了两种查询方法：1）按日期及订单状态查询。即输入一个订单区间或选择一种订单状态，即可查出。2）按订单号查询。根据购物系统给出的订单号查询。这两种方法只能使用一种，不可同时使用。

查询结果会简要列出订单信息，如订单号、金额、付款方式、送货方式、订单状态、下单时间等。如果想查看订单详细信息，可以通过点击订单号，系统会给出此订单中所购商品的详细信息，如：商品名称、商店名称、单价、数量、价格小计等。至此，购买者的网上购物过程即告结束。

③会员登录

在本系统中，所有注册会员购物订单，系统中都有记录且购物金额有一定的继承性，为了对会员信息保密及系统安全考虑，系统设置必须通过用户登录才能查询订单或查看所属购物优惠级别等等。

注册会员后，可通过会员登录查看所属组别、所有员组及相应的购物优惠比例。

会员登录可以通过两种方法：

在首页中的“会员登录”图框中直接输入会员号及密码即可；

通过系统导航栏中的会员登录项，转入“会员登录”页面来登录。

④会员资料修改

会员的注册资料难免会有所变化需要修改，会员可以修改除了"会员代号"以外的所有资料。

要修改会员资料，您只需点击页底的“修改注册信息”设置，通过会员登录后，就可以在“会员信息修改”页面进行修改。所有修改经保存后才有效。

⑤进入购物区

系统首页为总的购物区，你可以在此浏览、挑选或有目的的查询某类商品，本系统设置以下四种方式浏览商品。

⑥选购商品

通过不同的方式浏览、查询各种商品，将所要购买的商品投入购物车。在购物车设置中会列出所购商品的各项信息，如商品编号、商品名称、商品单价、选购数量、会员价格小计等等。在购物车中可以修改购买数量或取消商品的购买。如果还要选购可通过“返回继续购物”按钮来实现，或通过“网上银行付款”按钮付款结帐来结束选购商品。

⑦付款结帐

直接通过网上银行转帐给商店。

⑧其它设置

当然，购物过程中或购物后还会出现类似客户不知道该如何购物怎么办？没有购买到所需商品、查看选购的物品或要求退换购买的物品、对此网上商店有疑议、或要查看购物订单等一系列问题。本系统也同时提供了这些功能，如查询密码、购物指南、缺货登记、购物车、售后服务、客户投诉、新闻动态以及帮助信息等等。

- 应用的主界面如图 6.4:

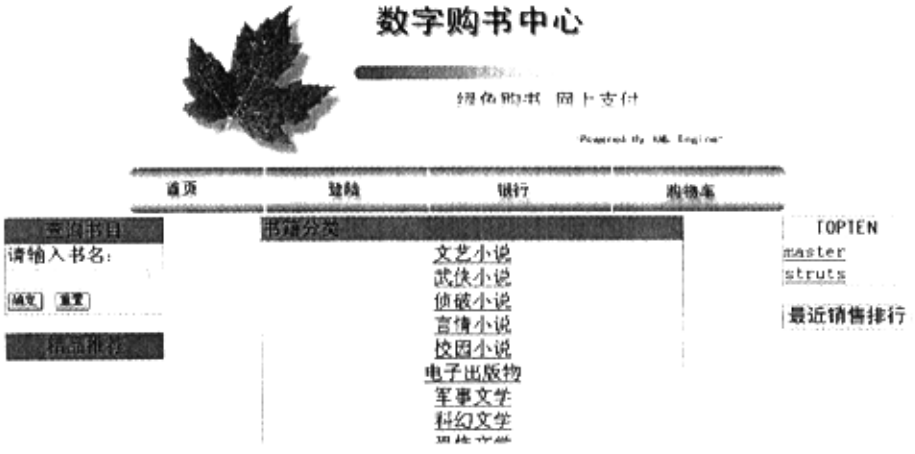


图 6.4 主界面

6.2 客户端设计

6.2.1 MVC 设计模式

MVC 最初是在 Smalltalk-80 中被用来构建用户界面的。M 代表模型 Model, V 代表视图 View， C 代表控制器 Controller。MVC 的目的是增加代码的重用率，减少数据表达，数据描述和应用操作的耦合度。 同时也使得软件可维护性，可修复性，可扩展性，灵活性以及封装性大大提高。

单用户的应用通常是以事件驱动的用户界面为组织结构的。开发人员用一个界面工具画了一个用户接口界面，然后编写代码根据用户输入去执行相应的动作，许多交互式的开发环境鼓励这么做，因为它强调先有界面然后再有功能。一些软件设计模式策略是这样的，然后经常将固定后的代码融入最后的系统当中。导致的结果就是，程序组织围绕用户界面元素和用户在那些界面元素上的动作，数据的存储，应用的功能以及用来显示的代码都杂乱无章的缠绕在一起。在单用

户的系统里代码结构是可以这样的，因为系统需求不会频繁变化。但是对一个大的系统如大型 Web 系统，或电子商务系统来说就不太适用了。

通过把数据模式从各种可以被存取和控制的数据中分离出来可以改善分布式系统的设计。MVC 设计模式由三部分组成。模型是应用对象，没有用户界面。视图表示它在屏幕上的显示，代表流向用户的数据。控制器定义用户界面对用户输入的响应方式，负责把用户的动作转成针对 Model 的操作。Model 通过更新 View 的数据来反映数据的变化。 三者关系如图 6.5：

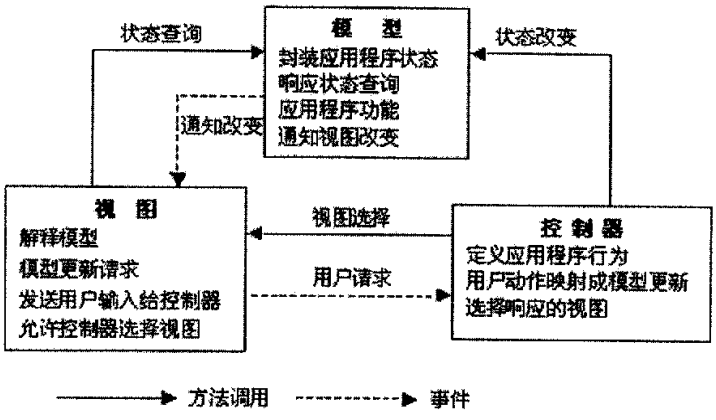


图 6.5：MVC 组件的类型和功能

对 MVC 关系图的理解

	模型 M	视图 V	控制器 C
分工	抽象系统应用的功能 封装系统的状态 提供使用系统功能的方法和路径 管理数据的存储和一致性 当数据发生变化时通知相关部分	抽象数据表达 表示针对用户的数据 维护与 Model 数据的一致性	抽象用户和系统的事件的语义映射 把用户输入翻译为系统事件 根据用户的输入和上下文情况选择合适的显示数据
协作	当他改变系统数据时通知 View 能够被 View 检索数据 提供对 Controller 的操作途径	把 Model 表征给用户 当数据被相关 Model 改变时更新表示的数据 把用户输入提交给 Cotroller	把用户输入转变成对 Model 的系统行为 根据用户输入和 Mode 的动作结果选择合适的 View

图 6.6 MVC 的分工与协作

模型部件是软件所处理问题逻辑在独立于外在显示内容和形式情况下的内

在抽象，封装了问题的核心数据、逻辑和功能的计算关系，他独立于具体的界面表达和 I/O 操作。

模型包含了应用问题的核心数据、逻辑关系和计算功能，它封装了所需的数据，提供了完成问题处理的操作过程。控制器依据 I/O 的需要调用这些操作过程。模型还为视图获取显示数据而提供了访问其数据的操作。

这种变化-传播机制体现在各个相互依赖部件之间的注册关系上。模型数据和状态的变化会激发这种变化-传播机制，它是模型、视图和控制器之间联系的纽带。

视图部件把表示模型数据及逻辑关系和状态的信息及特定形式展示给用户。它从模型获得显示信息，对于相同的信息可以有多个不同的显示形式或视图。视图通过显示的形式，把信息转达给用户。不同视图通过不同的显示，来表达模型的数据和状态信息。每个视图有一个更新操作，它可被变化-传播机制所激活。当调用更新操作时，视图获得来自模型的数据值，并用它们来更新显示。

在初始化时，通过与变化-传播机制的注册关系建立起所有视图与模型间的关联。视图与控制器之间保持着一对一的关系，每个视图创建一个相应的控制器。视图提供给控制器处理显示的操作。因此，控制器可以获得主动激发界面更新的能力。

控制部件是处理用户与软件的交互操作的，其职责是控制提供模型中任何变化的传播，确保用户界面于模型间的对应联系；它接受用户的输入，将输入反馈给模型，进而实现对模型的计算控制，是使模型和视图协调工作的部件。通常一个视图具有一个控制器。

