返回总目录

目　　录

第四部分　XML 高阶

第十四章　XML 在数据库中的应用

本章着重介绍 XML 与数据库技术的结合，探讨 XML 作为一种新技术在“火爆”的数据库领域的应用方式与前景。具体阐述了 XML 存储获取数据，对关系数据的建模以及开发元数据的一些方法和实例，同时对 XML 查询语言给予了极大的关注，进行了比较详细的介绍。

本章包括以下内容：

- XML=数据库？
- 来自业界的支持
- XML 查询语言——XQL

14.1　XML=数据库

在当今信息时代，互联网技术的迅速发展改变了人们的生活和工作方式，海量的数据交换和数据存储为人类提供了一个广阔的资源共享空间。而这其中数据库技术起到了关键作用。如果读者熟悉数据库，那么会注意到，XML 文档在很多方面都与传统的关系和对象数据库相似。一旦掌握了可以精确表示文档的 XML 语言，我们就能像处理其他类型的数据一样来处理文档了。下面我们先看一个具体的记录计算机书籍信息数据的 XML 文档。

Pcbook.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE dlib SYSTEM "pcbook.dtd">
<dlib>
    <pcbook>
        <dc_title> Microsoft Windows 2000 网络基础结构</dc_title>
     <dc_creator>[美]John Shum</dc_creator>
        <dc_subject>计算机技术</dc_subject>
        <dc_description>书中全面介绍了如何实现 Microsoft Windows 2000 网络基础结构。全书共有 16 个单元、3 个附录，内容包括：关于 Microsoft Windows 2000 网络基础结构的初步介绍、利用 DHCP 自动分配 IP 地址、利用 DNS 实现名称解析、利用 WINS 实现名称解析、利用公共密钥基础结构配置网络安全性、利用 IPSec 配置网络安全性、配置远程访问、支持到网络的远程访问、利用 IAS 扩展远程访问能力、将基于 Windows 2000 的服务器配置成路由器、针对网络配置 Internet 访问、配置 Web 服务器、利用 RIS 部署 Windows 2000 Professional、管理 Windows 2000 网络、查找和解决 Windows 2000 网络服务中的问题、在操作系统之间配置网络连接等。
        </dc_description>
        <dc_publisher>北京希望电子出版社 </dc_publisher>
        <dc_contributor>北京希望电子出版社 </dc_contributor>
        <dc_date>2000-12-??</dc_date>
        <dc_type>外文翻译计算机书籍</dc_type>
        <dc_format>电子文档、源数据空间：6G</dc_format>
        <dc_identifier>7-900056-00-9/TP·00</dc_identifier>
        <dc_source>IDG</dc_source>
```

```
        <dc_language>chinese</dc_language>
        <dc_relation>http://www.bhp.com.cn</dc_relation>
        <dc_coverage>国外 计算机书籍 2000 年</dc_coverage>
        <dc_righits>北京希望电子出版社</dc_righits>
    </pcbook>

    <pcbook>
        <dc_title>PHP 4.0 程序设计</dc_title>
        <dc_creator>网胜工作室 译著</dc_creator>
        <dc_subject>计算机技术</dc_subject>
        <dc_description>本书列举了大量有实用价值的例子来讲述 PHP 和 MySQL 的编程。对 PHP 语
        言的基本语句、函数类型和具体应用、MySQL 的使用，特别是对 PHP 与 MySQL 的综合应
        用都进行了深入讲解。本书是数据库、网站开发和网页设计人员的参考书。既适合没有任何
        经验的编程初学者，也适合有一定网络程序编程经验的程序人员；既可以作为 PHP 和 MySQL
        的学习教材，也可以作为 PHP 编程参考手册来使用。</dc_description>
        <dc_publisher>北京希望电子出版社</dc_publisher>
        <dc_contributor>北京希望电子出版社</dc_contributor>
        <dc_date>2000-8-??</dc_date>
        <dc_type>外文翻译计算机书籍</dc_type>
        <dc_format>电子文档、源数据空间：3G</dc_format>
        <dc_identifier>7-900049-04-5/TP·04</dc_identifier>
        <dc_source>IDG</dc_source>
        <dc_language>chinese</dc_language>
        <dc_relation> http://www.bhp.com.cn </dc_relation>
        <dc_coverage>国外 计算机书籍 2000 年</dc_coverage>
        <dc_righits>北京希望电子出版社</dc_righits>
    </pcbook>
</dlib>
```

上述 XML 文档引用 pcbook.dtd，把计算机书籍的各种属性信息数据储存在特定的索引中，这样可以很容易地根据书名（Title）、作者（Creator）或标识（Identifier）等对文档进行查询和排序，具体操作我们将会在第三节 XQL 中详细讲述。

14.1.1 文档、数据、数据库

在使用 XML 和数据库之前，我们需要考虑：使用数据库的目的是什么？是要显示数据？还是需要一个保存自己主页的空间？数据库在电子商务运用程序中是要把 XML 当做数据传输格式传送吗？

举例说明，假设我们是把 XML 做为一种数据传输格式使用在电子商务运用程序中。那么意味着需要传输的数据格式主要具有高度规范结构，那么在 XML 中的那些自己的编码规范对我们而言并不重要了，这样我们的兴趣就仅仅是在数据上而不是在这些数据在文档中如何物理存储了。如果运用程序关系简单，那么一个关系数据库和数据传输中间件就能够满足我们的需求；如果关系庞大和复杂，那么我们就需要一个完全支持 XML 的开发环境了。

从另一方面来说，假设我们是要实现从杂散的 XML 文件中创建一个网站的功能。我们不仅需要管理这个网站，还要提供给用户查询其中内容的功能。这时我们的文件的格式将是高度的不规范，而实体的使用对我们来说就变得很重要，因为这些文件的结构是网站的基本功能需

求。在这个例子中，就需要一些 native XML 数据库而不是普通的关系数据库，执行解释、XML 实体使用和支持查询语言（例如 XQL）。

XML 作为结构化的文档，其很多特性与数据库是一样的。它可以保存抽象数据，并防止它们与修饰信息混在一起。

进一步来说，我们还可以使用这种结构化标记来表示一些传统概念中无法用文档表示的数据，如：基因模型、元素原子图形等。这些对于传统的数据库来说处理起来过于复杂甚至不能实现。现在，XML 为我们打开了文档的大门，将各种各样的数据集成进来。文档的数据库化和数据库的文档化是 XML 得到计算机业界全力支持的重要原因。

对于大型复杂的文档，XML 是一种理想语言。它不仅允许指定文档中的词汇，还允许指定元素之间的关系。例如，如果把许多商业合同放在一个 Web 中，那么就需要每个合同都有一个电话号码和电子邮件地址。如果是正在向数据库中输入数据，那么就能保证不会遗漏字段。甚至当没有输入数据时，还可以提供一个缺省值来使用。

XML 还提供一种集成多个数据源数据并且作为单个文档显示的客户机端包括机制。数据位置甚至可以重新安排。根据用户的操作，部分数据可以被显示或者隐藏。当使用大型信息仓库如关系数据库时，这种结构非常有用。

1．数据和文档的对比

也许在大多数情况下，判断是否采用数据库最重要的因素是我们使用数据库是用来保存数据还是保存文件。如果想保存数据，我们需要的数据库主要是面向数据存储的，例如一个关系数据库或者是一个面向对象的数据库，或者也可以是一个在数据库和 XML 文档之间传递数据的中间件。从另一个角度来说，如果想存储文件，那么我们就需要一个专门设计来存储文件的内容管理系统。

究竟是需要存储数据还是文件取决于 XML 文件，因为 XML 文件分为两类：数据为主的文件和文档为主的文件。

（1）数据为主的文件。数据为主的文件的特点是结构非常规范，数据格式良好（就是说，数据中最小的独立单元是 PCDATA-only 元素级别或者是一个属性）和少量甚至没有混合内容。其中同类型元素和 PCDATA 的出现顺序并不重要。例如 XML 文件内容是百货超市销售清单、长途客车运行时间表、书店最新图书列表等。数据为主的文件经常被用来完成机器理解和处理，这时 XML 的调用是多余的，它仅仅是一种数据传输。

例如，下面的销售单就是一个数据为主的文档：

```
<Orders>
    <BookOrder SINumber="11111">
        <Customer ID="12345">
            <CustomerName>John Hill</CustName>
            <Street>108 Green St.</Street>
            <City>London</City>
            <PostCode>26789</PostCode>
        </Customer>
        <OrderDate>20000818</OrderDate>
        <Line LineNumber="1">
            <Part PartNumber="10001">
```

```
            <Description>
                <p><b>XML Learning:</b><br />
                A good book to learn XML.</p>
            </Description>
            <Price>49.99</Price>
        </Part>
        <Quantity>10</Quantity>
    </Line>
    <Line LineNumber="2">
        <Part PartNumber="10002">
            <Description>
                <p><b>How to use XML<b><br />
                Some important skills in writing XML program.</p>
            </Description>
            <Price>73.55</Price>
        </Part>
        <Quantity>5</Quantity>
    </Line>
  </BookOrder>
</Orders>
```

在 XML 编写的文档中，实际上许多都是数据为主的。例如，考虑在 Amazon.com 网站上显示一本书的各种信息的页面，虽然这个页面是一个相当巨大的文本，但是这个文本的结构是高度规范的，所有页面都包含有介绍书的共同点，并且每一页中文本的大小是受限制的。也就是说，一个简单的、数据为主的 XML 文档 + 包含有每一页信息的数据库 + XSL 样板文件，就能够实现这个网站的结构了。通常，目前任何一个动态建立 HTML 的网站都可以由上面介绍的这种结构来实现的。

下面是一个很简单的例子：

```
<Lease>
    <Lessee>ABC Industries</Lessee> agrees to lease the property at
    <Address>123 Main St., Chicago, IL</Address> from <Lessor>XYZ
    Properties</Lessor> for a term of not less than <LeaseTerm
    TimeUnit="Months">18</LeaseTerm> at a cost of <Price
    Currency="USD" TimeUnit="Months">1000</Price>.
</Lease>
```

它是使用下面的这个 XML 文档和一个简单的样板文件实现的：

```
<Lease>
    <Lessee>ABC Industries</Lessee>
    <Address>123 Main St., Chicago, IL</Address>
    <Lessor>XYZ Properties</Lessor>
    <LeaseTerm TimeUnit="Months">18</LeaseTerm>
    <Price Currency="USD" TimeUnit="Months">1000</Price>
</Lease>
```

（2）以文档为主的文件。以文档为主的文件的特点是：不规范的结构，大量的原始数据（就是说，最小的独立数据单元是包含有混合内容的元素级或者本身就是一个文档）和大量混

合内容。其中同类型元素和 PCDATA 出现顺序是非常重要的。例如一首诗歌、一则招聘启事、一封电子邮件，以及几乎所有的 XHTML 文档。以文档为主的文件通常是用来设计人工理解和处理。

例如，下面的产品描述就是一个以文档为主的文件：

```
<Product>
<Name>Turkey Wrench</Name>
<Developer>Full Fabrication Labs, Inc.</Developer>
<Summary>Like a monkey wrench, but not as big.</Summary>
<Description>
<Para>The turkey wrench, which comes in both right- and
left-handed versions (skyhook optional), is made of the finest
stainless steel. The Readi-grip rubberized handle quickly adapts
to your hands, even in the greasiest situations. Adjustment is
possible through a variety of custom dials.</Para>

<Para>You can:</Para>
<List>
<Item><Link URL="Order.html">Order your own turkey wrench</Link></Item>
<Item><Link URL="Wrenches.htm">Read more about wrenches</Link></Item>
<Item><Link URL="catalog.zip">Download the catalog</Link></Item>
</List>