控制器通过时间触发的方式，接受用户的输入。控制器如何获得事件依赖于界面的运行平台。控制器通过事件处理过程对输入事件进行处理，并为每个输入事件提供了相应的操作服务，把事件转化成对模型或相关视图的激发操作。

如果控制器的行为依赖于模型的状态，则控制器应该在变化-传播机制中进行注册，并提供一个更新操作。这样，可以由模型的变化来改变控制器的行为，如禁止某些操作。

模型、视图与控制器的分离，使得一个模型可以具有多个显示视图。如果用户通过某个视图的控制器改变了模型的数据，所有其它依赖于这些数据的视图都应反映到这些变化。因此，无论何时发生了何种数据变化，控制器都会将变化通知所有的视图，导致显示的更新。这实际上是一种模型的变化-传播机制。

6.2.2 MVC1

为了解决 Web 应用中的 MVC 问题，sun 公司提出了 MVC 1 模式

Model 1 结构如图 6.7 所示：

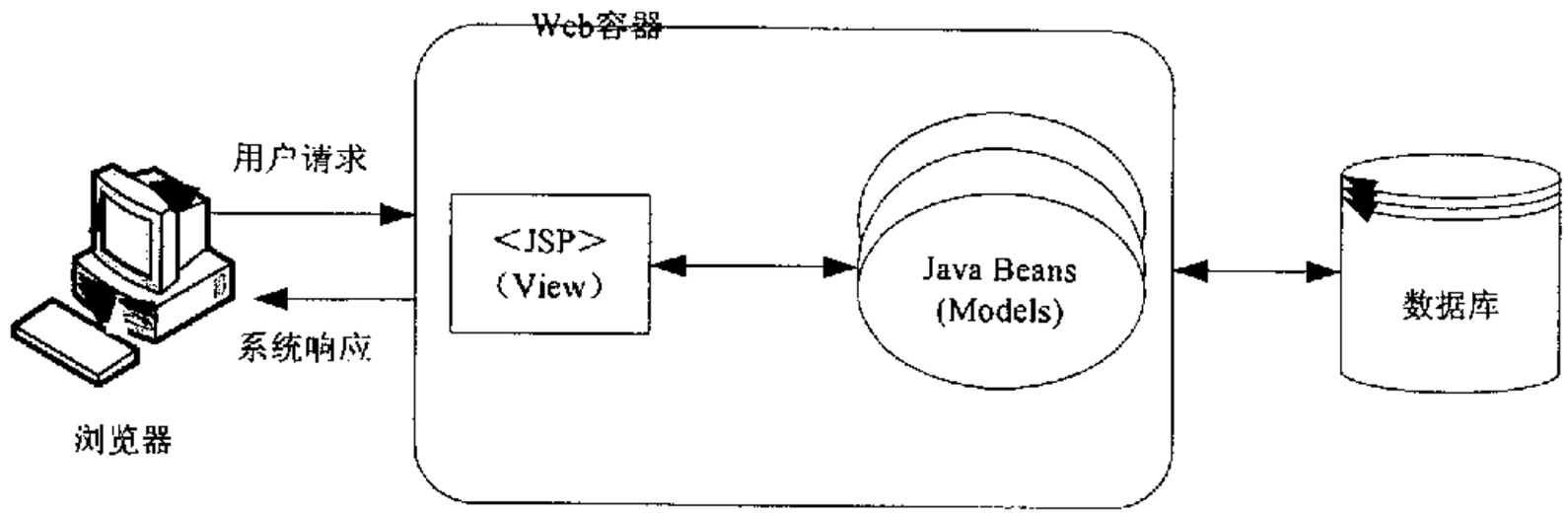


图 6.7 Model 1 结构图

model 1 是一个以 JSP 文件为中心的模式，在这种模式中 JSP 页面不仅负责表现逻辑，也负责控制逻辑。专业书籍上称之为逻辑耦合在页面中，这种处理方式，对一些规模很小的项目如：一个简单的留言簿，也没什么太大的坏处，实际上，人们开始接触一些对自己来说是新的东西的时候，比如，用 JSP 访问数据库时，往往喜欢别人能提供一个包含这一切的单个 JSP 页面，因为这样在一个页面上他就可以把握全局，便于理解。但是，用 Model 1 模式开发大型时，程序流向由一些互相能够感知的页面决定，当页面很多时要清楚地把握其流向将是很复杂的事情，当您修改一页时可能会影响相关的很多页面，大有牵一发而动全身的感觉，使得程序的修改与维护变得异常困难；还有一个问题就是程序逻辑开发与页面设计纠缠在一起，既不便于分工合作也不利于代码的重用，这样的程序其健壮性和可伸缩性都不好。

6.2.3 MVC2

为了克服 Model 1 的缺陷，人们引入了 Model 2，如图 6.8 所示：

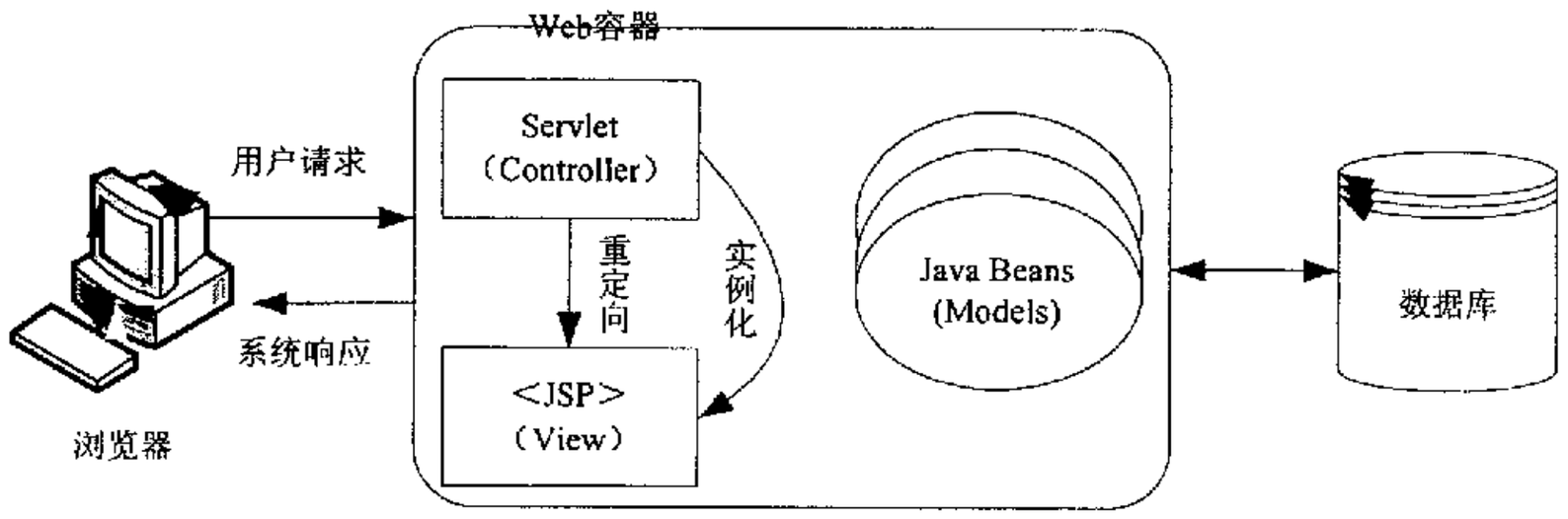


图 6.8 Model2 结构图

不再直接送给一个处理业务逻辑的 JSP 页面，而是送给这个控制器，再由控制器根据具体的请求调用不同的事务逻辑，并将处理结果返回到合适的页面。因此，这个 servlet 控制器为应用程序提供了一个进行前-后端处理的中枢。一方面为输入数据的验证、身份认证、日志及实现国际化编程提供了一个合适的切入点；另一方面也提供了将业务逻辑从JSP文件剥离的可能。业务逻辑从JSP页面分离后，JSP 文件蜕变成一个单纯完成显示任务的东西，这就是常说的 View。而独立出来的事务逻辑变成人们常说的 Model,再加上控制器 Control 本身，就构成了 MVC 模式。实践证明，MVC 模式为大型程序的开发及维护提供了巨大的便利。

6.2.4 struts

6.2.4.1struts 概述

Struts 就是一种具体实现 MVC2 的程序框架。它的大致结构如图 6.9 所示

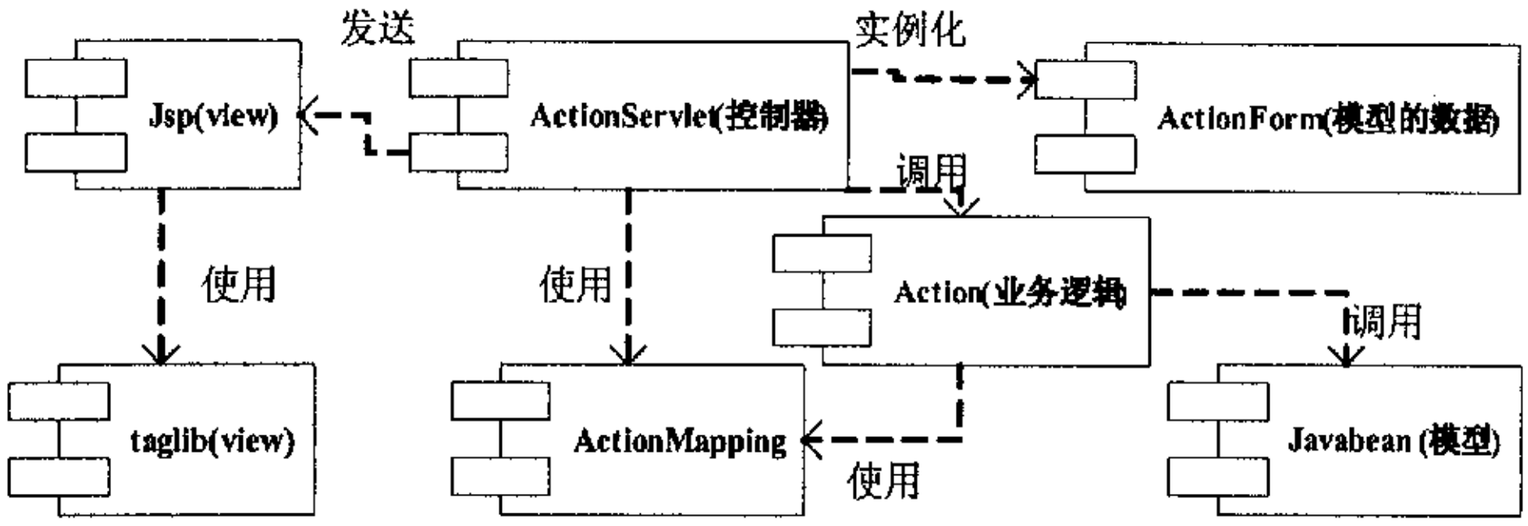


图 6.9 Struts 结构图

图 6.9 基本勾画出了一个基于 Struts 的应用程序的结构，从左到右，分别是其表示层（view）、控制层(controller)、和模型层(Model)。其表示层使用 Struts 标签库构建。来自客户的所有需要通过框架的请求统一由叫 ActionServlet 的 servlet 接收（ActionServlet Struts 已经为我们写好了，只要您应用没有什么特别的要求，它基本上都能满足您的要求），根据接收的请求参数和 Struts 配置(struts-config.xml)中 ActionMapping，将请求送给合适的 Action 去处理，解决由谁做的问题，它们共同构成 Struts 的控制器。Action 则是 Struts 应用中真正干活的组件，开发人员一般都要在这里耗费大量的时间，它解决的是做什么的问题，它通过调用需要的业务组件（模型）来完成应用的业务，业务组件解决的是如何做的问题，并将执行的结果返回一个代表所需的描绘响应的 JSP（或 Action）的 ActionForward 对象给 ActionServlet 以将响应呈现给客户。

ActionForward 对象给 ActionServlet 以将响应呈现给客户。

过程如图 6.10 所示：

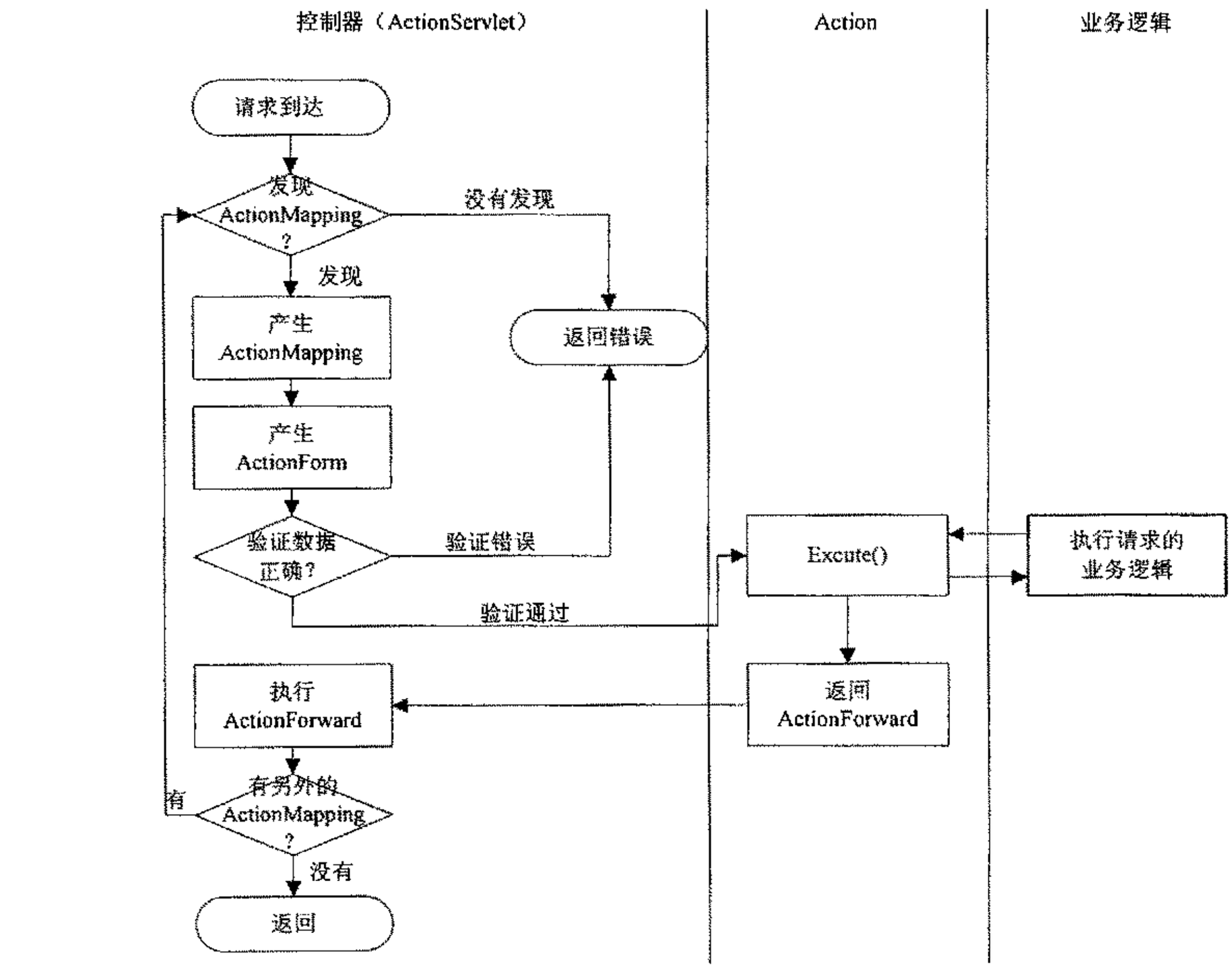


图 6.10　Struts　对请求的处理流程

这里要特别说明一下的是：就是 Action 这个类，上面已经说到了它是 Struts 中真正干活的地方，也是值得我们高度关注的地方。可是，关于它到底是属于控制层还是属于模型层，存在两种不同的意见，一种认为它属于模型层，另一些则认为它属于控制层。显然，将业务对象从 Action 分离出来后有利于它的重用，同时也增强了应用程序的健壮性和设计的灵活性。因此，它实际上可以看作是 Controller 与 Model 的适配器，如果硬要把它归于那一部分，它应该是 Controller 的一部分，换句话说，它不应该包含过多的业务逻辑，而应该只是简单地收集业务方法所需要的数据并传递给业务对象。实际上，它的主要职责是：

- 校验前提条件或者声明
- 调用需要的业务逻辑方法

- 检测或处理其他错误
- 路由控制到相关视图

上面这样简单的描述，初学者可能会感到有些难以接受，下面举个比较具体的例子来进一步帮助我们理解。如：假设，我们做的是个电子商务程序，现在程序要完成的操作任务是提交定单并返回定单号给客户，这就是关于做什么的问题，应该由 Action 类完成，但具体怎么获得数据库连接，插入定单数据到数据库表中，又怎么从数据库表中取得这个定单号（一般是自增数据列的数据），这一系列复杂的问题，这都是解决怎么做的问题，则应该由一个（假设名为 orderBo）业务对象即 Model 来完成。orderBo 可能用一个返回整型值的名为 submitOrder 的方法来做这件事，Action 则是先校验定单数据是否正确，以免常说的垃圾进垃圾出；如果正确则简单地调用 orderBo 的 submitOrder 方法来得到定单号；它还要处理在调用过程中可能出现任何错误；最后根据不同的情况返回不同的结果给客户