<Para>The turkey wrench costs just $19.99 and, if you
order now, comes with a hand-crafted shrimp hammer as a
bonus gift.</Para>
</Description>
```

事实上，数据为主的文件和文档为主的文件之间的区别并不是很清晰。例如，即使是一个数据为主的文件（例如一张发货单），也有可能包含大量的不规范结构的数据，如发货单的描述部分。而一个以文档为主的文件（例如一本计算机书籍），也可能包含有规范的数据结构（通常是元数据 metadata），如作者名和出版社名称。除了这些，我们用来判断是否是两者中其一的另外一个重要特点是我们是对数据还是对文档感兴趣，这也将决定我们要采用什么样的系统，这方面的选择和处理将在下面讨论。

14.1.2 XML 在数据库中的模式

正如 XML 推出的出发点并不是为了取代 HTML 一样，XML 首先应是一种用于互联网上数据交换的标准。最简单的（也可能是最复杂的）XML 应用是做不同类型系统间的交换格式传送。虽然它们变化无穷，所涉及的概念却是相当简单的。XML 可以用作几乎任何以文本为基础的信息的集装箱，简化了从一个应用程序到另一个应用程序间传递信息的工作，即使应用程序设计者不知道其它的产品，只知道文件格式规则。制成表的数据通常在相关的数据库中传送，可能不适合 XML 的层状结构，XML 可以毫无问题地灵活处理表格。数据库厂商（像 Oracle，Sybase 和 IBM)正在把 XML 解析器和界面加到他们现存的产品中。几乎任何应用程序，不只是数据库，都需要把信息转换成 XML 可以使用的、可普遍理解的层次格式。完成这样的工作需要把信息从一个应用程序中卸下，并装到另外一个应用程序中。这两个应用程序需要包含能

解析 XML 的单元和到它们内部数据结构的相关文件内容，并用内部数据结构，把它们输出到 XML 文件。

通常，XML 在数据库中的应用模型需要借助三层架构来实现（见图 14-1）。在这种模式下，一般会有一个代理程序运行于中间层，通过它来访问数据库管理系统中的数据和输出 XML 文档。实际上，代理程序是一种在客户端桌面应用层与底层数据层之间传递数据的工具。利用 CSS 或 XSL 技术，XML 可以实现基于 Web 浏览器的多样式可视化显示。另外，这种代理程序还应该可以进行双向的基于事件的数据更新，也就是说，客户端的数据变化（如数据的插入、删除、修改等）可以通过代理程序反映到底层数据库，而数据库的更新也能够通知到客户端。表面上看，这种机制同传统的三层架构没有什么区别，但实际上是不同的，因为此时在传输过程中的数据都是已经 XML 化了的。

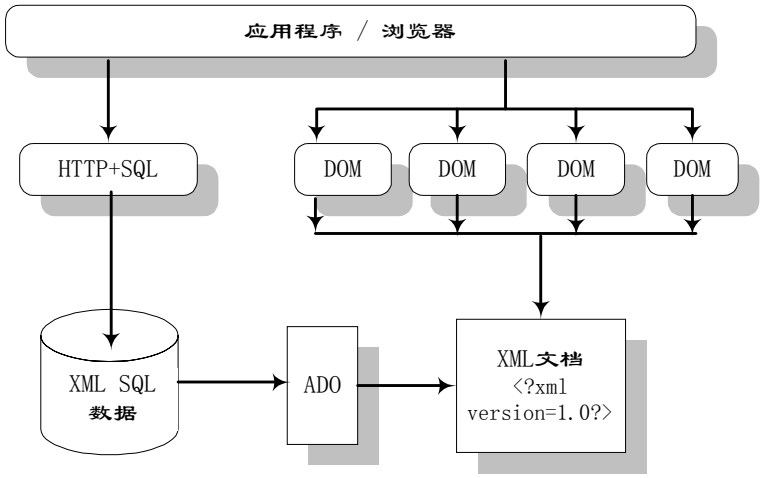


图 14-1　XML 三层应用架构

由图 14-1，可以清楚地看到 XML 的应用架构：后端数据库，针对数据的处理逻辑的中间层服务器，以及数据进一步显示和处理的客户端。数据库可以从多个数据来源接收信息，可能已经是 XML 格式的数据。中间层然后聚拢数据并在最终的表现层上公布。

XML 提供了一种连接关系数据库和面向对象数据库以及其他数据库管理系统之间的纽带。XML 文档本身节点是一种有若干节点组成的属性结构，这种特点使得数据更适宜于用面向对象格式来存储，同时也有利于面向对象语言（C++，Java 等）调用 XML 编程接口访问 XML 节点。关系数据库和面向对象数据库首先需要将数据从数据库中提取出来，经过转换为或直接以 XML 数据形式发布到网上（局域网或 Internet 网），然后相互交换数据，经应用层系统处理后再转存入库。

开发一个访问数据库的 XML 应用系统需要同时借助 XML 编程接口和数据库编程接口，前者用于对 XML 文档的解析、定位和查询，所需技术包括 XML DOM 和 SAX；后者则是用于访问数据库，如数据库中数据的更新和检索等，需要利用的技术有 ODBC，JDBC，ADO 等。

1．存储和获取数据

数据的类型可以在以数据为主的文件的原始定义或数据库中的字段类型中得到。前一种

方法的一个具体例子是：把数据库中的数据保存为 XML 文件放到网站上；后一种方法的一个具体例子是：把大量的数据保存到关系数据库中。根据用户的具体需求，选择采用的软件工具或者是把 XML 数据读入到数据库或者是把数据库中的数据输出到 XML 文件，或者两者都支持。

（1）数据转录。当把数据保存在数据库中时，它经常需要抛弃与文档信息有关的大量内容，例如它的名称和 DTD，同时还有它的物理结构，例如实体的定义和使用、属性值和相同类型元素的出现顺序，还有二进制数据的存储方式（是经过 Base64 编码的还是没有经过解析的实体或者其他方式），CDATA 的内容和其他编码信息。简单而言，当从数据库中获取信息时，最后生成的 XML 文档结果可能不包含任何 CDATA 或者实体引用（entity usage）（除非预先定义了实体 lt（就是符号“<”）,gt(">"), amp（“&”）, apos（“'”）,quot（“"”）)和同类型元素、属性出现的顺序。

举个例子来说，假设我们需要把一个销售单的信息用 XML 格式从一个数据库中获取数据然后转录到另外一个数据库中。这个例子中，XML 文档并不关心销售单的编号是保存在销售单的日期的前面还是后面，也不用关心是否将顾客的名称保存在 CDATA section 作为一个扩展入口，或者直接当成一个 PCDATA。但是，在将这些相关数据从第一个数据库中转录到第二个数据库的过程中，这些信息都是非常重要的。这样，这个数据传输软件就需要考虑使用树状结构（它将一个单独的销售单的信息用组（group）来实现）。

另一个忽略文档信息和它的物理结构会带来麻烦的例子是：“借贷套利”（round-tripping）文档，它保存的数据是从一个数据库中的文档中获取，并且需要重新组装这些数据成为一个新的文档，而这个过程经常会导致新的文档的结构和原来的文档不一致。

从上面的例子可以看出，数据库和数据。输中间件的选择是根据用户的需求而变化的。

（2）将文档结构映射成数据库结构，为了能够在 XML 文档和数据库之间传递数据，有必要将文档的结构映射成数据库的结构，反之亦然，这种映射关系又分两类：模板驱动和模型驱动。

以模板驱动的映射　以模板驱动的映射，这种映射没有预先定义文档结构和数据库结构之间的映射关系，而是使用将命令语句内嵌入模板的方法，让数据传输中间件来执行该模板。例如，考虑下面的模板（注意该模板并不适应于所有的产品），在<SelectStmt>元素中内嵌了 SELECT 选择：

```
<?xml version="1.0"?>
<FlightInfo>
    <Intro>The following flights have available seats:</Intro>
    <SelectStmt>SELECT Airline, FltNumber, Depart, Arrive FROM Flights</SelectStmt>
    <Conclude>We hope one of these meets your needs</Conclude>
</FlightInfo>
```

当数据传输中间件处理到该文档时，每个 SELECT 选项都将被它们各自的结果所替换，得到下面的 XML 格式：

```
<?xml version="1.0"?>
<FlightInfo>
    <Intro>The following flights have available seats:</Intro>
    <Flights>
        <Row>
```

```
            <Airline>ACME</Airline>
            <FltNumber>123</FltNumber>
            <Depart>Dec 12, 1998 13:43</Depart>
            <Arrive>Dec 13, 1998 01:21</Arrive>
        </Row>
        ...
    </Flights>
    <Conclude>We hope one of these meets your needs</Conclude>
</FlightInfo>
```

这种以模板驱动的映射方法相当灵活。例如，一些产品允许在最后的结果中替换用户想要的内容，包括在 SELECT 中使用参数，而不是像上面的例子中简单地格式化结果。

另外它还支持使用编程结构例如循环和条件判断结构。还有就是它支持通过 HTTP 的传递参数。目前，以模板驱动的映射仅仅只支持从一个关系数据库转换成 XML 文档的情况。

以模型驱动的映射 “以模型驱动的映射模式”的原理就是利用 XML 文档中的数据模型的结构显性或隐性地将其映射成数据库的结构，反之亦然。它的缺点是灵活性不如模板驱动方式，但其优点是简单易用，这是因为它是基于具体的数据模型来进行映射的，通常它能够自己完成很多的转换工作，因此简单易用。由于将数据从数据库转换成 XML 的工作是根据一个单一的模型，所以通常在这种方式下还要综合搭配 XSL 来提供灵活性。

在 XML 文档中有两种模型是非常普遍的。第一种是被许多中间件包在转换 XML 文档成关系数据库数据所使用到的模型，就是将 XML 文档当成一个单独的表对象或者一系列表对象。也就是说，真正的 XML 文档必须类似于下面的格式，如果是单一的表对象的话，<database>元素就不需要出现了：

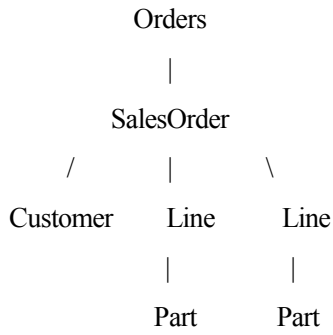
```
<database>
    <table>
        <row>
            <column1>...</column1>
            <column2>...</column2>
            ...
        </row>
        ...
    </table>
    ...
</database>
```

其中的“table”可理解为单个的结果集（当数据是从数据库往 XML 中传输时)或者是一个单独的表对象或者一个可更新的视图（当数据是从 XML 往数据库传输时）。如果数据是来自多个结果集的描述（当数据来自数据库中时）或者 XML 的文档包含有更深层次的嵌套元素，并且有必要表现成一系列表对象（当数据要转换到数据库中时)，那么类似于上面例子那么简单的传输是不可能的。

第二种普遍的数据模型是 XML 文档中的对象树，在这种模型下，元素通常对应了一个对象、属性或者 PCDATA 对象。这种模型直接映射成面向对象的数据库和树状结构数据库，当然借助传统的对象-关系映射技术和 SQL 3 对象视图也可以映射成关系数据库。需要注意的是，这种模型并不是文档对象模型（DOM)，DOM 是指文档本身是模型，而不是指文档中的数据。

一个具体的实例就是上面介绍的销售单文档可以被看成是有 5 个类的对象树：

Orders, SalesOrder, Customer, Line, and Part，关系如下图所示：



当一个 XML 文档模型化处理成一个对象树时，对元素和对象没有什么特殊的要求。例如，如果一个元素只包含有 PCDATA，例如销售单文档中的 CustName 元素，它能够被看做一个属性进行处理（就是仅仅只包含有单独的数值）。简单地说，有时将混合元素或者元素内容模型化处理成属性是非常有用的方法。一个现成的例子就是在销售单文档中对 Description 元素的处理：尽管它在 XHTML 的 Form 中有混合的内容，但是将 description 元素看作一个单独的属性来处理会更有用些，因为它的组成部分就本身而言并没有什么意义。

数据类型、空值（Null）、字符集设置和其他所有的类似集 这里我们将探讨一些在 XML 文档转换成数据库过程中有关存储数据的内容。通常，我们在选择什么样的中间件来解决这些问题的时候是不会考虑到这些问题的，但是如果注意到这些问题的存在时，希望我们的讨论对读者在选择中间件时有所帮助。

- 数据类型，XML 不支持任何有意义的数据类型，除非是不能够解释的实体，所有 XML 文档中的数据都被当成文本来对待，虽然它能够用其他的数据类型来表示，例如可以表示成日期或者整数。通常，数据转换中间件将把文本（在 XML 文档中的文本）转换成其它的数据类型（数据库中的数据类型），反之亦然。然而一些特定的数据类型在转换的过程中是受限制的，例如会受那些提供数据支持的 JDBC 驱动的限制。在这些众多的有可能的数据类型中，日期类型通常会导致麻烦。数字，特别是由于国际地域不同的数字格式，也可能导致问题。
- 二进制数据，有两种比较普遍的方法将二进制数据保存到 XML 文档中：对实体不做任何编码处理和对实体进行 Base64 编码处理（Base64 是一种 MIME 编码方法，可以将二进制数据影射成 US-ASCII 的子集)。对于关系数据库，这两种方法都被证实有可能存在问题，因为大家都知道当保存和获取二进制数据到数据库中的规则是非常严格的，这样中间件将有可能导致问题。另外，并没有一种标准的符号用来说明一个 XML 文档中的元素包含有 Base64 编码数据，从而中间件可能根本就不能够识别这种编码。最后，还有可能有些中间件在将数据存储入数据库的过程中根本就会忽略没有编码实体中的符号或者 Base64 编码中的元素。所以，如果对用户而言，二进制数据非常重要的话，那么一定要确认中间件是否支持二进制数据。
- 空值（Null），在数据库世界中，null 数据意味着数据为空。这不同于一个值为 0（对数字类型数据）或者长度为 0（对字符串类型）。例如，假设我们的数据是收集自一

个气象站，如果气象站的温度计出毛病了，那么数据库中将存储一个 null 值而不是一个 0，值为 0 完全是另外一回事了。

XML 也支持空值的概念，可以通过设置元素的类型和属性来实现。如果元素类型或属性的值为 null，XML 的处理方法是简单地不将其包含到文档中。但是对数据库来说，空的元素或者包含 0 长度字符串的属性并不意味着 null：它们的值是长度为 0 的字符串。

当将一个 XML 文档结构映射成数据库或者是反过程中，用户必须特别注意那些可选的数据类型和本来表示空值的属性。如果不这么做的话，结果将是可能出现插入错误（当将数据转换到数据库中时)或者非法文档错误（当将数据从数据库读出时）。

因为 XML 中相对与数据库而言在对符号意义的声明有更好的灵活性，具体来说，就是 XML 用户愿意将空元素或包含长度为 0 内容的属性认为是“null”。我们必须根据这个考虑选择什么样的中间件来处理这个问题。一些中间件可以让用户自定义在 XML 文档中用什么标志来表示“null”。

- 字符集设置，根据定义，一个 XML 文档能够包含任何 Unicode 字符，除了一些特殊的控制字符。但是不幸的是，许多数据库都限制或者不支持 Unicode 并且需要一些特殊的配置才能够处理非 ASCII 编码的数据字符。如果用户的数据包含有非 ASCII 字符，那么一定要确保数据库和中间件是否能够处理这些字符集。
- 处理指令（Processing Instructions），处理指令不是 XML 文档中的“数据”部分，目前许多中间件都不能够正常地处理它们。问题是这样的，尤其是在一个严格的将 XML 文档结构映射成数据库结构中，处理指令通常是很难处理的，因为题目可以出现在文档的任何位置，于是，中间件就非常困难的需要判断将它们保存到什么位置和读取的时候取回到什么位置。如果处理指令和文挡的“round-tripping”对用户而言是非常重要的话，就必须确保所选择的中间件能够处理这个问题。
- 存储标志（Storing Markup），有时候直接将包含有元素或者混合内容的元素不进行进一步的解析直接保存到数据库中是非常有用的。最普遍的实现方法是简单地把这个标志本身直接保存到数据库中。不幸的是，这将带来另外一个问题：当从数据库中读取这些数据时，将很难判断数据库中的标志到底是真是假，特别是一些由 lt 和 gt 转义的字符。

  例如下面的描述：

  ```
  <description>
     <b>Confusing example:</b> &lt;foo/&gt;
  </description>
  ```

  保存到数据库中将变成这样：

  ```
  <b>Confusing example:</b> <foo/>
  ```

  这时数据库将不能够判断<b>和<foo>是标志还是文本了。解决方法有以下几种，例如将标志的符号使用其它非标志符号替代，但是这时要非常小心，因为也许别的运用程序在使用这些数据时就会出现不兼容的现象。例如，如果想查询数据库中的小于号（“<”）和 lt 标志（“<”）时就要特别小心了。
- 从数据库的结构生成 DTD 和逆反过程。在 XML 文档和数据库之间转换数据的一个

普遍问题是：如何从数据库的结构生成 DTD 和其逆反过程。简而言之，目前有许多软件都提供了可以直接使用的操作功能，但是它产生的结果对许多用户来说用处和帮助不大，也许没有多少人喜欢。

例如，下面的过程（已经简化过的）就是从一个 XML 文档到关系数据库中生成 DTD 的：

① 对每一种包含有元素或者混合内容的元素类型，新建一个 table 和一个主关键字段。

② 对混合内容中的每一个元素，建立一个分开的 table，在其中保存 PCDATA，通过主关键字连接到父表中。

③ 对于元素类型中每个有单一数值的属性和只包含有 PCDATA 内容的子元素类型在该 table 中新建立一列（字段）。如果子元素类型或者属性是可选的，允许该字段为空。

④ 对于每个有多值的属性或者仅含有 PCDATA 内容的子元素类型，再建立一个分开的 table 来保存它们的值，通过它们的父表的主关键字连接到父表。

⑤ 对于每个子元素，这些子元素本身还有元素或者混合内容，使用父表中的关键字将父元素表连接到子元素表中。

下面则是一个简化过的从关系数据库的结构生成 XML 文档的过程：

① 1 对每个 table，新建一个元素。

② 对表中的每列，建立一个属性或者只含 PCDATA 的子元素。

③ 对每个包含有在主键/外键关键字关系中主键值的列，新建一个子元素。

不幸的是，在这些过程中存在许多缺陷。例如，其中没有实现对数据类型的预先定义，在 DTD 中没有实现对列长度的预先定义的方法。因为任何的预先定义，例如通过读一个示例文档，当读取其他“类型”的文档或者其他文档中包含有超过字段长度内容的文档时，就会发生错误（解决这个问题的办法是使用 XML schema 文档中数据类型定义)。简单来说，当从一个关系结构中生成 DTD 时，是没有办法预先判断子元素“应该”出现的顺序或者类似数据库中的行标识。

尽管有这些缺陷，但是根据数据库结构生成 DTD 的软件能够给我们带来了一个很好的开端，特别是对于那些非常庞大和复杂的系统。

2. 存储和获取文件（Documents）

文件格式可以是 XML，当然也可以是其它格式，例如 RTF，PDF 或者 SGML，然后由它们转换成 XML。于是，如果用户使用 XML 文件，就需要一种存储文件和从 XML 文档中获取文件的方法，就象当初把其它格式的文件转换成 XML 文件格式一样。本小节只讨论这种情况，至于有关转换软件工具的信息，可以参考 14.2.3 中的相关列表。

对于简单的文件集，文件系统或者一些版本控制系统（例如运用在软件版本控制中的系统）也许就能够满足需求了。但是，如果用户有一个复杂的文件集，就需要一个内容管理系统。内容管理系统是相对于文件管理系统的，它主要表现在这种系统通常允许用户把文件分割成不连续的内容块，例如实例，程序，章节和其他选项：元数据 metadata（作者名称，修订日期，文件数目）；而不是把将每个文件统一成一个整体进行管理。这种系统不仅简化了如何协调多个合作者同时对同一个文件进行工作之类的问题，它还允许用户基于当前已经存在的文件而合

成一个新的文件。

与 XML 文档数据保存在数据库中的情况不同，内容管理系统通常还支持“round-tripping”类型的文件，如一个需要经常维护文件物理结构的情况。内容管理系统也提供其他的一些功能，包括：

- 版本访问和控制
- 搜索引擎
- 编辑器
- 发布引擎，例如发布到论文、CD 或者 Web
- 分离的内容和样式
- 可通过脚本或者编程扩展
- 与数据库数据的综合

3. 内容管理系统和关系数据库

因为关系数据库系统的广泛使用，所以这个字眼“将 XML 文件保存到数据库中”很多人都等同于内容管理系统，许多人都认为将 XML 文件（类似与这些 XML 文档中的数据）保存在关系数据库中是个很好的方法。事实上在很多情况下，这样简化是不正确的。

最主要的原因是关系数据库在许多本来应该是内容管理系统固有的功能上的表现并不令人满意：例如排序、分层、不规范结构、可变长字段。举例来说，如果我们想把 PCDATA 和子元素中在父元素中出现的顺序信息存储下来的话，我们就自己来把子元素按顺序存储在分开的列中。或者我们可以使用 SQL 让下面的查询实现起来简单些：“get me all chapters in which the third paragraph mentions part 123 in bold”。所以，很多内容管理系统都是建立在面向对象或者树状结构数据库基础之上的。

这并不是说关系数据库就不能够被用来存储 XML 文档了，这只是意味这我们将受到一些限制和很多工作需要自己来完成。

最简单的将 XML 文件保存在一个关系数据库中的方法就是将每一个文档不进行任何解释直接存储到列中。尽管这个方法有明显的缺陷，例如不能够从已经存在的文件来建立新的文档的缺点，但是在某些特定情况下它却非常简单而实用，例如当我们需要处理的是使用 XHTML 编写的内容时，就没有必要将其分割成小的部分。另外，关系数据库有能力处理有规律逐渐增加的文本内容，并且能够进行全文索引，还能够执行专门的搜索功能，例如模糊查询或者使用同义字查询等。这样，直接不解释地将文件存储在单独的一列中也许可以满足简单的文档管理。

一个更复杂的例子是编写一个位于关系数据库顶层的内容管理系统。一个这样的系统在 Mark Birbeck 的相关 XML-L（http://listserv.heanet.ie/lists/xml-l.html）邮件列表中描述过。该系统由 5 个表组成：

- 属性定义：定义属性，包括它们的类型，合法值等。
- 元素/属性关联：定义何种属性对应与何种元素。
- 内容模型定义：定义何种元素能够包含何种其他元素。
- 属性值：包含属性数值和指出在属性定义何元素/属性关联表中的适当行所在位置。
- 元素值：包含元素数值（PCDATA 或者指向其他元素值的指针），该元素在父元素中出现的位置，一个指向父元素中值的指针和一个指向元素/属性关联表中适当位置的指

针。

前三种简单的表相当于一个简单的 DTD；后面的两个表包含实际的数据。通过重复查询后两个表，就能够重新构建一个 XML 文档的任何一个部分。对于更加详细的细节，可以参考“Record ends, Mixed content, and storing XML documents on relational database”（该文档位于 http://listserv.heanet.ie/cgi-bin/wa?A1=ind9812&L=xml-l#26）以及“storing XML documents on relational database”（该文档位于 http://listserv.heanet.ie/cgi-bin/wa?A1=ind9812&L=xml-l#55）。

如果想获取其他关于将 XML 或树状结构信息保存在关系数据库中的信息，可以参考文献“XML documents in relational databases”，该文档位于 http://listserv.heanet.ie/cgi-XML。

14.1.3 在 XML 中对关系数据建模

在这一节中，我们将讨论 XML 中的数据建模问题。

1. 建模方法

在将传统数据映射到 XML 文档的过程中，并不只存在唯一的解决方案。一种常用的方法是把数据库中的个体信息分别转换到预先规定格式的 schema 中；另一种方法是将数据库的全部内容（包括数据、关系、完整性等）进行整体转换。这两种方法对于数据建模都是可行和实用的，但都有无法避免的缺点：需要实质性的自定义码，事务处理的概念模糊不清。下面我们提出一种介于两者之间的基于关系数据的解决方法，这种方法的优点在于可以生成易于理解的符合现存的数据结构和 XML 概念（如：ID/IDREF）的表格（schemas）。

我们先看一个具体实例。下面是一个有三个表的存储职员信息的数据库。EMPLOYEE 存储个人信息和各种关系的概要；PERF_REVIEW 存储职员的业绩和评价；COMP_CHANGE 存储业绩评价的更新信息。

```
TABLE EMPLOYEE
NUM LONGINT PRIMARY KEY

FNAME STRING 32

LNAME STRING 32

HIRE_DATE DATE

TERM_DATE DATE MAY BE NULL

TABLE PERF_REVIEW
EMP_NUM LONGINT PRIMARY KEY FOREIGN KEY

REVIEW_DATE DATE PRIMARY KEY

REVIEW TEXT

TABLE COMP_CHANGE

EMP_NUM LONGINT FOREIGN KEY
```

```
REVIEW_DATE DATE MAY BE NULL

EFF_DATE DATE

SALARY INT
```

从上面所列数据库中提取信息而得到的一个简单 XML 文档如下：

```
<EMPLOYEE
NUM = '2361'
FNAME = 'Wilbert'
LNAME = 'Winston'
HIRE_DATE = '4/4/88'>
</EMPLOYEE>
```

可以看到，我们生成的 XML 文档中的元素名与数据库中表名是一致的，而且表中的字段值存放在 XML 文档元素的属性值中。由此，我们可以对数据建模作一些较大的改进：

- 可以获取数据类型信息。
- 可以保持键值关系。
- 灵活应用 XML 的 ID/IDREF。

要实现这些改进，我们需要用到下面将要介绍的方法来获取一些相关信息。

2. 数据类型模型化

传统的数据源需要有很严格的数据类型定义，而 XML 文档通过定义将信息存储在文本表格中，保持原有的数据类型是必要的。W3C 在已提交的各种 Schema 基础上汇编了一个数据类型集，我们将采用这些数据类型名称约定。

string	number	dateTime
boolean	float	date
uri	int	time

取定名称之后，存储空间问题又出现了，例如：需要多少字节来存储“int”类型的数据？我们用元数据属性 dsize 来获取这些信息，如下所示：

dtype	dsize	Example
string	maximum number of characters	23
float	bytes (4 or 8)	4
number	x.y where x = digits allowed to left of decimal y = digits allowed to right of decimal	14.4
int	bytes (1, 2, 4 or 8)*	8

3. 数据关系模型化

数据库数据映射到 XML 文档的复杂性主要在于映射数据关系，这类似于关系数据库中 primary-foreign key（主键-外键）的设置。

（1）Primary Keys（主键）。熟悉关系数据库的人都知道，主键是数据库中特定的表被访问时唯一标识数据的字段。XML 也有 ID 来唯一标识元素并进行访问：

```
<EMPLOYEE pkey_id = "EMPLOYEE.2361">

<EMPLOYEE.NUM>2361</EMPLOYEE.NUM>

<EMPLOYEE.FNAME>Wilbert</EMPLOYEE.FNAME>

<EMPLOYEE.LNAME>Winston</EMPLOYEE.LNAME>

<EMPLOYEE.HIRE_DATE>4/4/88</EMPLOYEE.HIRE_DATE>

</EMPLOYEE>
```

（2）Foreign Keys（外键）。外键是通过设定关系连接不同数据表的字段，XML 中有 ID 提供与 Primary Keys 相类似的功能，因此，IDREF 就相应地提供与 Foreign Keys 相类似的功能。

```
<EMPLOYEE pkey_id = "EMPLOYEE.2361">

<EMPLOYEE.NUM>2361</EMPLOYEE.NUM>
...

<PERF_REVIEW PERF_REVIEW.EMP_NUM_idref = "EMPLOYEE.2361">

<PERF_REVIEW.EMP_NUM>2361</PERF_REVIEW.EMP_NUM>
...
```

（3）Nesting（嵌套）。在某些情况下，实现上述机制还有另一种方法——嵌套。这种方法利用了 XML 中元素关系可从文本和 id/idref 中获取的特性。下例中，我们将 PERF_REVIEW 元素放进 EMPLOYEE 元素中与之相关联。

```
<EMPLOYEE EMPLOYEE.NUM_id = "EMPLOYEE.2361"
NUM = '2361'
FNAME = 'Wilbert'
LNAME = 'Winston'
HIRE_DATE = '4/4/88'>

<PERF_REVIEW
REVIEW_DATE = '1/1/98'
REVIEW = 'lousy'/>

<PERF_REVIEW
REVIEW_DATE = '1/1/99'
REVIEW = 'worse'/>
</EMPLOYEE>
```

14.1.4 小结

采用一些标准约定，XML 可以对诸如关系数据库一类的资源和信息进行建模。通过获取数据类型、键值关系和 id/idref 信息，可以析取出简化信息交换过程的元数据文档。我们上面讨论的方法简单可行，并已应用于现有的一些 XML 成熟产品中，对各种结构的数据库进行处理。下面是综合而成的几个实例。

例 14. 1　DTD with Columns as Elements

```
<!ELEMENT EMPLOYEE (
EMPLOYEE.NUM? ,
EMPLOYEE.FNAME ,
EMPLOYEE.LNAME ,
EMPLOYEE.HIRE_DATE ,
EMPLOYEE.TERM_DATE? )>

<!ATTLIST EMPLOYEE pkey_id ID #REQUIRED
e-pkey NMTOKEN #FIXED 'EMPLOYEE.NUM' >

<!ELEMENT EMPLOYEE.NUM (#PCDATA )>

<!ATTLIST EMPLOYEE.NUM e-dtype NMTOKEN #FIXED 'int' >

<!ELEMENT EMPLOYEE.FNAME (#PCDATA )>

<!ATTLIST EMPLOYEE.FNAME e-dtype NMTOKEN #FIXED 'string'
e-dSize NMTOKEN #FIXED '32' >

<!ELEMENT EMPLOYEE.LNAME (#PCDATA )>

<!ATTLIST EMPLOYEE.LNAME e-dtype NMTOKEN #FIXED 'string'
e-dSize NMTOKEN #FIXED '32' >

<!ELEMENT EMPLOYEE.HIRE_DATE (#PCDATA )>

<!ATTLIST EMPLOYEE.HIRE_DATE e-dtype NMTOKEN #FIXED 'date' >

<!ELEMENT EMPLOYEE.TERM_DATE (#PCDATA )>

<!ATTLIST EMPLOYEE.TERM_DATE e-dtype NMTOKEN #FIXED 'date' >


<!ELEMENT PERF_REVIEW (
PERF_REVIEW.EMP_NUM ,

PERF_REVIEW.REVIEW_DATE ,

PERF_REVIEW.REVIEW )>

<!ATTLIST PERF_REVIEW PERF_REVIEW.EMP_NUM_idref IDREF #REQUIRED >

<!ELEMENT PERF_REVIEW.EMP_NUM (#PCDATA )>

<!ATTLIST PERF_REVIEW.EMP_NUM e-dtype NMTOKEN #FIXED 'int'
e-fkey NMTOKEN #FIXED 'EMPLOYEE.NUM' >
```

<!ELEMENT PERF_REVIEW.REVIEW_DATE (#PCDATA)>

<!ATTLIST PERF_REVIEW.REVIEW_DATE e-dtype NMTOKEN #FIXED 'date' >

<!ELEMENT PERF_REVIEW.REVIEW (#PCDATA)>

<!ATTLIST PERF_REVIEW.REVIEW e-dtype NMTOKEN #FIXED 'string'
e-dSize NMTOKEN #FIXED '50' >

<!ELEMENT COMP_CHANGE (
COMP_CHANGE.EMP_NUM ,

COMP_CHANGE.REVIEW_DATE? ,

COMP_CHANGE.EFF_DATE ,

COMP_CHANGE.SALARY)>

<!ATTLIST COMP_CHANGE COMP_CHANGE.EMP_NUM_idref IDREF #REQUIRED >

<!ELEMENT COMP_CHANGE.EMP_NUM (#PCDATA)>

<!ATTLIST COMP_CHANGE.EMP_NUM e-dtype NMTOKEN #FIXED 'int'

e-fkey NMTOKEN #FIXED 'EMPLOYEE.NUM' >

<!ELEMENT COMP_CHANGE.REVIEW_DATE (#PCDATA)><!ATTLIST COMP_CHANGE.REVIEW_DATE e-dtype NMTOKEN #FIXED 'date' >

<!ELEMENT COMP_CHANGE.EFF_DATE (#PCDATA)><!ATTLIST COMP_CHANGE.EFF_DATE e-dtype NMTOKEN #FIXED 'date' >

<!ELEMENT COMP_CHANGE.SALARY (#PCDATA)><!ATTLIST COMP_CHANGE.SALARY e-dtype NMTOKEN #FIXED 'int' >

例 14. 2　DTD with Columns as Attributes

<!ELEMENT EMPLOYEE EMPTY>

<!ATTLIST EMPLOYEE pkey_id ID #REQUIRED
NUM CDATA #REQUIRED

FNAME CDATA #REQUIRED

LNAME CDATA #REQUIRED

HIRE_DATE CDATA #REQUIRED

TERM_DATE CDATA #IMPLIED

e-pkey NMTOKEN #FIXED 'NUM'

a-dtype NMTOKENS 'NUM int

FNAME string

LNAME string

HIRE_DATE date

TERM_DATE date'

a-dSize NMTOKENS 'FNAME 32 LNAME 32' >

<!ELEMENT PERF_REVIEW EMPTY>

<!ATTLIST PERF_REVIEW PERF_REVIEW.EMP_NUM_idref IDREF #REQUIRED

EMP_NUM CDATA #REQUIRED

REVIEW_DATE CDATA #REQUIRED

REVIEW CDATA #REQUIRED

a-dtype NMTOKENS 'EMP_NUM int

REVIEW_DATE date

REVIEW date'

a-fkey NMTOKENS 'EMP_NUM EMPLOYEE.NUM'

a-dSize NMTOKENS 'REVIEW 50' >

<!ELEMENT COMP_CHANGE EMPTY>

<!ATTLIST COMP_CHANGE COMP_CHANGE.EMP_NUM_idref IDREF #REQUIRED

EMP_NUM CDATA #REQUIRED

REVIEW_DATE CDATA #IMPLIED

EFF_DATE CDATA #REQUIRED

SALARY CDATA #REQUIRED

a-dtype NMTOKENS 'EMP_NUM int

REVIEW_DATE date

EFF_DATE date

SALARY int'

a-fkey NMTOKENS 'EMP_NUM EMPLOYEE.NUM' >

例 14. 3　XML Data with Columns as Elements