6.2.4.2 为什么要使用 Struts 框架

首先，它是建立在 MVC 这种公认的好的模式上的，Struts 在 M、V 和 C 上都有涉及，但它主要是提供一个好的控制器和一套定制的标签库上，也就是说它的着力点在 C 和 V 上，因此，它天生就有 MVC 所带来的一系列优点，如：结构层次分明，高可重用性，增加了程序的健壮性和可伸缩性，便于开发与设计分工，提供集中统一的权限控制、校验、国际化、日志等等；其次，它是个开源项目得到了包括它的发明者 Craig R.McClanahan 在内的一些程序大师和高手持续而细心的呵护，并且经受了实战的检验，使其功能越来越强大，体系也日臻完善；最后，是它对其他技术和框架显示出很好的融合性。如，现在，它已经与 tiles 融为一体，可以展望，它很快就会与 JSF 等融会在一起。当然，和其他任何技术一样，它也不是十全十美的，如：它对类和一些属性、参数的命名显得有些随意，给使用带来一些不便；还有如 Action 类 execute 方法的只能接收一个 ActionForm 参数等。但瑕不掩瑜，这些没有影响它被广泛使用。

6.2.5 应用视图（MVC 中的 V）的设计

MVC 中的 V，含义是表示逻辑，视图。主要由 JSP 生成页面完成视图，Struts 提供丰富的 JSP 标签库：Html、Bean、Logic、Template 等，这有利于分开表示逻辑和程序逻辑。

6.2.5.1 JSP 页面设计

adminLogin.jsp

该页面是网上银行管理员的登录页面

adminQuery.jsp

该页面是网上银行管理员查询客户资料页面

allUserAccount.jsp

该页面是显示网上银行客户资料页面

bankindex.jsp

该页面是网上银行主界面

banner.jsp

该页面是每个页面的页眉，提供 Logo、主页名称以及银行、购物车、登录等相关链接

copyright.jsp

该页面是每个页面的页脚，显示版权信息

left.jsp

该页面是左栏，显示“精品推荐”栏目，以及相关书籍链接

login.jsp

该页面是网上书店客户登录页面

mainMenu.jsp

该页面是网上书店显示书籍列表的页面

newAccount.jsp

该页面是网上书店新客户注册页面

order.jsp

该页面是网上书店显示客户订单页面

pay.jsp

该页面是在网上书店购书的客户向银行付款的页面

right.jsp

该页面是右栏，显示“TOPTEN”“最近销售排行”栏目，以及相关书籍链接

showSortname.jsp

该页面是网上书店显示书籍分类列表的页面

success.jsp

该页面是客户付款后显示购书交易成功的页面

userQuery.jsp

该页面是网上银行用户查询余额的页面

userSum.jsp

该页面是网上银行显示用户余额的页面

需要指出的是，由于使用了 struts 框架，以上所有页面中都实现了零代码，显示和逻辑交错混杂的情况不复存在，这有利于美工和逻辑设计人员的分工合作，这也是 MVC 设计模式要达到的目标之一

6.2.5.2 ActionForm 的设计

ActionForm 类被用作从 HTML 表单捕获用户输入的数据，然后传输给 Action 类。由于 HTML 输入组件并非自身具有输入缓冲器，而用户经常会输入非法的数据，Web 应用程序也需要途径来临时存储数据，意思可能出错时重新显示出来，ActionForm 扮演着一个缓冲器的角色，在检验用户输入数据时，它保存着用户输入的状态。在应用程序中，ActionForm 类同时也扮演“防火墙”这样的角色，有助于在检验规则进行检验的时候，防治业务曾出现可以的或者非法的输入。ActionForm 也被用作将数据从 Action 类回传给 HTML 表单，由于 HTML 表单总是从 ActionForm 中获得数据，这样也使得 HTML 表单更具有一致性。

本应用有一下九个 ActionForm 类（图 6.11）：

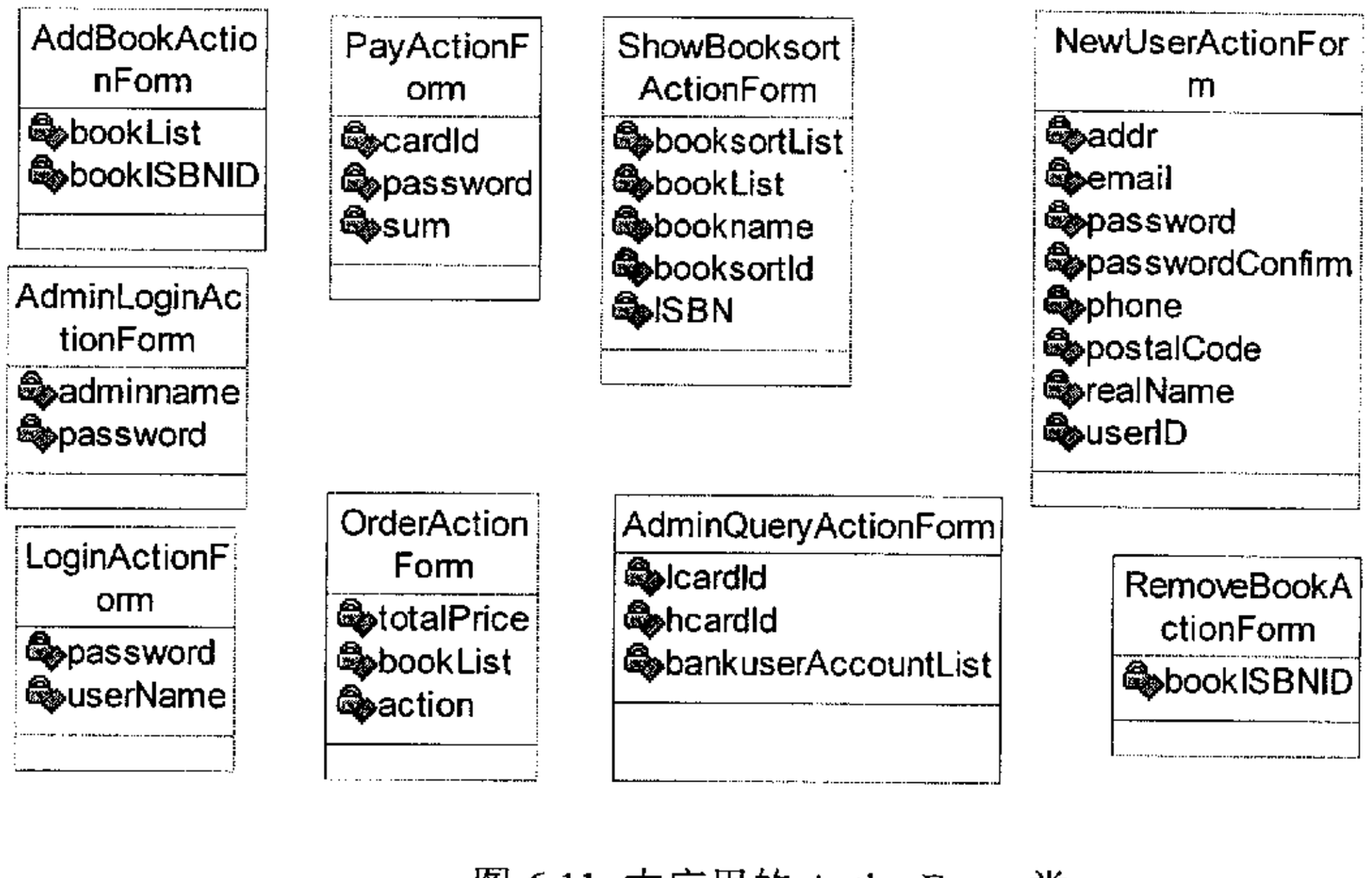


图 6.11 本应用的 ActionForm 类

loginActionForm

该 ActionForm 类接收网上书店用户登录的输入信息

addBookActionForm

该 ActionForm 类接收网上书店用户向购物车添加书籍的输入信息

orderActionForm

该 ActionForm 类接收网上书店用户购物订单的总金额、书籍、确认信息

newUserActionForm

该 ActionForm 类接收网上书店新用户注册信息

payActionForm

该 ActionForm 类接收网上书店用户付款信息

showBooksortActionForm

该 ActionForm 类接收网上书店用户选择书籍分类信息

adminQueryActionForm

该 ActionForm 类接收网上银行管理员查询输入信息

adminLoginActionForm

该 ActionForm 类接收网上银行管理员登录输入信息

removeBookActionForm

该 ActionForm 类接收网上书店用户删除购物车上的书籍的输入信息

一旦创建了由 ActionForm 派生而来的类，就需要在 struts 配置文件 struts-config.xml 中配置它。本应用的配置如图 6.12：

```
<form-beans>
   <form-bean name="loginActionForm" type="edu.sjtu.ebookstore.action.LoginActionForm" />
   <form-bean name="addBookActionForm" type="edu.sjtu.ebookstore.action.AddBookActionForm" />
   <form-bean name="orderActionForm" type="edu.sjtu.ebookstore.action.OrderActionForm" />
   <form-bean name="newUserActionForm" type="edu.sjtu.ebookstore.action.NewUserActionForm" />
   <form-bean name="payActionForm" type="edu.sjtu.ebookstore.action.PayActionForm" />
   <form-bean name="showBooksortActionForm" type="edu.sjtu.ebookstore.action.ShowBooksortActionForm" />
   <form-bean name="adminQueryActionForm" type="edu.sjtu.ebookstore.action.AdminQueryActionForm" />
   <form-bean name="adminLoginActionForm" type="edu.sjtu.ebookstore.action.AdminLoginActionForm" />
   <form-bean name="showBooksortActionForm" type="edu.sjtu.ebookstore.action.ShowBooksortActionForm" />
   <form-bean name="removeBookActionForm" type="edu.sjtu.ebookstore.action.RemoveBookActionForm" />
</form-beans>
```