```
<?xml version ="1.0"?>

<!--Generated by XML Authority. Conforms to XML Data subset for IE 5-->

<Schema name = ""

xmlns = "urn:schemas-microsoft-com:xml-data"

xmlns:dt = "urn:schemas-microsoft-com:datatypes"

xmlns:xa = "www.extensibility.com/schemas/xdr/metaprops.xdr">

<ElementType name = "EMPLOYEE" xa:pkey = "EMPLOYEE.NUM" content = "eltOnly" order = "seq">

<AttributeType name = "pkey_id" dt:type = "ID" required = "yes"/>

<attribute type = "pkey_id"/>

<element type = "EMPLOYEE.NUM" />

<element type = "EMPLOYEE.FNAME" />

<element type = "EMPLOYEE.LNAME" />
<element type = "EMPLOYEE.HIRE_DATE />

<element type = "EMPLOYEE.TERM_DATE" minOccurs = "0" maxOccurs = "1"/>

</ElementType>

<ElementType name = "EMPLOYEE.NUM" content = "textOnly" dt:type = "i4"/>
```

<ElementType name = "EMPLOYEE.FNAME" content = "textOnly" dt:type = "string"/>

<ElementType name = "EMPLOYEE.LNAME" content = "textOnly" dt:type = "string"/>

<ElementType name = "EMPLOYEE.HIRE_DATE" content = "textOnly" dt:type = "date"/>

<ElementType name = "EMPLOYEE.TERM_DATE" content = "textOnly" dt:type = "date"/>

<ElementType name = "PERF_REVIEW" content = "eltOnly" order = "seq">

<AttributeType name = "PERF_REVIEW.EMP_NUM_idref" dt:type = "IDREF" required = "yes"/>

<attribute type = "PERF_REVIEW.EMP_NUM_idref"/>

<element type = "PERF_REVIEW.EMP_NUM" />

<element type = "PERF_REVIEW.REVIEW_DATE />

<element type = "PERF_REVIEW.REVIEW" />

</ElementType>

<ElementType name = "PERF_REVIEW.EMP_NUM" xa:fkey = "EMPLOYEE.NUM" content = "textOnly" dt:type = "i4"/>

<ElementType name = "PERF_REVIEW.REVIEW_DATE" content = "textOnly" dt:type = "date"/>

<ElementType name = "PERF_REVIEW.REVIEW" content = "textOnly" dt:type = "string"/>

<ElementType name = "COMP_CHANGE" content = "eltOnly" order = "seq">
<AttributeType name = "COMP_CHANGE.EMP_NUM_idref" dt:type = "IDREF" required = "yes"/>

<attribute type = "COMP_CHANGE.EMP_NUM_idref"/>

<element type = "COMP_CHANGE.EMP_NUM" />

<element type = "COMP_CHANGE.REVIEW_DATE" minOccurs = "0" maxOccurs = "1" />

<element type = "COMP_CHANGE.EFF_DATE" />

<element type = "COMP_CHANGE.SALARY" />

</ElementType>

<ElementType name = "COMP_CHANGE.EMP_NUM" xa:fkey = "EMPLOYEE.NUM" content = "textOnly" dt:type = "i4"/>

<ElementType name = "COMP_CHANGE.REVIEW_DATE" content = "textOnly" dt:type = "date"/>

<ElementType name = "COMP_CHANGE.EFF_DATE" content = "textOnly" dt:type = "date"/>

<ElementType name = "COMP_CHANGE.SALARY" content = "textOnly" dt:type = "i2"/>

</Schema>

14.1.5 XML 与元数据的制定

Metadata（元数据）是“关于数据的数据”。元数据为各种形态的数字化信息单元和资源集合提供规范、普遍的描述方法和检索工具；元数据为分布的、由多种数字化资源有机构成的信息体系（如海量资源库、多媒体信息中心等）提供整合的工具与纽带；同时元数据与数据仓库的结合也非常紧密，具体技术请参阅数据仓库相关资料和书籍。

Metadata 的应用目的：

（1）确认和检索（Discovery and Identification），主要致力于如何帮助人们检索和确认所需要的资源，数据元素往往限于作者、标题、主题、位置等简单信息，Dublin Core 是其典型代表。

（2）著录描述（Cataloging），用于对数据单元进行详细、全面的著录描述，数据元素囊括内容、载体、位置与获取方式、制作与利用方法、甚至相关数据单元方面等，数据元素数量往往较多，MARC、GILS 和 FGDC/CSDGM 是这类 Metadata 的典型代表。

（3）资源管理（Resource Administration），支持资源的存储和使用管理，数据元素除比较全面地注重描述信息外，还往往包括权利管理（Rights/Privacy Management）、电子签名（Digital Signature）、资源评鉴（Seal of Approval/Rating）、使用管理（Access Management）、支付审计（Payment and Accounting）等方面的信息。

（4）资源保护与长期保存（Preservation and Archiving），支持对资源进行长期保存，数据元素除对资源进行描述和确认外，往往包括详细的格式信息、制作信息、保护条件、转换方式（Migration Methods）、保存责任等内容。

根据不同领域的数据特点和应用需要，90 年代以来，许多 Metadata 格式在各个不同领域出现，例如：

- 网络资源：Dublin Core，IAFA Template，CDF，Web Collections
- 文献资料：MARC（with 856 Field），Dublic Core
- 人文科学：TEI Header
- 社会科学数据集：ICPSR SGML Codebook
- 博物馆与艺术作品：CIMI，CDWA，RLG REACH Element Set，VRA Core
- 政府信息：GILS
- 地理空间信息：FGDC/CSDGM
- 数字图像：MOA2 metadata，CDL metadata，Open Archives Format，VRA Core，NISO/CLIR/RLG
- Technical Metadata for Images
- 档案库与资源集合：EAD
- 技术报告：RFC 1807
- 连续图像：MPEG-7

在元数据发展初期人们常使用自定义的记录语言（例如 MARC）或数据库记录结构（如 ROADS 等），但随着元数据格式的增多和互操作的要求，人们开始采用一些标准化的 DDL 来描述元数据，例如 SGML 和 XML，其中以 XML 最有潜力。

下面是按国际组织 Dublin Core Metadata Initiative 拟定的用于标识电子资源的一种简要目录模式--Dublin Core 的元数据标准而设计的标记计算机书籍信息元数据的文档类型定义：

Pcbook.dtd

```
<!-- pcbook info  -->
<!ELEMENT dlib (pcbook+)>
<!ELEMENT  pcbook  (dc_title+,  dc_creator,  dc_subject,  dc_description,  dc_publisher,
                    dc_contributor, dc_date, dc_type, dc_format, dc_identifier, dc_source,
                    dc_language, dc_relation,dc_coverage, dc_righits)>
<!ELEMENT dc_title (#PCDATA)>
<!ELEMENT dc_creator (#PCDATA)>
<!ELEMENT dc_subject (#PCDATA)>
<!ELEMENT dc_description (#PCDATA)>
<!ELEMENT dc_publisher (#PCDATA)>
<!ELEMENT dc_contributor (#PCDATA)>
<!ELEMENT dc_date (#PCDATA)>
<!ELEMENT dc_type (#PCDATA)>
<!ELEMENT dc_format (#PCDATA)>
<!ELEMENT dc_identifier (#PCDATA)>
<!ELEMENT dc_source (#PCDATA)>
<!ELEMENT dc_language (#PCDATA)>
<!ELEMENT dc_relation (#PCDATA)>
<!ELEMENT dc_coverage (#PCDATA)>
<!ELEMENT dc_righits (#PCDATA)>
```

附：Dublin Core 的十五项广义元数据（Metadata）

（1）名称（Title）

标识：Title

定义：分配给资源的名称。

解释：使资源为众所周知的有代表性的正规名称。

（2）创作、制作者（Creator）

标识：Creator

定义：制作资源内容的主要责任实体。

解释：创作、制作者包括个人、组织或机构。 应该是用于标识创作、制作者实体的具有代表性的名称。

（3）主题及关键词（Subject and Keywords）

标识：Subject

定义：资源内容的主题。

解释：用以描述资源主要内容的关键词语或分类号码表示的有代表性的主题词。

（4）说明（Description）

标识：Description

定义：有关资源内容的说明。

解释：该说明可以包括但并不限于：摘要，内容目次，内容图示或内容的文字说明。

（5）出版者（Publisher）

标识：Publisher

定义：制作资源有重要作用的责任实体。

解释：如包括个人、组织或机构的出版者。应是用于标识出版者实体的有代表性的名称。

（6）发行者（Contributor）

标识：Contributor

定义：对资源内容负有发行责任的实体。

解释：发行者包括个人、组织或机构。应是用于标识发行者实体的有代表性的名称。

（7）时间（Date）

标识：Date

定义：与资源使用期限相关的日期、时间。

解释：资源产生或有效使用的日期、时间。推荐使用 ISO 8601[W3CDFT]定义的编码形式，跟随的是 YYYY-MM-DD 形式。

（8）类型（Type）

标识：Type

定义：资源内容方面的特征或体裁。

解释：类型包括种类、功能、体裁或作品集成级别等描述性术语。推荐从可控词表（如 Dublin Core Types[DCT1]）中选用有关术语。对于资源物理或数字化方面表示，采用“格式”项描述。

（9）格式（Format）

标识：Format

定义：资源物理或数字化的特有表示。

解释：格式可包括媒体类型或资源容量。也可用于限定资源显示或操作所需的软件、硬件或其它设备，如容量包括数据所占空间和存在期间。

（10）标识（Identifier）

标识：Identifier

定义：依据有关规定分配给资源的标识性信息。

解释：推荐使用依据格式化标识系统规定的字符或号码标识资源。如正规标识系统包括统一资源标识（URI），统一资源地址（URL）、数字对象标识（DOI）以及国际标准书号（ISBN）、国际标准刊号（ISSN）等。

（11）来源（Source）

标识：Source

定义：可获取现存资源的有关信息。

解释：可从原资源整体或部分获得现有资源。建议使用正规标识系统确定的字符或号码标引资源来源信息。

（12）语言（Language）

标识：Language

定义：资源知识内容使用的语种。

解释：推荐使用由 RFC1766 定义的语种代码，它由两位字符（源自 ISO639）组成。随后可选用两字符的国家代码（源自 ISO 3166）。如“en”表示英语，“fr”表示法语。

（13）相关资源（Relation）

标识：Relation

定义：对相关资源的参照。

解释：推荐用依据正规标识系统确定的字符或号码标引资源参照信息。

（14）范围（Coverage）

标识：Coverage

定义：资源内容的领域或范围。

解释：范围包括空间定位（地名或地理座标），时代（年代、日期或日期范围）或权限范围。

（15）版权（Righs）

标识：Rights

定义：持有或拥有该资源权力的信息。

解释：版权项包括资源版权管理的说明。版权信息通常包含智力知识内容所有权（IPR）、著作权和各种拥有权。如果缺少版权项，就意味着不考虑有关资源的上述版权和其它权力。

14.2 来自业界的支持

14.2.1 Microsoft SQL Server 的宣言

在 1996 年秋天以前，Microsoft 中的很多小组，包括 Office 小组、电子商务站点服务器小组、数据访问小组等，都在寻求一种具有网上互操作性的开放文档格式。于是便出现了后面我们将要重点介绍的“频道定义格式”（Channel Definition Format，CDF）。CDF 是 XML 在网上的第一个主要应用，它取得了巨大成功。在随后的一段时间里，Microsoft 内部的每个小组都陆续使用 XML 编写应用程序，因为 XML 表示数据的简单语法正好满足了他们的需要，这股技术风潮相当猛烈。到 1997 年 10 月，Bill Gates 宣布 XML 是“一个突破性的技术”，就预示了 XML 的光明前路。

Microsoft 于 2000 年 1 月宣布其 SQL Server 对 XML 提供支持，并且发布了一个预览版本，它意味着 Microsoft 在其战略决策上将 XML 技术放在一个何等重要的地位，而这一消息的发布无疑将给广大的 XML 技术人员和 SQL Server 用户带来强劲动力。Microsoft SQL Server 的 XML 支持计划，是其下一步庞大计划的一部分，即旨在产生一组功能强大的产品和服务来实现所谓的 BizTalk 框架。BizTalk 是 Microsoft 现有的、帮助商务公司实现应用软件一体化的分布式互连网应用 DNA（Windows Distributed interNet Applications ）体系结构的延伸和扩展。微软在其 Windows 分布式 Internet 应用（即 Windows DNA）架构中集成了 XML 技术。通过中间层的代理程序，可获取的数据来源可以不必局限于某台固定的数据库服务器，而可以是分布于企业内，甚至于遍及全球各地的数据库服务器。另外，借助于 XML Schema，开发者就能更为精确地描述和交换数据，因而大大地提高这种应用的效率。

以后的 Microsoft 产品和工具本身都将包含对 BizTalk 服务体系结构的支持。Microsoft

Commerce Platform，Office，BackOffice 以及 Windows 都将利用 BizTalk XML Schema 来保存文档的额外信息、并且用它来实现 BackOffice 和基于 Windows 应用软件的一体化集成。Microsoft 开发产品套件和 Microsoft Office 2000 发行的下一个主要版本都打算将 HTML 提升成为一种内置支持的文件格式，并且使用 XML 来存储额外的文档信息。Microsoft SQL Server 正是其实现这一目的的底层数据的有力工具。

从涉及和接受 XML 开始，Microsoft 就一直致力于将 XML 技术与其数据库旗舰产品 SQL Server 相集成，以帮助建立下一代高效的基于 Web 的企业应用。Microsoft 宣称，其下一版本 SQL Server，即代号为 Shiloh，将是一个完全支持 XML 的产品，利用该产品，用户可以在 Web 浏览器下输入一个 URL 地址，即可访问 SQL Server 数据库，而返回的结果可以是一个 XML 文档。另外，它还允许通过输入样式参数，指定样式信息，以便在浏览器中输出丰富的页面。一个典型的 URL 如下所述：href = http: // localhost / Northwind ? sql = select + firstname,lastname + from + employees + for + xml + auto。Microsoft 宣称，SQL Server Shiloh 将在最近正式发布，现在发布的预览版只是其全部支持中的一部分，其技术核心是 IIS ISAPI 的一个扩展，支持环境是 Windows NT 4.0 + IIS 4.0，或者 Windows 2000 +IIS 5.0，数据库为 SQL Server 2000、SQL Server 7.0 或 SQL Server 6.5 Service Pack 5。一个例外是，安装于 Windows 98 上的 Personal Web Server 将不提供对 XML 的支持。

建立 B2C、B2B 和外部网 Web 解决方案的公司希望用 XML 来简化后端系统集成和经由防火墙的数据传输。尽管许多公司只希望用中间层 XML 解决方案来解决其数据通讯问题，但是开发人员还认识到 XML 文档和数据的高速存储和生成功能的价值。Microsoft SQL Server 2000 提供了集成的 XML 支持，对于 Web 开发人员和数据库程序员而言，这一支持具有灵活性、高性能和易用性。

SQL Server 2000 中丰富的 XML 功能帮助 Web 开发人员绕过了复杂的关系数据库编程。他们可以使用诸如 XPath、URL 查询和 XML 更新记录的技术。类似的是，数据库开发人员并不需要学习一种面向对象的语言或深入了解 XML。他们可以利用 FOR XML 子句来提供对现有关系数据库的 XML 访问，该子句返回来自 SELECT 语句和 OPENXML T/SQL 关键字的 XML 数据。OpenXML 提供了 XML 数据的关系视图，T-SQL 可以利用该视图查询来自 XML 的数据，将 XML 数据与现有关系表连接，以及更新数据库。通过 FOR XML 子句，SQL Server 允许查询返回来自标准 SELECT 语句的 XML（而不是标准的行集）格式的数据。

除了检索 XML 格式的数据之外，以 XML 格式高效地存储数据的功能、在完全利用由高性能数据库（如 SQL Server）提供的速度的同时保持数据关系和层次结构的功能也是很重要的。SQL Server 2000 可以提供关系数据的 XML 视图，并可将 XML 数据映射为关系表。尽管 XML 视图允许访问 XML 文档形式的关系表，但 OpenXML 允许用关系 SQL 语法处理 XML 文档。OpenXML 是一个 T-SQL 关键字，它通过内存中 XML 文档提供一个可更新行集。行集中的记录可以存放在数据库表（类似于由表和视图提供的行集）中。OpenXML 可以用在 SELECT 和 SELECT INTO 语句中，只要语句中出现了诸如表、视图或 OPENROWSET 的行集提供程序。

SQL Server 2000 中的两个 XML 特性将在产品发布后可用。XML gram 是发送给服务器的命令，它允许 Web 开发人员使用 XML 插入、更新、删除来自 SQL Server 2000 表的数据。XML 的大容量装载设备实现了由 XML 封装的数据大容量装载。

1．XML SQL 技术预览提供了以下几种 XML 访问功能

（1）利用 HTTP 协议以 URL 的方式访问 SQL Server 数据库。

这是最基本的访问方式。另外，通过指定样式模板参数，可以返回具有一定样式信息的数据。该样式模板是一个包含一条或多条 SQL 语句的合法 XML 文档。返回的 XML 文档可以通过指定的 XML 模式来定义，三种模式为：RAW，AUTO，EXPLICIT。

实例：http://IISServer/northwind?sql=SELECT+*+FROM+Customers+FOR+XML+AUTO

（2）利用在 SELECT 语句中附加 FOR XML 来返回 XML 格式数据。

作为对 XML 模式的一种补充，通过在 FOR XML 中指定 DTD 或 XML schema 来达到对返回 XML 文档的格式化。

实例：如果我们安装了 SQL Server 并建立了 Northwind 数据库，那么可以通过浏览器来进行查询。

href=http://localhost/Northwind?sql=select+firstname,lastname+from+employees+for+xml+auto

（3）利用基于 XML 的 UPDATE 语句来更新数据库中的记录。

SQL Server 支持基于 XML 的插入、删除、修改等数据库更新操作。通用的更新语法是：

```
<sql:sync xmlns:sql="urn:schemas-microsoft-com:xml-sql">
<sql:before>
<TABLENAME [sql:id="value"] col="value" col="value"…../>
</sql:before>
<sql:after>
<TABLENAME [sql:id="value"] [sql:at-identity="value"] col="value" col="value"…../>
</sql:after>
</sql:sync>
```

插入操作的更新语法是：

```
<sql:sync xmlns:sql="urn:schemas-microsoft-com:xml-sql">
[…
</sql:before>]
<sql:after>
<TABLENAME [sql:id="value"] [sql:at-identity="value"] col="value" col="value"…../>
</sql:after>
</sql:sync>
```

删除操作的更新语法是：

```
<sql:sync>
<sql:before>
<TABLENAME KeyCol1="PKCol1Value" KeyCol2="PKCol2Value"
..
ColA="Value" ColB="Value"../>
</sql:before>
[<sql:after>
</sql:after>]
</sql:sync>
```

修改操作的更新语法是：

```
<sql:sync>
<sql:before>
<TABLENAME [sql:id="value"] KeyCol1="KeyCol1Value" KeyCol2="KeyCol2Value"
..
ColA="OldValue" ColB="OldValue"../>
</sql:before>
<sql:after>
<TABLENAME [sql:id="value"] KeyCol1="KeyCol1Value" KeyCol2="KeyCol2Value"
..
colA="NewValue" colB="NewValue"…../>
</sql:after>
</sql:sync>
```

2. 利用对象模型在 Script 脚本程序中实现同样的功能

XML SQL 技术预览提供的 osqlxml.dll 允许利用对象模型在 Script 脚本程序中实现基于 XML 的数据库操作。下面是一个 ASP 示例：

```
<%@ LANGUAGE = VBScript %>
<% Response.ContentType = "text/xml" %>>
<%
SET oSQLXML = CreateObject("Microsoft.SQLXMLRequest")
oSQLXML.Connection = "Driver=SQL Server;
                Server=FRANKMAN-CAVE;Database=Northwind;uid=sa;pwd="
oSQLXML.ExecuteQuery("Select * from customers for XML AUTO")
Response.BinaryWrite oSQLXML.ResultAsBinary
%>
```

14.2.2 Oracle 的“声音”

作为数据库领域的领跑者，Oracle 公司同样对 XML 非常重视。旗舰产品 Oracle8i 是面向 Internet 计算环境的数据库，它改变了信息管理和访问的方式。Oracle8i 将新的特性融入到了传统的 Oracle 服务器之中，从而成为一个面向 Web 信息管理的数据库。

Oracle 公司今年三月正式宣布，为消除可扩充标记语言（XML）和 Java 之间的开发差距，它将扩大对 XML 的支持。尤其是，该公司已增加其 JDeveloper 3.1 产品的 XML 支持力度，并把 XML Schema Parser（XML 方案语言分析程序）添加到它的 XML Developer’s Kit。

为帮助建立对称性，以防止 Java/XML 突变，Oracle JDeveloper 3.1 现可支持 XML 结构化查询语言（XSQL）页面，从而允许开发人员编辑和调试 Java 程序，这些 Java 程序可查询数据，并送回格式化 XML。该支持也允许开发人员把 XML 插入他们的数据库，而无需重新编写代码。

此外，JDeveloper 的集成式小服务程序引擎（Servlet Engine）也允许开发人员观看相同环境中用 Java 代码生成并作为他们程序源码的 XML 输出结果。JDeveloper 3.1 也包括突出 XML 彩色编码语法重点和检查 XML 和可扩展风格语言（XSL）内部语法的功能。

除了在 JDeveloper 3.1 中支持 XML 以外，Oracle 公司也已给它的 Oracle XML Developer’s Kit（XDK）增加 XML Schema Processor for Java。根据万维网联盟最近的 XML Schema 标准工作草案规定，Schema Processor 既可支持简单数据类型，又可支持复杂数据类型，从而可处理和确认 XML Schemas。Oracle XML Developer’s Kit 现可从 Oracle 公司技术网（OTN）下载

获得，网址为 technet.oracle.com。JDeveloper 3.1 从四月份开始就可以在 OTN 上获得。

1. Oracle 的 XML 战略

Oracle 的 XML 战略很简单：为开发者提供使用 XML 高效、低投入地创建可靠的、可扩展的 Internet 应用程序的最佳平台

使用 Oracle8i，开发者可以得到基于 Java 的 XML 支持。其他相应的 XML 特性在 Oracle 全系列产品中都有具体的涉及：从数据库到应用服务器到开发工具到打包的应用程序。下面是 Oracle8i Java/XML 开发平台结构图。

Oracle 8i 数据库产品中提供了对 XML 的支持，使其成为开发基于 XML 的 Internet 应用的理想平台。作为 Oracle 8i 数据库的一个主要组成部分，Java 版本的 XML SQL 实用工具包包含了一组 Java 类，可完成下述两大功能：

- 向数据库发送查询语句，然后从返回的结果中输出 XML 文档（文本或 DOM 节点树）。
- 将 XML 数据保存至数据库中。

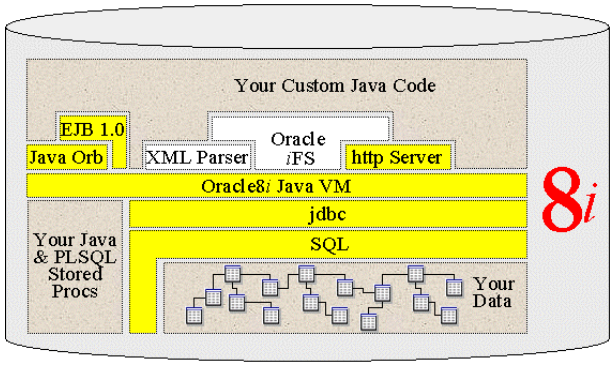


图 14-2　Oracle8i Java/XML 开发平台

这两大功能可以说是 SQL 数据库支持 XML 的最基本要求，但足以体现当今 XML 技术发展的潮流。与微软下一代数据库产品——Microsoft SQL Server Shiloh 在 2000 年中正式发布相比，无形之中，Oracle 占据了良好的先机。

Oracle Java 版本的 XML SQL 实用工具具有下列特性：

- 能够从 SQL 查询中产生 XML 文档。
- 能够从 SQL 查询语句或 JDBC ResultSet 对象中输出文本或文档对象模型节点树（DOM）。
- 能够将 XML 文档数据写入数据库表或视图中。
- 支持 W3C 的 XML 1.0 推荐标准。
- 可以通过扩展进而支持 SAX 1.0 文档访问。
- 支持基于下列字符集的文档： UTF-8，UTF-16，ISO-10646-UCS-2，ISO-10646-UCS-4，EUC-KR，US-ASCII，EBCDIC-CP-*，ISO-8859-1to –9，BIG，GB2312，EUC-JP，KOI8-R，ISO-2022-JP，ISO-2022-KR，Shift_JIS。

XML SQL 实用工具输出的 XML 文档实际上根据数据库表内在结构而动态确定的，比如，将 ROWSET 做为返回结果集的根元素，每行数据将 ROW 做为元素标记，而每个字段名称都

将做为 ROW 元素下的子元素。下面给出一个典型示例：

假定查询语句为：

SELECT ISBN, BOOKNAME FROM BOOKS，则返回的 XML 文档可能类似下面的语句：