图 6.12 struts 配置文件中配置 ActionForm

类型字段的值必须为 ActionForm 派生而来的 Java 类名全称。一旦定义好了 form-bean，就能在一个或者多个 action 中使用它。多个 action 可以共享同一个 ActionForm。

6.2.6 应用控制器（MVC 中的 C）设计

MVC 中的 C，含义是程序的控制器。控制器组件负责检测用户的输入，又可能也会更新域模型，为客户端选择下一个视图。控制器有助于把模型的表示层和实际的模型分离开来。在 struts 中，默认的情况下承担 MVC 中的 Controller 角色的是一个 Servlet，叫 ActionServlet。ActionServlet 是一个通用的组件。这个控制组件提供了处理所有发送到 struts 的 HTTP 请求的入口点，它截取和分发这些请求到相应的动作类（这些动作类都是 Action 类的子类）。另外控制组件也负责用相应的请求参数填充 ActionForm，并传给动作类。动作类实现商业逻辑，它可以访问 javabean 或者调用 EJB（本应用中调用 Web Service）。最后动作类把控制权传给后续的 JSP 文件，后则生成视图。所有这些控制逻辑利用 Struts-config.xml 文件配置。

6.2.6.1 Action 类设计

本应用中的 Controller 由默认的 ActionServlet 和 Action 类共同担任，每个 Action 类都代表一个客户端的业务操作，包括访问 Web Service。本应用有以下十四个 Action 类：

AddBookAction	向购物车添加书籍
AdminLoginAction	管理员登录
AdminQueryAction	管理员查询
LoginAction	用户登录
LogoffAction	用户退出
NewUserAction	新用户注册
OrderAction	生成订单
PayAction	付款
RemoveBookAction	从购物车取出书籍
SearchAction	查找书籍
ShowBookAction	显示书籍
ShowBookRandomAction	显示经典书籍
ShowBooksortAction	显示书籍种类
UserQueryAction	用户查询

6.3 服务端设计

6.3.1 JAX-RPC

基于 XML 的远程过程调用的 Java API for XML(JAX-RPC)简化了 Web 服务的创建过程（Web 服务本身就合并了基于 XML 的 PRC）。它定义了 Java 类型和 XML 类型之间的映射(序列化)，目的是为了隐藏 XML 的细节，提供一种熟悉的方法调用范例。JAX-RPC 完全包容了 Web 服务的异构性。它允许一个 JAX-RPC 客户端与另一个部署在不同平台上的，用不同语言编写的 Web 服务进行对话。JAX-RPC 提供了一系列规范，包括调用模式、客户端生成、参数模式、Java 到 WSDL 和 WSDL 到 Java 的类型映射、以及调用 Web 服务的客户端 APIs。

怎样把一个 Java 对象序列化?一个简单的例子:

对于 Java 接口:

```
public interface Hello
{
    public String sayHelloTo(String name);
}
```

客户程序在调用 sayHelloTo()方法时提供了一个名字，它希望从服务器接收到一则个性化的“Hello”信息。

必须串行化方法调用，并把消息发送给服务端。

假设要模拟 sayHelloTo(“Deng”)调用， 发送的请求可以是：

```
<?xml version=“1.0”?>
<Hello>
<sayHelloTo>
<name>Deng</name>
</sayHelloTo>
</Hello>
```

在这里，接口的名字作为根结点。方法名字和参数名字都当作节点。

6.3.2 Apache Axis

我们使用 Apache Tomcat-Axis 来发布 Web 服务

Apache Tomcat-Axis 组合提供一个 JAX-RPC 1.0 兼容的运行时引擎，它具有客户

端和服务器端库以及部署工具。图 6.13 详细阐述了标准的同步请求--响应模式的 Web 服务调用框架。封装了 Web 服务客户端的 Application1 使用 JAX-RPC 运行时环境执行一个远程过程调用请求封装了 Web 服务服务器端的 Application 2 的公共方法。客户端使用运行时库把 Java 对象序列化成一个 SOAP 消息，然后通过 HTTP 协议把该消息发送给 Web 服务端点。当部署在 Apache Tomcat 上的 Web 服务接收到这个请求时，服务器端的 JAX-RPC 运行时库把 SOAP 消息反序列化为 Java 类型，并且调用 Web 服务上的方法，接着对 Application 2 进行调用。在处理完请求后，Web 服务以类似的方式把响应发回给客户端。

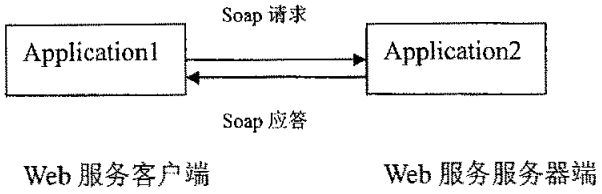


图 6.13　Web Service 请求响应模式

6.3.3 应用模型(MVC 中的 M)的设计

如前所述,MVC 中的 M 代表着模型。一个应用的模型组件对于该应用来讲是最有价值的软件部分。模型包含着商业实体以及如何管理访问和修改数据的规则。这些工作在一个特定的场所完成，以维护有效数据的完整性，减少冗余，增强可重用性，这十分重要。

模型具有不同的含义,最普通的含义是表示事物的面貌.模型可以表示一个买卖货物的商店,可以表示一套方法,可以表示房屋等等。在本应用中，模型表示一个整合部分银行功能的数字书店。

6.3.3.1 开发概念模型

在问题域的分析中，概念模型应该基于问题域的真实实体来开发的。在概念模型中的实体与系统软件无关，而与现实中的那些体现业务的基本情况的实体有更多的关系。概念模型通常说明概念、概念间的关联以及每个概念所固有的属性等，在该模型中对于行为不作描述。概念模型仅有系统用例而来，其目的在于有助于识别那些在设计阶段很有可能成为类的实体，同时也为了更好地理解问题域。图 6.14 展示应用的概念模型。

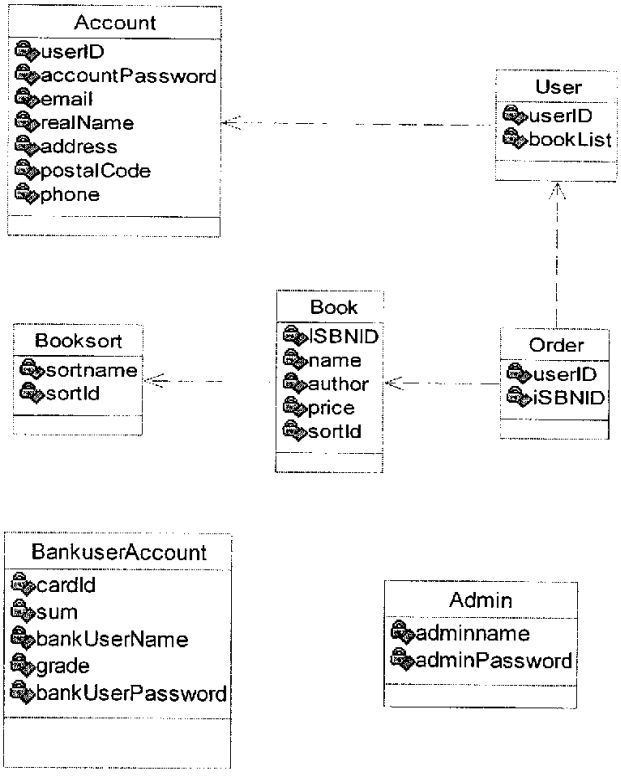


图 6.14　本应用的概念模型

在图 6.14 中，注意到只有实体的关系和属性被显示出来，没有表示行为的方法。该图清晰地展示了问题域中所用到的实体。每个模型的意义如下:

Account：网上书店的注册用户信息。

Ueser：在线用户信息

Order：订单

Book：库存书籍

Booksort：库存书籍分类

BankuserAccount：网上银行客户

Admin：网上银行管理员

通过检查概念模型，发现系统构件的问题，早期进行修改，对今后我们的系统开发很有好处，越晚期的更改，将付出越大的代价。

6.3.3.2 建立业务对象

在概念模型的基础上创建业务对象是很方便的事情，业务对象包括数据和行

为。该对象也是数据库表中一条条记录的具体表示。

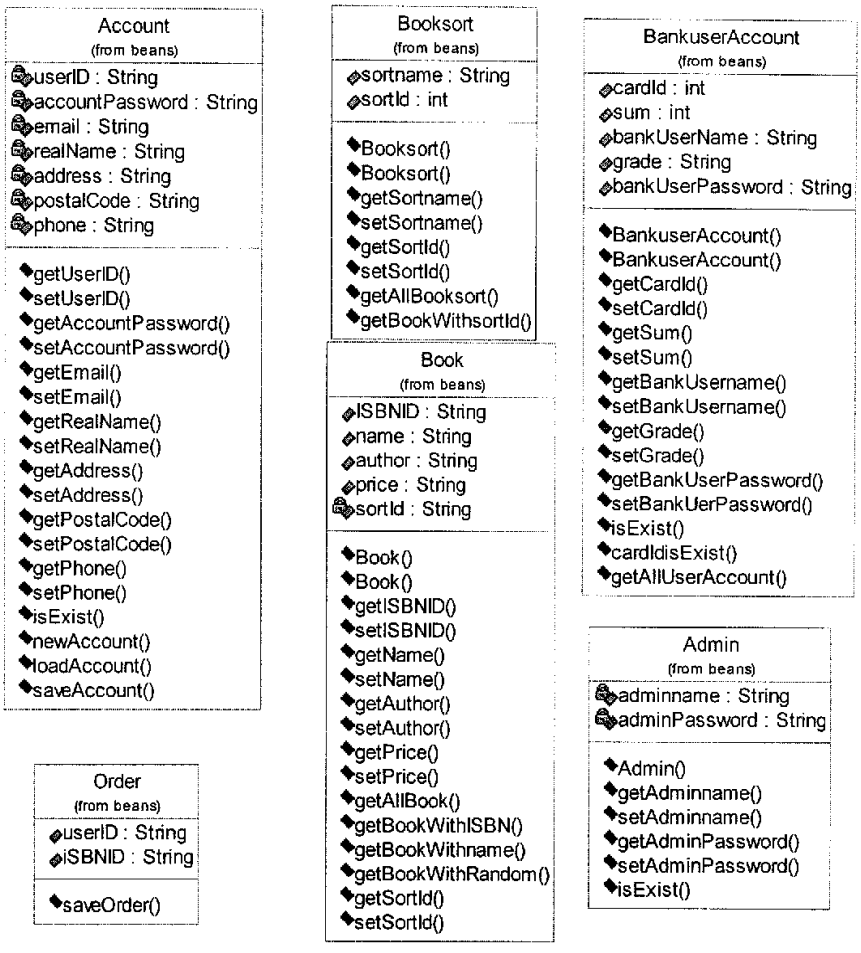


图 6.15　本应用的业务对象

可以看到，业务对象已经具有了处理数据的方法和规则。主要方法：

Account 类：boolean isExist(String userID,String accountPadssword)　判断该用户是否合法

void newAccount(String userID,String password,String email,String realName,String address,String postalCode,String phone)　建立新用户档案

void loadAccount()　供用户调用档案查看

void saveAccount()　用户修改档案后存储档案

Order 类：void saveOrder(String ISBNID,String userID)　存储用户订单

Admin 类：boolean isExist(String adminname,String adminPassword)　判断管理员

是否合法

Book 类： BookHelp[] getAllBook() throws SQLException 查询所有书籍，返回一个包含所有书籍的对象数组

BookHelp getBookWithISBN(String ISBN) 通过书刊号查询书籍，返回包含一本书信息的对象

BookHelp[] getBookWithname(String bookname) 通过书刊名查询书籍，返回对象数组

Booksort 类：BookHelp[] getBookWithsortId(int sortId) 通过书类型号，返回所有该类型号书籍的对象数组

BankuserAccount 类：boolean cardIdisExist(int cardId) 判断银行卡是否合法

BankHelp[] getAllUserAccount(int lcardId,int hcardId) 管理员查询卡号范围内的所有客户信息，返回对象数组

6.3.3.3 建立数据库

与业务对象相对应，我们建立以下数据库和数据表：

网上书店数据库名：ebookstore

表格名	表格说明	表格字段	字段描述
Account	网上书店的注册用户信息表	userID	唯一主键。唯一标识用户的字符串。 限长 20
		password	用户密码
		email	用户电子邮箱
		realName	用户真实姓名
		address	用户地址
		postalCode	用户邮政编码
		phone	用户电话
Book	网上书店库存书籍信息表	indexID	唯一主键。唯一标识书籍的整数。 限长为 20。
		ISBNID	书刊号
		Name	书名
		Author	作者

		Price	价格
		sortId	书籍类型号，与 ebooksort 表中的 sortId 关联
Ebooksort	网上书店库存书籍类型表	Sortname	书籍类型名称
		sortId	为一主键。唯一表示书籍类型号
Yourorder	用户订单表	userID	用户名唯一标识
		ISBNID	书刊号
		price	书籍价格
		Num	书籍数量

表 6-1 网上书店数据表说明

网上银行数据库名：bank

表格名	表格说明	表格字段	字段描述
Admintable	网上银行管理员表	Adminname	唯一主键。唯一标识管理员的字符串。 限长 20
		password	管理员密码
Hsumtable	网上银行客户表	cardID	唯一主键。唯一标识银行客户卡号的整数。 限长为 20。
		Sum	帐户余额
		Username	客户姓名
		Grade	客户级别
		Password	客户密码
		ISBNID	书刊号
		price	书籍价格
		Num	书籍数量

表 6-2 网上银行数据表说明

数据库建立好后，就可以把业务对象映射到数据库，可以有多种途径，比如对象—关系映射框架，直接用 JDBC 调用。由于我们的演示程序数据不复杂所以这里使用的是直接 JDBC 调用。

6.3.3.4 发布服务

M 构建好后需要把业务对象发布成 Web Service，一旦发布成功，业务对象的方法和属性就可以被远程客户访问和调用。这个发布工作我们使用 Apache Axis 来自动进行。从 Axis 提供的控制台上我们可以看到业务对象及其方法已经发布成 Services（如图 6.16）：

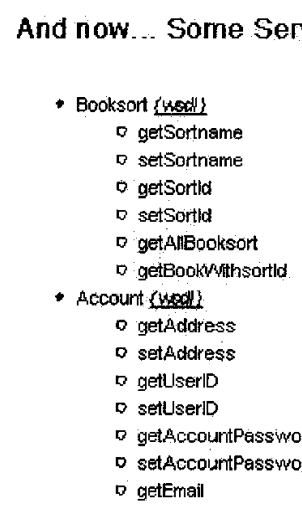


图 6.16 在 Axis 控制台上查看 Web Service 片段

从图 6.16 中，我们注意到，每个发布成 Web Service 的对象都产生一个 WSDL 文件，如前所述，WSDL 是描 述 Web 服务的技术调用语法，WSDL 描述说明的是 Web 服务的以下三个基本属性：

服务做些什么--服务所提供的操作(方法)。

如何访问服务--数据格式详情以及访问服务操作的必要协议。

服务位于何处--由特定协议决定的网络地址，如 URL。

图 6.17 是 Account 类的 WSDL 文件片段：

```
<wsdl:service name="AccountService">

   <wsdl:port binding="impl:AccountSoapBinding" name="Account">

      <wsdlsoap:address location="http://localhost:8080/DigitalComWS/services/Account"/>

   </wsdl:port>

</wsdl:service>
```

图 6.17 Account 类的 WSDL 文件片段

图 6.17 清楚的说明了服务位于何处。

由于本应用没有 UDDI 使用，因此客户对 Web Service 的查找是通过对 WSDL 文件的导入来完成的，通过对 WSDL 文件的导入，客户可以找到 Web Service，并且能够按照 WSDL 的描述调用其方法。在本应用中，对应于六个业务对象，一共有六个 WSDL 文件。如图 6.18：