```
<? xml    version = "1.0" encoding="GB2312" ?>
<ROWSET>
  <ROW id="1">
    <ISBN>7-302-02015-9</ISBN>
    <BOOKNAME>VISUAL C++技术内幕</BOOKNAME>
  </ROW>
  <ROW id="2">
    <ISBN>7-302-00860-4</ISBN>
    <BOOKNAME>C 程序设计</BOOKNAME>
  </ROW>
  ……
  ……
</ROWSET>
```

14.2.3 结合 XML 的数据库工具

下面是部分当前可以利用的数据库工具，这些工具分成中间件、XML 服务器和内容管理系统三类。这种分类的边界经常是模糊的，但是作为一个大概的规则，中间件是指配合现有的数据库一起使用的软件，XML 服务器是把一个数据库和一个中间件软件结合在一起，协调它们用来进行数据的存储和运用程序的开发，一个内容管理系统也是将一个数据库和一个中间件软件结合在一起，协调它们用来进行文件的存储。

一般来说，中间件和 XML 服务器需要完成的编程工作要面面俱到但都是些琐碎的工作。而内容管理系统恰恰相反，需要做到仅仅是配置而已，尽管这对内容管理系统本身而言也许会是些琐碎麻烦的事情。这两者的不同部分是因为内容管理系统更加成熟，其实主要是因为中间件和 XML 服务器在实际运用中通常只是一个实际需要完成的庞大运用程序的一部分而已。

注：这些工具的信息都是基于阅读它们的文档和网站得来的，信息获取的截止日期是 2000 年 3 月 22 日。如需了解最新动态，请查阅相关网站和文档。笔者并没有使用这些产品的实际经验，所以建议用户在进行自己的设计时仅仅把这些信息当成一个介绍。

1. 中间件

中间件是一种使用在数据库和 XML 文件之间传输数据的软件，它可以用多种程序代码来编写，但大部分的中间件都是使用 ODBC，JDBC 或者 OLE DB。

4ODS

开发者：FourThought

URL：http://opentechnology.org/4Suite/

许可证：开放源代码（Open Source）

数据库类型：面向对象数据库（ODMG 2.0）

传输方向：Database=>XML，XML=>Database

这是一个符合 ODMG 2.0 面向对象数据库标准的前端工具，功能非常强大。各种数据库驱动都可以嵌入 4ODS 来进行后端的数据存储；4ODS 特别适用于为 XML 文档提供相对固定

的结构。4ODS 很快得到广泛的应用。

ADO

开发者：Microsoft

URL：http://www.vbxml.com/xml/guides/Developers/ADO_persist_XML.asp

许可证：商业

数据库类型：关系数据库（ODBC 或者 OLE-DB）

转化方向：Database=>XML, XML=>Database（?）

在微软的站点上，我们无法找到 ADO 和 XML 如何协同工作的技术细节。但微软在其站点上展示了一个 ADO 记录集和 XML 文档相互转化的简单实例。这有可能是模型驱动的：记录集模型化一个表。因为 ADO 记录集可以模型化并存储到数据库中，那么 XML 文档或许也可以像 ADO 记录集那样打开和存储在数据库中。

微软的 ADO 2.5 声明支持 XML 和数据库之间的数据传输，ADO 把 XML 当成一个 OLE DB 驱动对待。

Beanstalk

开发者：Transparency

URL：http://www.transparency.com/

许可证：Commercial

数据库类型：关系数据库（ODBC）

转化方向：Database=>XML, XML=>Database

这是一个位于应用程序和数据库之间的“对象——关系”引擎。

Castor

开发者：exolab.org

URL：http://castor.exolab.org/index.html

许可证：开放源代码（Open Source）

数据库类型：关系数据库（JDBC）

转化方向：Database=>XML, XML=>Database（?）

Castor 是一个支持 Java 对象使用 XML、关系数据表和 LDAP 的强大引擎。

ASP2XML

开发者：Stonebroom

URL：http://www.stonebroom.com/asp2xml.htm

许可证：商业

数据库类型：关系数据库（ODBC 或者 OLE-DB）

转化方向：Database=>XML, XML=>Database

这是一个 OLE COM 组件，用来在 XML 文档和任意 ODBC 或者 OLE-DB 数据源之间传输数据。该产品是模型驱动的，并且将 XML 文档模型化成一个单一的表对象。当将数据从数据库传输成 XML 时，用户制定一个单一的 SELECT 声明，输出包含 ASP2XML 自定义的标志。当将数据从 XML 传输给数据库时，XML 文档中必须包含有 ASP2XML 自定义的标志，它们是该中间件处理时要使用到底。这个组件可以使用在 Active Server Pages 脚本中，也可以当一个普通的组件使用。

DatabaseDOM

开发者：IBM

URL：http://www.alphaworks.ibm.com/tech/databasedom

许可证：仅仅是评估版本

数据库类型：关系数据库（JDBC）

传输方向：Database=>XML, XML=>Database

这是一个在 DOM 树和一个 JDBC 数据库之间阐述数据的 JavaBean。它是模板驱动的，在模板中根据数据库访问信息和数据布局分成不同的小节。当将数据从数据库传输给 DOM 树时，用户制定一 SELECT 声明或者一个表名和一个 WHERE 字句；当将数据从 DOM 树传输给数据库时，用户仅仅只要制定一个单一的表名。在该网站上申明该产品是“设计给 WebSphere 使用”；所以并不能够明确说明是否适用于其他环境。

DataCraft

开发者：IBM

URL：http://www.alphaworks.ibm.com/formula/datacraft

许可证：仅仅是评估版本

数据库类型：DB2, Microsoft Access

传输方向：Database=>XML

一个可以用来生成 SELECT 说明的工具，能够把数据库中单个表中的数据结果传输到一个 XML 文档中。在该网站上如此说：“... an application generation tool targeted for RDF/XML applications in the context of Web-commerce applications. DataCraft, a facility capable of generating visual query skeletons and running the queries against DB2, is an excellent tool for Web-Database application generation using XML. DataCraft provides client tools visually navigating resource schema, and query language building queries visually from the schema based on XML and RDF. DataCraft uses RDF and XML to describe data collection structures and to exchange resource schema and query between the server and client.”

DB2XML

开发者：Volker Turau

URL：http://www.informatik.fh-wiesbaden.de/~turau/DB2XML/index.html

许可证：开放源代码（Open Source）

数据库类型：关系数据库 （JDBC）

传输方向：Database=>XML

是一个 Java 类，用来将关系数据库中的数据传输到一个 XML 文档中。这些类可以使用在一个单独的运用程序中也可以当成一个 servlet。该产品是基于模型驱动的，根据用户特定的一个或者多个 SELECT 声明，将 XML 文档模型化成一系列的表对象。可以选定需要输出的标志名称，以及是否包括文档中的数据库元数据。返回结果可以是一个文件，或者数据流、DOM 对象和一个支持传输的 XSL。

DBIx::XML_RDB

开发者：Matt Sergeant

URL：http://theory.uwinnipeg.ca/CPAN/data/DBIx-XML_RDB/XML_RDB.html

许可证：开放源代码（Open Source）

数据库类型：关系数据库（DBI）

转化方向：Database=>XML, XML=>Database

该产品是一个 PERL 的 module，用来在 XML 和 DBI 数据库之间传输数据。它是基于模型驱动的，将 XML 文档模型化成一系列的表对象。该包裹包括一个数据传输（转换）module 和两个实用工具：一个用来导出数据一个用来导入数据。当然一个 Win32 OLE 的 wrapper 允许用户在任何一个支持 OLE 的运用程序中调用这个 module。值得注意的是，这个 module 支持传输（转换）二进制数据，将数据进行 UTF-8 编码。

DB-X

开发者：Swift Inc

URL：http://210.238.31.12/en/frame/products/XMLServerWare/DB-X/

许可证：商业

数据库类型：关系数据库（ADO）

转化方向：Database=>XML

它是一个 Windows 运用程序（当然也可以用做 ISAPI 扩展），专门用来将一个或者多个 ADO 的 Recordsets 中的数据转换到一个 XML 文档中。该产品是模板驱动的，使用特定的 DB-X 元素将查询语句内嵌入模板中。查询后得到的结果可以存放在文档中的任何一个部分，并且该语言非常灵活，支持脚本程序结构，例如循环和条件判断。

InterAccess

开发者：XML Software Corporation

URL：http://www.xmlsoft.com.au/iaccess.html

许可证：商业

数据库类型：关系数据库（ODBC, OLE DB）

转化方向：Database=>XML, XML=>Database

这是一个基于 Client/Server 模式通过 Internet 访问 ODBC/OLE DB 数据库的软件包，使用 TCP/IP 作为数据传输协议。

ODBC2XML

开发者：Intelligent Systems Research

URL：http://members.xoom.com/gvaughan/odbc2xml.htm

许可证：共享软件

数据库类型：关系数据库（ODBC）

转化方向：Database=>XML

它是一个 Windows 下的动态链接库（DLL，可以运行在命令行方式），用来将数据从 ODBC 数据库转换到一个 XML 文档中。该产品是模板驱动的，将 SELECT 内嵌入处理指令中。处理起来也非常灵活，因为内嵌查询的结果可以被直接放置到一个元素或者属性中，也可以附加其它的查询，所以可以用来生成嵌套的 XML 文档。

XML SQL Utility for Java 和 XSQL Servlet

开发者：Oracle

URL：http://technet.oracle.com/tech/xml/

许可证：仅仅供开发使用

数据库类型：关系数据库（JDBC）

转化方向：Database=>XML（两者都可）, XML=>Database（只有 SQL Utility 可以 ）XML SQL Utility for Java 是一套 Java 类，用来在关系数据库和 XML 文档资料之间传递数据。这些类可以运用在前端或者是用户自己编写的程序中。该产品是模型驱动的。如果数据库支持 SQL 3 对象视图，那么文档的模型将是一个对象树；否则模型是文档被当成一个单独的表对象。当从数据库往 XML 中转换数据时，用户可以运用 SELECT 声明也可以使用 JDBC 结果集；返回的结果可以是一个 XML 文档也可以是一个 DOM 文档。当从 XML 向数据库转换数据时，用户可以使用一个 XML 文档也可以使用一个 DOM 文档。在输出的结果中特定的标志是可选的。

XSQL Servlet 是一个 Java servlet，它使用 XML SQL Utility for Java 来将数据从关系数据库中转换到 XML 文档中。这个 servlet 是模板驱动的，将 SELECT 申明内嵌入元素<query>中；在处理时，它们将被查询结果替换。它支持通过 HTTP 方式传递查询参数，也支持根据提供的 XSL 处理输出的文档。

XMLDB and XML Servlet

开发者：Cerium Component Software Incorporated

URL：http://ceriumworks.com/tech.html

许可证：商业

数据库类型：关系数据库

转化方向：（XML Servlet） Database=>XML, XML=>Database

XMLDB 使用一个 XML 文档，专门生成一个 SQL 代码 CREATE TABLE 申明来建立储存 XML 文档的表。该申明可以包含 NOT NULL, PRIMARY KEY 和 FOREIGN KEY 等约束条件。

XML Servlet 是模板驱动的，使用一个在后台（数据库）和前端（浏览器）基于 XML 的语言结构，例如 SQL 查询和 HTML 表格。就是说在数据库中执行 SQL 查询将结果显示在浏览器中或者到一个 HTML 表格然后进入数据库。该产品为了建立 XML 文件和数据库之间的映射关系，将 XML 文档资料当成一个单独的表列出来。

XML Lightweight Extractor (XLE)

开发者：IBM

URL：http://www.alphaworks.ibm.com/tech/xle

许可证：仅仅是评估版本

数据库类型：关系数据库（JDBC）

转化方向：Database=>XML

XOSL

开发者：Mey & Westphal RIPOSTE Software

URL：http://www.riposte.com/xosl/

许可证：仅仅供开发使用

数据库类型：关系数据库（ADO）

转化方向：Database=>XML

2. XML 服务器

DataChannel Server
开发者：DataChannel
URL：http://www.datachannel.com/products/dcs4.html
数据库类型：分层数据库 / "Native XML"

eXcelon
开发者：eXcelon Corp.
URL：http://www.exceloncorp.com/products/excelon.html
数据库类型：面向对象、关系数据库（ODBC and OLE-DB ）通过 eXcelon 浏览器。

Rhythmyx
开发者：Percussion Software
URL：http://www.percussion.com/Products/Rhythmyx/rhythmyxover.htm
数据库类型：关系数据库 （JDBC, ODBC）

XML Portal Server (XPS)
开发者：Sequoia Software Corp
URL：http://www.sequoiasw.com/xps/index.asp
数据库类型：分布式数据库。

3. 内容管理系统

Astoria
开发者：Chrystal Software
URL：http://www.chrystal.com/products/astoria/astoria.htm

BladeRunner
开发者：Interleaf
URL：http://www.xmlecontent.com/products/brintro.htm

Documentum
开发者：Documentum, Inc
URL：http://www.documentum.com/products/content/index.html

Epic
开发者：Arbortext
URL：http://www.arbortext.com/Products/Epic/epic.html

GroveMinder
开发者：TechnoTeacher
URL：http://www.techno.com/

POET Content Management Suite
开发者：POET
URL：http://www.poet.com/products/cms/cms.html

SIM (Structured Information Manager)
开发者：Progressive Information Technologies

URL：http://www.simdb.com/simdb%20content/About%20SIM

WSDOM XML-Portal

开发者：Radian Systems

URL：http://www.radsys.com/products/portal.htm

14.3 XML 查询语言——XQL

14.3.1 概念与特性

XML 查询语言（XQL）是用于定位和过滤 XML 文档中元素和文本的符号。它是 XSL 模式语法的自然扩展，为指向特定的元素或查找具有指定特征的节点提供了简明的可以理解的符号。该建议于 1998 年 9 月提交给 XSL 工作小组（http://www.w3.org/Style/XSL/Group/1998/09/XQL-proposal.html），在考虑 XSL 模式语法的扩展时作为基础输入。XSL 模式语言（ http://www.w3.org/TR/WD-xsl, section 2.6）提供了易于理解的方式描述待处理节点。它是说明性而不是过程性语言，只需使用类目录结构的简单模式描述需要查找节点的类型。例如，book/author 表示查找包含在 book 元素中的 author 元素。

XQL（XML 查询语言）提供对 XSL 模式语言的自然扩展。在 XSL 表示类型节点的基础上增加了布尔逻辑、过滤、节点集合索引等。

XQL 是为 XML 文档特定设计的，它是一种通用查询语言，提供用于查询、定位和模式的简单语法。XQL 精炼、简单且具有强大的功能。

XQL 是在查询语言 UnQL 和 StruQL 基础上设计的，它能对 XML 文档进行查询、构造、转换和集成。它集中了查询语言技术和 XML 语法格式，通过说明路径表达式和模式的方式，给出 XML 数据的提取条件（WHERE 子句）。同时，XML-QL 中可以给出构造查询输出的 XML 数据的模板，其输出结果仍为 XML 文档（CONSTRUCT 子句）。当操作对象限定为关系类数据时，XQL 与关系演算或关系代数有同样的表达能力，即 XML-QL 是关系完备的。

XQL 有类似 select-from-where 的结构（where-in-construct），与 SQL 很相似。但 XQL 有一些很重要的区别于基于结构化数据查询语言的特点。其 WHERE 子句由模式和关系表达式组成，这意味着被选出的数据项要满足两个条件：

- 数据项的类型（或 Schema）和值必须与指定的模式匹配。
- 数据项的值要满足关系表达式。

在查询条件中加入模式匹配是 XQL 与半结构化查询语言和结构化查询语言最大的不同之处。XQL 的标准结构见图 14-3。

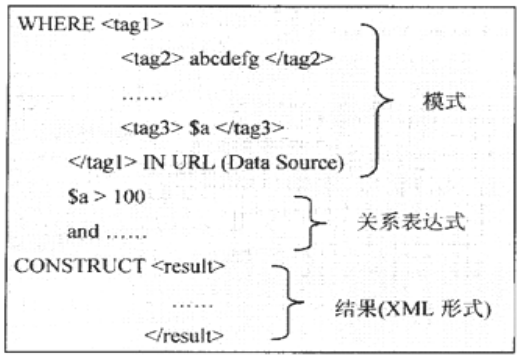


图 14-3　XQL 标准结构

XQL 的设计目标如下：

- XQL 字符串必须紧凑。
- XQL 应该易于打字和阅读。
- XQL 语法应简单，以便用于简单和通常情况。
- XQL 应该用易于嵌入程序，脚本和 XML 或 HTML 属性的字符串表述。
- XQL 应易于分析。
- XQL 应该用嵌入 URL 的字符串表述。
- XQL 应能够指定 XML 文档中可能的任何路径和指定任何条件集合以查找路径上的节点。
- XQL 应能唯一指定 XML 文档中的所有节点。
- XQL 查询可以返回任何数量的结果，包括 0。
- XQL 查询是说明性，而非过程性。即：说明查找什么，而不是如何查找。这一点尤为重要，因为查询优化器必须能够自由选择索引和其他结构以便高效地找到结果。
- XQL 查询可以在文档的任何级别开始计算，而不必从文档根元素开始导航。
- XQL 查询按照文档顺序，无重复地返回结果。

14.3.2　XQL 模式及实现

1. XQL 的特点

我们将通过一些例子来让大家先了解并初步熟悉 XQL 的主要特点。

（1）用模式（Patterns）来匹配数据 XQL 利用元素模式来匹配 XML 文档。下面的例子将从 www.a.b.c/bib.xml 所指的 XML 的文档中找出所有 Addison-Wesley 出版的书的作者。

```
WHERE
        Addison    Wesley        $a
         IN “www.a.b.c/bib.xml"
CONSTRUCT  $a
```

（2）构造新的 XML 数据。许多 XML 应用需要查询结果构造出新的 XML 数据。XQL 通过在 CONSTRUCT 子句中定义一个输出模板来支持 XML 数据的输出。例如下面的 CONSTRUCT 子句。

```
CONSTRUCT       < result >
```

```
        <author>$a</>
```

（3）用嵌套查询来处理可选元素。XML 与关系数据的一个主要区别在于 XML 经常有可选的元素。假如我们要列出所有的书名，有价格的要同时列出价格。看如下的查询：

```
WHERE<book><title>$T</><price> $p</>
   </> IN www.a.b.c/bib.xml
CONSTRUCT <result><booktitle> $t</>
                <bookprice>$p</>
          </>
```

上述查询是不正确的，因为它要求必须含价格，没有价格的书将不被列出。XQL 可通过嵌套查询来处理可选部分，我们将上面的查询修改如下：

```
WHERE<book>$b</>IN "www.a.b.c/bib.xml",
      <titile>$t</> IN $b
CONSTRUCT <result><booktitle>$t</>
    WHERE <price>$p</>IN $b
    CONSTRUCT <bookprice>$p</>
  </>
```

IN 的右边可为 URL，也可为变量。第二个 WHERE… CONSTRUCT 查询称为嵌套查询。

（4）用嵌套查询来分组。关系数据库与 XML 的另一个区别是嵌套和分组。例如，我们可以将一本书的所有的作者组合到一个<book>元素中。假如我们要找出每个作者写的所有的书的书名，可用下面的查询来重组数据：

```
WHERE<book  >$p</>IN"www.a.b.c/bib.xml",
      <author>$a</>IN $p
CONSTRUCT<result>
          <author>$a</>
          WHERE<author> $a</>,
                <title>$t</> IN $p
          CONSTRUCT <title> $t</>
         </>
```

（5）绑定元素和内容。在 XQL 中，变量被绑定到半结构化数据的节点上。对 XML 而言，变量被绑定为元素内容而非元素本身。XQL 文法允许我们将变量绑定到元素。

（6）连接。在两个匹配表达式中使用同一个变量可实现链接。下面的例子找出至少出版了两本书的所有作者。

```
WHERE<book><author>$a</></>
  content_as $b1 IN"www.a.b.c/bib.xml",
      <book><author> $a</></>
       content_as $b2 IN"www.a.b.c/bib.xml",
  b1 != b2
CONSTRUCT <result> $a</>
```

（7）Tag 变量。有时 XML 文档使用不同的 Tag 代表具有相同概念的不同实体。

（8）正则路径表达式。考虑下面的 DTD 规定的递归定义：

```
  <!ELEMENT part (name brand part   )>
  <!ELEMENT name CDATA>
  <!ELEMENT brand CDATA>
```

其中每一个元素可以包含其他任意深度的嵌套部分。为实现这样的结构，XML-QL 提供

了规则的路径表达方式，它可以详细说明任意深度的元素路径。例如下面的查询将产生每一个组成元素 name 部分的内容，这些组成元素 brand 的内容为“Ford”，而不论 name 的内容包含多少层的嵌套。

```
WHERE < part      >< name > $ r< / >< brand >
Ford< / > < / > IN “www.a.b.c/bib.xml"
CONSTRUCT < result > $ r< / >
```

这里的 part 是正则路径表达方式，它可以表达 DTD 中规定模式中组成部分的任意序列。

（9）XML 数据转换。XQL 具有把符合某一 DTD 的 XML 文档变为符合另外一种 DTD 的 XML 文档的功能。

（10）从不同的 XML 数据源集成数据。XML-QL 可以同时查询由多个数据源产生的数据的完整视图。

（11）顺序。XML 的数据模式是顺序化的，并且在一些应用中是非常重要的。XML-QL 有两种不同的语法规则：一个是顺序化的数据结构，另一个是非顺序化的数据结构，并且需要不同的查询处理器为这两种模式服务。ORDER-BY 子句可以指定输出中的排序元素。

2. XML-QL 基本语法

XQL 基本语法模仿 URI 目录导航语法，但不是通过物理文件结构指定导航，而是通过 XML 树的元素。例如，下例 URI 表示在 bar 目录中查找 foo.jpg 文件：

```
bar/foo.jpg
```

类似地，在 XQL 中，下例表示查找在 baz 元素中查找 fuz 元素集合：

```
baz/fuz
```

注　本节中涉及的许多示例，都参照附录 Sample Data 中的数据。

（1）上下文。“上下文”是当前查询操作的节点集合。要理解上下文的概念，需要考虑包含多个节点的树。从根节点开始查找所有命名为“X”的节点将返回结果集合，从树枝开始查找所有命名为“X”的节点将返回不同的结果集合。因此，查询结果取决于执行时的上下文。有许多不同方法可以用于指定查询的输入上下文。

XQL 允许查询在当前上下文和“根上下文”选择作为输入上下文。“根上下文”是一个包含文档中最根部元素的上下文。缺省地，查询使用当前上下文，以“/”（前倾斜）作为前缀的查询使用根上下文。查询可以随意使用“./”（点，前倾斜）作为前缀显式说明使用当前上下文。这两个符号都是文件系统导航符号的模拟。

“./”前缀只在一种情况下必须存在。使用“//”操作符表示递归后代。当该操作符出现在查询开头时，初始的“/”引起相对于文档或存储根部的递归后代执行。前缀“.//”允许查询当前上下文的递归后代。

示例：在当前上下文中查找所有 author 元素。由于句点不单独使用，该例向前应用了其他特征：

```
./author
```

请注意，该例等价于：

```
author
```

查找文档中的根元素（bookstore）：

```
/bookstore
```

在当前文档中查找所有出现在任何地方的 author 元素：

```
//author
```

在文档根部查找所有 book 元素的 style 属性值与 bookstore 元素 specialty 属性值相等的所有 book 元素。

```
book[/bookstore/@specialty = @style]
```

（2）查询结果。XQL 表达式返回的集合在定义范围内保持文档顺序，层次结构和标识。即：元素集合总是无重复地以文档顺序返回，属性集合将无重复地返回，但由于属性无顺序地定义，返回集合并不意味着顺序。

（3）集合。带有特定标记名称的所有元素集合用标记名称自身表示。元素查询可以被限定在当前上下文“./”，但当前上下文是认定的，通常无需显式指定。

示例：

查找所有 first-name 元素。以下例子是等价的：

```
./first-name
first-name
```

查找所有未限定的 book 元素：

```
book
```

查找所有 first.name 元素：

```
first.name
```

（4）选择子辈和后代。某些类型的元素集合可以用路径操作符（“/”或“//”）指定。路径操作符将左部作为查询来源的参数集合，右部表示需要查询的元素。子女操作符（“/”）查询左部元素集合的直接子女集合，后代操作符（“//”）查询左部元素集合的任意后代集合。事实上，“//”可以被视为一个或者多个层次结构的替代方案。注意路径操作符在查询过程中改变上下文。通过将其串在一起，用户可以深入文档内部。

示例：

在 author 元素中查找所有 first-name 元素。首先查找当前环境的 author 子女，然后查找相对于 author 元素环境的 first-name 子女：

```
author/first-name
```

在 bookstore 元素一级或多级以下（任意后代），查找所有 title 元素：

```
bookstore//title
```

注意，该例与下述查找所有 bookstore 元素的孙子 title 元素的查询并不相同：

```
bookstore/*/title
```

查找在 book/excerpt 元素中任意位置的 emph 元素，而且 book/excerpt 元素在 bookstore 元素中任意位置：

```
bookstore//book/excerpt//emph
```

在当前环境一级或多级以下查找所有 title 元素。注意该情况本来是要求句点符号的唯一情况：

```
.//title
```

（5）搜集子元素。可以不用标记名称而用“*”集合引用一个元素。“*”集合不管子元素的标记名称，返回当前环境的所有子元素。

示例：

查找 author 元素的所有子元素：

author/*

查找 book 元素的所有 last-name 孙子元素：

book/*/last-name

查找当前环境的所有孙子元素：

/

查找具有指定属性的所有元素。注意该例使用了在过滤中涉及的子查询和在查找属性中讨论的属性：

*[@specialty]

（6）查找属性。属性名称用“@”符号开头。XQL 设计公平对待属性和子元素，在可能的情况下，两种类型的能力是等价的。

注意，属性不能包含子元素。因此，在查询中属性不能使用路径操作符。否则将导致语法错误。同样地，属性没有顺序，索引不能用于属性。

示例:

查找当前环境的 style 属性：

@style

在当前环境中查找 price 元素的 exchange 属性：

price/@exchange

下例不是有效的查询：

price/@exchange/total

查找所有包含 style 属性的 book 元素。注意该例使用了在过滤中涉及的子查询：

book[@style]

查找所有 book 元素的 style 属性：

book/@style

（7）分组。为使操作清晰或者常用的优先级不足以表示某个操作时，圆括号可以用于将集合操作符分组。

（8）过滤。通过对集合增加过滤子句“[ ]”，可以对任何集合进行限制和分枝。过滤是 SQL 中包含 ANY 语义的 WHERE 子句的模拟。过滤子句中包含了一个查询，称为子查询。子查询计算出布尔值，对集合中的每一个元素进行测试。集合中未通过子查询测试的元素将从结果集合中省略。

为方便起见，如果集合放于过滤之中，如果集合中包含元素，则产生布尔值 TRUE；如果集合为空，则产生 FALSE。本质上，如 author/degree 的表达式表示集合到布尔值的转换函数，像如下虚构的“存在一个”方法：

author[.there-exists-a(degree)]

注意，在表达式的给定级别上可以出现任何数目的过滤，但空过滤是不允许的。

示例：

查找至少包含一个 excerpt 元素的 book 元素：

book[excerpt]

查找至少包含一个 excerpt 元素的 book 元素的所有 title：

book[excerpt]/title

查找至少包含一个 excerpt 元素的 book 元素的 author，而且 author 元素至少包含一个 degree 元素：

book[excerpt]/author[degree]

查找包含 author 元素的 book 元素，而且 author 元素至少包含一个 degree 元素：

book[author/degree]

查找包含 excerpt 和 title 的所有 book 元素：

book[excerpt][title]

（9）布尔表达式。布尔表达式可以在子查询中使用。例如，可以使用布尔表达式查找特定值的节点集合，或者特定范围的节点集合。布尔表达式采取${op}$的形式，而{op}可以是任何{b|a}形式的表达式——即：操作符接收左值和右值参数，返回布尔结果。应用可以根据需要提供附加的布尔操作符，但实现结果令人不那么令人满意。

注意，XQL 表达式一节定义了附加的布尔操作符。操作符是大小写敏感的。

布尔与和布尔或　and和or用于执行布尔与和布尔或。

布尔操作符和分组括号结合起来，可以用于构成非常复杂的逻辑表达式。注意空格是无意义的，可以被省略，或者如下所示包含空格，用于增加可读性。

示例:

查找至少包含一个 degree 元素和一个 award 元素的 author 元素：

author[degree and award]

查找至少包含一个 degree 元素或者 award 元素，而且包含至少一个 publication 元素的 author 元素：

author[(degree or award) and publication]

布尔非　not是用于子查询中表达式值求反的布尔操作符。

示例：

查找至少包含一个 degree 元素，而且不包含 publication 元素的 author 元素：

author[degree and not publication]

查找包含 publication 元素，而且不包含 degree 元素和 award 元素的 author 元素：

author[not (degree or award) and publication]

（10）等值。“=”符号用于相等判断，“!=”用于不相等判断。作为选择，eq和ne 也可用于相等和不相等。

单引号或双引号可以用于在表达式中分隔字符串，使得在脚本语言中创建或传递 XQL 变得更为容易。

对于比较元素值而言，暗含了 value（）方法。也就是说，last-name < ‘foo’实际上表示 last-name!value（） < foo 。

注意，过滤总是和上下文相关的。即：表达式 book[author]表示对每一个找到的 book 元素，查看其是否包含 author 子元素。同样地，book[author = Bob]表示对每一个找到的 book 元素，查看其是否包含命名为 author，且值为“Bob”的子元素。我们也可以通过使用“.”查看当前上下文的值。例如，book[. = Trenton]表示对每一个找到的 book 元素，查看其值是否等于“Trenton”。

示例:

查找 last name 等于 Bob 的所有 author：

author[last-name = Bob]

author[last-name eq Bob]

查找 from 属性不等于 Harvard 的所有 author：

```
degree[@from !=  Harvard ]
degree[@from $ne$  Harvard ]
```

查找 last-name 与/guest/last-name 相同的所有 author：

```
author[last-name = /guest/last-name]
```

查找文本为’ Matthew Bob’的所有 author：

```
author[. =  Matthew Bob ]
```

比较与向量 比较的左值可以是一个向量或数量。而比较的右值必须是一个数量或者一个可以在运行时转换为数量的值。

如果比较的左值是一个集合，那么 any（exists）语义用于比较操作符。也就是说，如果集合中的任何元素满足条件，则比较结果为真。

比较与文字常量 表达式的左值不能是文字常量。也就是说，'1'=a 是不允许的。

比较过程的文字常量转换 所有元素和属性都是字符串，但是许多时候要求数字比较。如果右值是一个属性，text（lvalue）与 text（rvalue）进行比较。如果右值是一个文字常量，则运用下列规则：

常量类型	比较	示例
字符串	Text(lvalue) op text(rvalue)	a < foo
整数	(long) lvalue op (long) rvalue	a < 3
实数	(double) lvalue op (double) rvalue	a < 3.1

（11）方法。XQL 为高级集合操作提供方法。这些方法提供节点的特定集合（查看集合方法），同时提供关于集合和节点的信息。

方法的形式为：{方法}（参数表）。

考虑查询 book[author]。它将发现包含 author 元素的所有 book。一般来讲，我们称对应于特定 author 的 book 元素为该 author 的参考节点。也就是说，每一个检查的 author 元素都是其中一个 book 元素的 author(关于参考节点和其他术语的更为彻底的定义,请查看 Annotated XQL BNF 附录）。方法应用于参考节点。

例如：text（）方法返回节点内包含的文字，不包括所有结构（也就是说，是包含在元素及其后代的所有文字节点的连接）。下列表达式将返回所有名为 Bob 的 author：

```
author[text() = Bob ]
```

下例将返回包含 first-name 子节点，且该子节点的文字为 Bob 的所有 author：

```
author[first-name!text() =  Bob ]
```

下例将返回包含名为 Bob 的子节点的所有 author：

```
author[*!text() =  Bob ]
```

注意 方法名称对大小写敏感。

信息方法 下列方法提供关于集合内节点的信息。这些方法返回字符串或数字，而且可以用于连接子查询内的比较操作符。

text()	包含在元素内的文字。该方法连接所有后代节点的文字，不包括标记名称或属性值，注释等。而且将如 text()中讨论一样整理字符串(返回字符串)。
value()	返回一个元素值的类型转换版本(参看 Datatypes)。如果数据类型不支持或者该数据类型不提供，返回结果与 text()相同。
nodeType()	返回一个指示节点类型的数字：

元素	1
属性	2
文字	3
处理指令	7
注释	8
文档	9

nodeName()	节点的标记名称，包括名字空间前缀(返回字符串)。

文本 text（）方法连接所有节点后代的文字，根据节点属性可选地规范化空白空间。如果节点或者最近的祖先节点的 xml:space 属性设为 preserve，空白空间将被保留。当空白空间规范化时，将在整个字符串范围内进行。空格用于分隔节点间的文字。当实体引用在文档中使用时，扩展时空格不出现在实体引用的周围。

示例：

查找 last name 为 Bob 的 author：

```
author[last-name!text() =  Bob ]
```

该查询等价于：

```
author[last-name =  Bob ]
```

查找值为 Matthew Bob 的 author：

```
author[text() =  Matthew Bob ]
author[. =  Matthew Bob ]
```

集合索引函数 index（）返回父节点内该节点的索引号。索引号是从 0 开始的，所以 0 是第一个元素（返回一个数字）。

查找前三个 degree：degree[index（） lt 3]

注意索引函数与父节点相关。考虑下列数据：