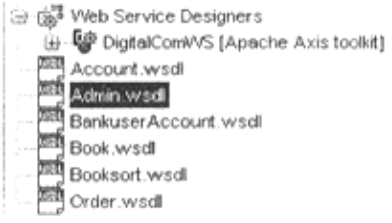


图 6.18　在 Axis 控制台查看本应用的 WSDL 文件

6.4 典型应用在 XML Engine 上的运行效果

为了说明本应用达到了设计时的目的，我们分别在不使用 XML Engine 和使用 XML Engine 的情况下截获网络上的 SOAP 包，进行对比

6.4.1 不使用 XML Engine 时的 SOAP 包

以网上银行管理员查询客户信息的操作为例，得到的 SOAP 包片段如下（片段 6.1）：

```
<multiRef id="id0" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns3:BankHelp"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns3="http://helps.ebookstore.sjtu.edu">
  <bankUserName xsi:type="xsd:string"></bankUserName>
  <bankUserPassword xsi:type="xsd:string" xsi:nil="true"/>
  <cardId xsi:type="xsd:int">10000</cardId>
  <grade xsi:type="xsd:string">manager</grade>
  <sum xsi:type="xsd:int">100</sum>
</multiRef>
<multiRef id="id18" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns4:BankHelp" xmlns:ns4="http://helps.ebookstore.sjtu.edu"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
  <bankUserName xsi:type="xsd:string"></bankUserName>
  <bankUserPassword xsi:type="xsd:string" xsi:nil="true"/>
  <cardId xsi:type="xsd:int">10007</cardId>
  <grade xsi:type="xsd:string">leader</grade>
  <sum xsi:type="xsd:int">800</sum>
</multiRef>
<multiRef id="id15" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns5:BankHelp" xmlns:ns5="http://helps.ebookstore.sjtu.edu"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
  <bankUserName xsi:type="xsd:string"></bankUserName>
  <bankUserPassword xsi:type="xsd:string" xsi:nil="true"/>
  <cardId xsi:type="xsd:int">10004</cardId>
  <grade xsi:type="xsd:string">manager</grade>
  <sum xsi:type="xsd:int">500</sum>
</multiRef>
```

片段 6.1 不使用 XML Engine 时的 SOAP 包片段

从片段 6.1 可以得出结论，本应用达到了设计要求：SOAP 包正常，上面的片段是一个复杂结构数据的 SOAP 包的传送，该片段包含三个卡号的信息：10007、10004 和 10000。用户可以任意的指定加密对象，比如：卡号 10004 的所有信息、grade 为 manager 的所有卡的信息、金额在 800 元以上的所有卡的信息等等。

6.4.2 使用 XML Engine 后的 SOAP 包

同样的，我们以网上银行管理员查询客户信息的操作为例，但是我们在用户配置文件中配置为对卡号 10007 加密，得到的 SOAP 包片段如下（片段 6.2）：

```
<multiRef id="id0" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns3:BankHelp"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns3="http://helps.ebookstore.sjtu.edu">
  <bankUserName xsi:type="xsd:string"></bankUserName>
  <bankUserPassword xsi:type="xsd:string" xsi:nil="true"/>
  <cardId xsi:type="xsd:int">10000</cardId>
  <grade xsi:type="xsd:string">manager</grade>
  <sum xsi:type="xsd:int">100</sum>
</multiRef>
<EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#"
Type="http://www.w3.org/2001/04/xmlenc#Content">
<EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
<KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
<KeyName>/tmp/Cert/Zhang/Private/deskey.bin</KeyName>
</KeyInfo>
<CipherData>
<CipherValue>q/FfW7g3qirDS1XPcXEHdXf5g6S8xs6+EWfe3LuVqvEJvneKzgWP
szFcZ6/otLKI
OCCnKy/6SghRVqdNiycdisE6A4+HZ38/zCNHAC5FsitpHNiKcn1Hiuq0QVUtDO
VY
WOeecq0ydc8GqN48Cvw1jGg8KaCrNdl86Ga/rJL+NQk27qJqkDyq0x0WW8yAWh
30
```

5cXWIPmuZiVfcE6F9ybfmeJtjjcbM7/JmpOi+5ZZV0TIoZGwpIfwwgM5k2VGITU
m
x0XHojSXTpMZz6FARV6VHjkSozEyTJfzmWAgwgVnOIsC45TtZ9fJHnNsPbx1sV
3e
FKEP8B4nfLRZ+aWknZ9u70QepnIZyGQxM3rn/j7RuIGbtFLKZeWgN/v+k2X8ZS
aG
uP4sCAIObqYZc4QFeM6UITCyOMI8Dvxi1QolIYSV3/SQ8+YOiCaSpdP2vCuXQ
1SM
tokCLrXzlC8ebLZNMb7Z7htwE6h/ApRG26KgdpQ3UdAQmFlaDmpAt7A2vyJFnf
8A
JlXwz8KUIwd4X4f+Hx0Q4+LJzH0m0nUK5IXwWdYu0wpyqpUKmH5o13mVuGg
+olaT
qUkdD4tyh+TEDGdFMyv59/Uy2RxPCVQlKMs8kMqY7IgkrIk5WW03hnq36oL5E
f6D
</CipherValue>
</CipherData>
</EncryptedData>
<multiRef id="id15" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns5:BankHelp" xmlns:ns5="http://helps.ebookstore.sjtu.edu"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
<bankUserName xsi:type="xsd:string"></bankUserName>
<bankUserPassword xsi:type="xsd:string" xsi:nil="true"/>
<cardId xsi:type="xsd:int">10004</cardId>
<grade xsi:type="xsd:string">manager</grade>
<sum xsi:type="xsd:int">500</sum>
</multiRef>

片段 6.2　使用 XML Engine 后的 SOAP 包片段

比较片段 6.1 和片段 6.2，可以发现：卡号为 10007 的<multiRef>标签包括该标签的所有内容已经被 EncryptedData 标签替换，内容已经加密。

同样的，对解密、签名、验证、过滤等功能测试也顺利通过，同时进行的容量测试也顺利通过，这表明，本系统和应用都是成功的。

第七章 今后的工作

目前我们设计并实现的 XMLEngine 安全产品，配合硬件加密机和密钥管理系统，已经初步通过阶段性测试和验收。但对于一个商业化的成熟的产品来讲，还有许多功能和题材可以挖掘。

7.1 SOAP 安全

我们的安全产品针对以 XML 为数据格式的报文，这其中当然包括 SOAP 包。SOAP 的两个主要设计目标是简单性和可扩展性，这就意味着有一些传统消息系统或分布式对象系统中的某些性质将不是 SOAP 规范的一部分。自从 SOAP 规范发布以来，SOAP 规范的加密性、认证和授权等安全机制一直受到人们的广泛关注，这三个方面对于任何的 B2B 来说都是很重要的，但 SOAP 标准在制定规范时并没有过多考虑 SOAP 的安全性要求，因为 SOAP 一个很重要的设计目标就在于它的简单性，尽可能的利用已有的标准和协议来实现相应的功能，而不是另起炉灶，重新定义一个崭新的协议，如果这样的话，会大大的降低它的实用性和兼容性。

SOAP 安全性扩展：数字签名（SOAP-DSIG）定义了用数字方式签名 SOAP 消息及确认签名的句法和处理规则。数字签名使初始用户和软件能够可靠地发送信息。但 SOAP 并不包括签名消息的规定，因此也无此安全性。SOAP-DSIG 必须使用安全套接字层 SSL(Secure Sockets Layer)，SSL 是由 Netscape 公司开发的一套 Internet 数据安全协议，它已被广泛地用于 Web 浏览器与服务器之间的身份认证和加密数据传输，而针对 XML 的数字签名不能保证发送方的身份和签名文档的发送次数，同时使用 XML 安全性也无法对发送方身份进行验证，这些又是利用 SSL 安全套接字层通信的基本功能之一，将二者结合起来优势互补。

7.2 AAA 门户认证系统

随着 Internet 的发展，越来越多的应用通过网络得以实现，拨号用户、专线用户以及各种商用业务的发展使 Internet 面临许多挑战。如何安全、有效、

可靠的保证计算机网络信息资源存取及用户如何以合法身份登陆、怎样授予相应的权限，又怎样记录用户做过什么的过程成为任何网络服务提供者需要考虑和解决的问题。正是基于此需求，AAA 协议逐渐发展完善起来，成为很多网络设备解决该类问题的标准。

AAA 即为 Authentication（认证），Authorization（授权），Accounting（记费）。认证（Authentication）即辨别用户是谁的过程。通常该过程通过输入有效的用户名和密码实现；授权（Authorization）：对完成认证过程的用户授予相应权限，解决他能做什么的问题。在一些身份认证的实现中，认证和授权是统一在一起的；计帐（Accounting）：统计用户做过什么的过程，用户使用的时间和费用。通常可以用用户占用系统时间、接受和发送的信息量来衡量。

AAA 通过 Authentication、Authorization 和 Accounting 集成控制了用户在自己特定角色特点所遵循的原则下如何访问多个网络及特定网络下的某个平台也及某种应用服务。通常意义上的 AAA 服务器都具有用户认证、授权处理用户请求以及收集用户使用情况的相关数据的功能。对一个服务提供商来说，这样的 AAA 服务器应该有一个应用型的特定模式的应用界面（接口），通过这个界面（接口）的服务必须通过授权。在实际使用中，AAA 服务器都带有一个用户数据库（可以是某系统用户数据库或独立的数据库系统），这个数据库中含有用户的初始化信息，它可以反映合法的属性值以及每个用户所享有的权限。通过它和客户端软件的数据交流来实施相关操作。

Authentication 通过终端用户的识别属性来判定它是否有进入网络的权限。终端用户一般需要提供一个用户名（该用户名在这个认证系统中应该是唯一的）和对应该用户名的口令。AAA 服务器将用户提交的信息和储存在数据库的和用户想关联的信息比较，如果匹配成功的话该次登陆生效，否则拒绝用户请求。

当用户通过认证以后，Authorization 就决定了该用户访问网络的全线范围及所享有何种服务。就有可能包括提供一个 IP 地址，或某种规则的滤镜以来确定那些应用和协议可以被支持。在 AAA 管理模式下 Authentication 和 Authorization 通常可以一起执行。

Accounting 提供了收集用户使用网络资源情况信息的方法。通过该类数据的收集，可以提供网络审查、发展以及结构调整的一些依据。
目前 AAA 应用系统越来越多的采用 XML 作为消息传递的载体，在这种情况下，使用 XMLEngine 平台就非常合适。利用 XMLEngine 基本组件构建“AAA 安全

门户”，实现统一的用户认证和授权功能。

安全消息平台的基本想法是在客户端对用户应用发出的消息（邮件、远程过程调用等）实施一定的安全保护，如加密、签名等，可能还会有对阅读者的访问控制等等。在传递、存储过程中，消息都处于被保护状态。消息将要被使用时（比如读取邮件），安全消息平台将消息还原成原始状态，供用户使用。

为了保证用户在使用安全消息平台时的透明性，我们使用 XMLEngine 组件进行支持，这需要对 XMLEngine 平台作出一些拆分和重组。

参考文献

[1]ITU, Information Technology – Open system interconnection – The Directory: Public-key And Attribute Certificate Frameworks, ITU-T Recommendation X.509, ITU, 2000

[2]ITU, Information Technology – Open system interconnection –THE DIRECTORY: AUTHENTICATION FRAMEWORK, ITU-T Recommendation X.509, ITU, 1997

[3]X.500 Editing Meeting, Copenhagen, 21-28 October 1999

[]Housley.R, Ford.W, Polk.W, etc., Internet X.509 Public Key Infrastructure Certificate and CRL Profile, RFC 2459, IETF, 1999

[4]C. Adams, S. Farrell, Internet X.509 Public Key Infrastructure Certificate Management Protocols, RFC 2510, IETF, 1999

[5]Sharon Boeyen,"X.509 4th edition:Overview of PKI &PMI Frameworks(Entrust Inc.)" http://www.entrust.com/resources/pdf/509_overview.pdf (2000)

[6]Sharon Boeyen, "X.509 4th edition :X.509 (2000): What's new? (Entrust, Inc.)" URL : http://www.entrust.com/resources/pdf/509_new.pdf (2000)

[7]D.W.Chadwick "An X.509 Role Based Privilege Management Infrastructure", Business Briefing - Global InfoSecurity 2002, World Markets Research Centre Ltd, ISBN: 1-903150-52-3, Oct 2001. On accompanying CD-ROM Reference Library/03.pdf

[8]D.W.Chadwick, A. Otenko. "RBAC Policies in XML for X.509 Based Privilege Management", accepted for SEC2002, Egypt, May 2002

[9]Toni Nykänen , Attribute Certificates in X.509 , http://www.tcm.hut.fi/Opinnot/Tik-110.501/2000/papers/abstract_nykanen.html

[10]D.W.Chadwick, O.Otenko, ISI, University of Salford, Salford, M5 4WT, The PERMIS X.509 Role Based Privilege Management Infrastructure, http://sec.isi.salford.ac.uk/download/SACMATfinal.pdf

[11]The PERMIS Consortium, The PERMIS Project, Permis COMPA v0_6 ENG.ppt PrivilEge and Role Management Infrastructure Standards validation, PERMIS presentation v1 ENG.ppt

[12]Baltimore:SelectAccess Product Overview, http://www.baltimore.com PKI based e|security, http://www.baltimore.com/pki-booklet.pdf

[13]Tivoli SecureWay Authorization, http://www.simc-inc.org/archive9900/Feb00/clark/sld001.htm http://www.tivoli.com

[14]Security Solutions,http://www-3.ibm.com/security/index.shtml

[]IBM @server mainframe adds advanced security features http://www.ibm.com/news/us/2002/03/256.html

[15]Authentication &Authorization Complementary Solutions for Securing the Digital Enterprise, BioNetrix & Netegrity – Complementary Solutions For Securing the Digital Enterprise, netegrity.pdf

[16]Public-Key Infrastructure– The VeriSign Difference, http://www.verisign.com EntrustGetAccess:Features&Benefits,http://www.entrust.com/getaccess/features.htm

[17]Entrust' XML Strategy for Authorization, Entrust Inc. http://www.entrust.com

[18]Web Services Trust and XML Security Standards, Entrust Inc. http://www.entrust.com

[19]Mark Glaser, SAML Looks to Allay XML Security Concerns http://dcb.sun.com/practices/Web Services/overviews/overview_saml.jsp

[20]James Kobielus, Simplification, not XML, is the key to PKI success,http://www.nwfusion.com/columnists/2001/0507kobielus.html

[21]Bruce Weiner,IBM Tivoli Access Manager for e-business v3.8 AuthMark Performance,http://www.mindcraft.com/whitepapers/pd38/pd38.html

[22]Bruce Weiner,Securant ClearTrust Version 4.2 AuthMark LoginPerformance,http://www.mindcraft.com/whitepapers/ct/ct42.html

[23]Bruce Weiner, Baltimore Technologies SelectAccess 3.1 AuthMark Performance,http://www.mindcraft.com/whitepapers/sa31/sa31.html

[24]Portals: The evolution of Extranet Access Management, http://www.entrust.com/resources/pdf/hurwitz_portals.pdf

[25]Entrust Unveils U.S. Government Blueprint to Link National Security and E-Government Initiatives，http://www.entrust.com/news/files/11_08_01_774.htm

[26]SDN.801: ACCESS CONTROL CONCEPT AND MECHANISMS, National Security Agency,USA http://www.sse.ie

[27]Chuck Cavaness,Jakarta Struts, O'Reilly 2002

[28]James Goodwill, Mastering Jakarta Struts, Wiley Publishing, Inc. 2002

[29]谭寒生，佘堃，周明天，X.509v4 的改进，电子学会第八届青年学术年会 2002CIEYC，2002.7；

[30]Harvey M.Deitel Java Web Services for Experienced Programmers.机械工业出版社,2003.7

[31]丁鹏，刘方，邵志峰，何丙胜，Struts 技术揭秘及 WEB 开发实例，清华大学出版社，2004

致　谢

本文在导师周明天教授的精心指导下完成。周老师渊博的学识、严谨的治学态度、丰富的实践经验、对学科发展方向的敏锐眼光给予了我极大的启迪和引导。导师在我的专业课程学习、研究课题选题、研究方法、论文写作等方面都进行了精心的指导，使本文得以顺利完成。他对我的研究工作的严格要求，促使我对科学研究的精神、方法、内在规律有所领会，这些收获是我今后工作的重要基础。在此，我向周老师表示深深的感谢。

感谢余坤副教授对我的指导和帮助。课题期间一直得到余老师很多鼓励和有益的启发。感谢陈福莉老师在课题期间给予我的很多帮助和鼓励。

感谢电子科技大学-卫士通联合实验室的所有同学，尤其是吴远程博士、谭良博士、林海宇、王成等同学的帮助，以及很多有益的探讨。

感谢我的父母和所有一直关心帮助我的亲人及朋友们，他们给了我许多支持与鼓励。

感谢所有参考文献的作者们，他们的辛勤工作和成果给了本文工作以极大的帮助和启发。

最后，衷心感谢为评阅本论文而辛勤工作的老师。

个人简历、在学期间的研究成果及发表的学术论文

个人简历

邓彦杰，四川简阳人，1971 年 3 月出生。于 1993 年 7 月在四川轻化工学院获得工学学士学位，于 2002 年 9 月进入电子科技大学计算机科学与工程学院攻读工学硕士学位。主要研究方向：中间件，信息安全

研究成果

1） 基于 XML 的安全通信平台—XML Engine 1.0 版

2） Role-Based Access Control(RBAC)基于角色的访问控制系统 1.1 版(四川省科技进步三等奖)

学术论文

邓彦杰，周明天，一种 XML 安全系统的设计与应用，《通信技术》2004.1