```
<x>
  <y/>
  <y/>
</x>
<x>
  <y/>
  <y/>
</x>
```

下列表达式在每一个 x 返回第一个 y：x/y[index（） = 0]

简写 对于比较的目的而言，如果省略了方法名，value（）方法是暗含的。换句话说，当两个项目进行比较时，比较在两个项目的值之间进行。记住在没有类型信息时，value（）返回文字。

下列例子是等价的：

```
author[last-name!value() =  Bob  $and$ first-name!value() =  Joe ]
author[last-name =  Bob  $and$ first-name =  Joe ]
price[@intl!value() =  canada ]
price[@intl =  canada ]
```

（12）集合内索引。XQL 很容易在节点集合中发现一个特定的节点。简单地将索引号放在方括号（‘[’和‘]’）里即可。序号是从 0 开始的。

例如，下例查找第一个 author 元素：

```
author[0]
```

下例查找含有 first-name 子元素的第三个 author 元素：

```
author[first-name][2]
```

注意索引是相对于父节点。换句话说，考虑下列数据：

```
<x>
  <y/>
  <y/>
</x>
<x>
  <y/>
  <y/>
</x>
```

下列表达式将从每一个 x 中返回第一个 y：

```
x/y[0]
```

下例将返回 x 内所有 y 组成的集合中的第一个 y：

```
(x/y)[0]
```

下例将返回第一个 x 的第一个 y：

```
x[0]/y[0]
```

在集合中返回最后的元素

end（）方法对集合的最后一个元素返回真。注意 end（）是相对父节点的。

示例：

查找最后的 book：

```
book[end()]
```

查找每一个 book 元素的最后的 author：

```
book/author[end()]
```

在 book 的所有 author 构成的集合中，查找最后的 author：

```
(book/author)[end()]
```

14.3.3 XQL 扩展

XQL 基本语法提供在 XQL 客户端所需的最小功能集合。XQL 扩展描述了扩展 XQL 能力的额外功能。

1. 名字空间

在查询中建立名字空间是需要的。由于名字空间是局部范围，这一点尤其重要。因此，XQL 需要一个可以为查询的全局范围或者查询内的局部范围建立名字空间的机制。

尽管我们不得不建立明确的名字空间语法，该语法必须满足下列要求：

- 该语法必须能够设定一个缺省的名字空间用于查询内部。
- 该语法必须能够设定一个名字空间用于任何指定的范围。
- 该语法必须能够设定一个名字空间用于任何级别的查询。换句话说，人们应该能够在查询开始设定名字空间，即使它们并没有用于查询内部较深的元素。
- 该语法必须能够建立一个匹配名字空间的长名称的前缀。

如果没有在查询中指定名字空间，前缀则用于匹配。

如果名字空间定义在 session 上，而不是在每一个查询基础上，XQL 处理器可以更有效地执行。本规范没有为指定用于整个 session 的名字空间的集合描述 API 或 XML 格式。这是由应用定义的。

下列方法可以用于节点，返回名字空间信息：

baseName()　　返回节点的名称部分，不包括前缀。

Namespace()　　返回节点名字空间的 URI。

Prefix()　　返回节点的前缀。

示例：

查找所有未加限定的 book 元素。注意该查询不返回 my:book 元素：book

查找所有包含前缀 my 的 book 元素。注意该查询不返回未加限定的 book 元素：

my:book

查找所有包含前缀 my 而且含 author 子元素的 book 元素：

my:book[author]

查找所有包含前缀 my 的 book 元素，其中包含 author 子元素而且 author 子元素也包含前缀 my：

my:book[my:author]

查找所有包含前缀 my 的所有元素：

my:*

在所有名字空间下查找所有的 book 元素：

*:book

在所有名字空间下查找所有元素：

*

在 book 元素中查找含前缀 my 的 style 属性：

book/@my:style

2. 查找属性集合

用@*可以返回元素的所有属性，这一点对那些将属性视为记录域的应用程序具有潜在的作用。

示例：

查找所有当前元素上下文的所有属性：@*

查找所有名字空间下的所有 style 属性：

@*:style

查找名字空间 my 下的所有属性，包括名字空间 my 下元素的未加限定的属性：

@my:*

3. 比较

二进制比较操作符集合可以用于比较数字和字符串，结果返回布尔值。lt，le，gt，ge分别用于小于、小于或等于、大于、大于或等于。这些操作符也可用于大小写不敏感的形式：ieq，ine，ilt，ile，igt，ige。

单引号或双引号可以用于在表达式中分隔字符串，使得在脚本语言中创建或传递 XQL 变得更为容易。

所有元素和属性都是字符串，但是许多时候要求进行数字比较。关于类型转换信息，查看比较过程中文字常量转换和数据类型。

<、<=、>和>=是lt，le，gt和ge的简写方式。

示例：

查找 last name 为 Bob 而且 price>50 的所有 author 元素：

```
author[last-name =  Bob  $and$ price $gt$ 50]
```

查找 from 属性值不等于 Harvard 的所有 author：

```
author[@from !=  Harvard ]
```

查找 last name 开始于 M 或更大字母的所有 author：

```
author[last-name $ge$  M ]
```

查找 last name 开始于 M、m 或更大字母的所有 author：

```
author[last-name $ige$  M ]
```

查找前两个 book 元素：

```
book[index() $le$ 2]
```

查找 publications 超过 10 的所有 author：

```
author[publications!count() $gt$ 10]
```

（1）数据类型。如果提供了数据类型，value（）函数使用该类型决定元素的类型。

对于比较目的而言，左值总是转换为右值的类型，因此保证了类型在比较过程中不发生变化。任何不能强制转换的左值将从结果集中省略。

（2）类型转换函数。XQL 提供函数用于值的类型转换。类型转换函数可以转换文字常量或集合。

当前，只提供了 date 函数。它将值转换为日期，该值必须是具有日期格式的 XML 数据类型。

示例：

查找所有在 1995 年 1 月 5 日前出版的 book：

```
books[pub_date < date( 1995-01-01 )]
```

查找出版日期（pub_date）早于属性 first 中存储的值的所有 book：

```
books[pub_date < date(@first)]
```

4. 任一（any）和所有（all）语义

作者可以通过any和all关键字显式指示是否使用任一和所有语义。

any表示如果集合中的任一元素满足条件，则条件为真。all表示如果集合中的所有元素满足条件，则条件为真。

any和all关键字可以出现在任何表达式前。

示例：

查找其中一个 last name 是 Bob 的所有 author 元素：

```
author[last-name = Bob ]
author[$any$ last-name = Bob ]
```

查找所有 last name 都不是 Bob 的所有 author 元素：

```
author[$all$ last-name != Bob ]
```

查找第一个 last name 是 Bob 的所有 author 元素：

```
author[last-name[0] = Bob ]
```

5. 并（Union）和交（intersection）

$union$操作符返回多个元素，返回元素没有重新排序，没有重复元素，双重指定是暗含的。选择列表中的所有元素必须是当前选择上下文的后代。注意：由于这是并操作，返回集合每一个类型列表中可能包含 0 或更多的元素。为限定返回集合至少包含列表中每一个元素，使用在过滤一节讨论的过滤。

|是$union$的简写。

$intersect$操作符返回在两个集合之间的公共元素。元素无需重新排序。

示例：

查找所有的 first-name 和 last-name：

```
first-name $union$ last-name
```

从 bookstore 中查找所有 book 和 magazine：

```
bookstore/(book | magazine)
```

查找所有 book 和所有 author：

```
book $union$ book/author
```

查找所有的 first-name、last-name、来自于 book 或 magazine 内 author 的 degree：

```
(book $union$ magazine)/author/(first-name $union$ last-name $union$ degree)
```

查找 author/first-name 等于’Bob’的所有 book 和 price 小于 10 的所有 magazine：

```
book[author/first-name = Bob ] $union$ magazine[price $lt$ 10]
```

6. 集合方法

集合方法为文档中不同类型节点提供访问方式。这些集合可以被约束和索引。集合返回满足特定条件的参考节点的子节点集合。

textNode()　　文字节点的集合。

Comment()　　注释节点的集合。

Pi()　　处理指示节点的集合。

Element([name])　　所有元素节点的集合。如果提供了可选的文字参数，将只返回匹配特定名称的元素子节点。

Attribute([name])　　所有属性节点的集合。如果提供了可选的文字参数，将只返回匹配特定名称的属性节点。

Node()　　返回所有非属性节点。

示例：

查找当前上下文中每一个 p 元素中的第二个文本节点：

p/textNode()[1]

查找文档中第二个注释节点。关于文档根部上下文设置和细节，请查看上下文：

//comment()[1]

7. 聚合方法

下列方法基于一个集合产生聚合结果。

count（）返回集合中的节点数目。

8. 其他方法

下述方法返回指示节点类型的字符串：nodeTypeString（）

N	返回第 n 个元素
-n	返回从最后的元素倒数第 n 个元素。如：-1 表示最后一个元素，-2 表示倒数第 2 个元素。
M to n	返回从 m 到 n 的元素，其中 m,n 是包含的。

9. 祖先（ancestor）

Ancestor 查找满足查询的最近祖先。结果返回一个元素或空集。

ancestor（query）满足提供查询的最近祖先。

示例：

查找当前元素的最近的 book 祖先：

```
ancestor(book)
```

查找包含在 book 元素最近的 author 祖先：

```
ancestor(book/author)
```

10. 脚标操作符

可以返回元素的排列。为实现该功能，在脚标操作符（方括号）内指定一个表达式而不是单个值。该表达式可以是逗号分隔的由下列条目组成的列表。

返回用于指示节点类型的下列字符串之一：

- document
- element
- attribute
- processing_instruction
- comment
- text

示例：

查找第一个和第四个 author 元素：

author[0,3]

查找第一个到第四个 author 元素：

author[0 to 3]

查找第一个，第三个到第五个和最后一个 author 元素：

```
author[0,2 $to$ 4, -1]
```

查找最后一个 author 元素：

```
author[-1]
```

附：本节实例参照数据

Sample Data

```
<?xml version='1.0'?>
<!-- This file represents a fragment of a book store inventory database -->
<bookstore specialty='novel'>
  <book style='autobiography'>
    <title>Seven Years in Trenton</title>
    <author>
      <first-name>Joe</first-name>
      <last-name>Bob</last-name>
      <award>Trenton Literary Review Honorable Mention</award>
    </author>
    <price>12</price>
  </book>
  <book style='textbook'>
    <title>History of Trenton</title>
    <author>
      <first-name>Mary</first-name>
      <last-name>Bob</last-name>
      <publication>
        Selected Short Stories of
        <first-name>Mary</first-name> <last-name>Bob</last-name>
      </publication>
    </author>
    <price>55</price>
  </book>
  <magazine style='glossy' frequency='monthly'>
    <title>Tracking Trenton</title>
    <price>2.50</price>
    <subscription price='24' per='year'/>
  </magazine>
  <book style='novel' id='myfave'>
    <title>Trenton Today, Trenton Tomorrow</title>
    <author>
      <first-name>Toni</first-name>
      <last-name>Bob</last-name>
      <degree from='Trenton U'>B.A.</degree>
      <degree from='Harvard'>Ph.D.</degree>
      <award>Pulizer</award>
      <publication>Still in Trenton</publication>
      <publication>Trenton Forever</publication>
    </author>
```

```
    <price intl='canada' exchange='0.7'>6.50</price>
    <excerpt>
      <p>It was a dark and stormy night.</p>
      <p>But then all nights in Trenton seem dark and
      stormy to someone who has gone through what
      <emph>I</emph> have.</p>
      <definition-list>
        <term>Trenton</term>
        <definition>misery</definition>
      </definition-list>
    </excerpt>
  </book>
  <my:book style='leather' price='29.50' xmlns:my='http://www.placeholder-name-here.com/schema/'>
    <my:title>Who's Who in Trenton</my:title>
    <my:author>Robert Bob</my:author>
  </my:book>
</bookstore>
```

14.3.4 综合实例

我们结合已经讨论和介绍的 XQL 方法和查询技巧，完成了一个比较复杂的 XML 查询系统的功能设计。我们可以在自己的程序中实现并对照给出的结果集进行验证。

1. 实例引用文档及数据类型定义

（1）本节中大多数查询实例都是基于名为 http://www.bn.com/bib.xml 的书目文档。其 DTD 如下：

```
<!ELEMENT bib    (book* )>
<!ELEMENT book    (title,    (author+ | editor+ ), publisher, price )>
<!ATTLIST book    year CDATA    #REQUIRED >
<!ELEMENT author    (last, first )>
<!ELEMENT editor    (last, first, affiliation )>
<!ELEMENT title    (#PCDATA )>
<!ELEMENT last    (#PCDATA )>
<!ELEMENT first    (#PCDATA )>
<!ELEMENT affiliation    (#PCDATA )>
<!ELEMENT publisher    (#PCDATA )>
<!ELEMENT price    (#PCDATA )>
```

下面是在 www.bn.com/bib.xml 中的数据：

```
<bib>
    <book year="1994">
        <title>TCP/IP Illustrated</title>
        <author><last>Stevens</last><first>W.</first></author>
        <publisher>Addison-Wesley</publisher>
        <price> 65.95</price>
    </book>
```

```
<book year="1992">
    <title>Advanced Programming in the Unix environment</title>
    <author><last>Stevens</last><first>W.</first></author>
    <publisher>Addison-Wesley</publisher>
    <price>65.95</price>
</book>

<book year="2000">
    <title>Data on the Web</title>
    <author><last>Abiteboul</last><first>Serge</first></author>
    <author><last>Buneman</last><first>Peter</first></author>
    <author><last>Suciu</last><first>Dan</first></author>
    <publisher>Morgan Kaufmann Publishers</publisher>
    <price> 39.95</price>
</book>

<book year="1999">
    <title>The Economics of Technology and Content for Digital TV</title>
    <editor>
            <last>Gerbarg</last><first>Darcy</first>
             <affiliation>CITI</affiliation>
    </editor>
        <publisher>Kluwer Academic Publishers</publisher>
    <price>129.95</price>
</book>

</bib>
```

（2）查询 Q5 使用名为 http://www.amazon.com/reviews.xml 的书籍评论和价格信息，其 DTD 如下：

```
<!ELEMENT reviews (entry*)>
<!ELEMENT entry     (title, price, review)>
<!ELEMENT title     (#PCDATA)>
<!ELEMENT price     (#PCDATA)>
<!ELEMENT review    (#PCDATA)>
```

下面是 Q5 的数据文档"http://www.amazon.com/reviews.xml":

```
<reviews>
    <entry>
        <title> Data on the Web</title>
        <price>34.95</price>
        <review>
                A very good discussion of semi-structured database
                systems and XML.
        </review>
    </entry>

    <entry>
```

```
        <title> Advanced Programming in the Unix environment</title>
        <price>65.95</price>
        <review>
                A clear and detailed discussion of UNIX programming.
        </review>
    </entry>

    <entry>
        <title>TCP/IP Illustrated</title>
        <price>65.95</price>
        <review>
                One of the best books on TCP/IP.
        </review>
    </entry>

</reviews>
```

（3）查询 Q8 使用名为 books.xml 的输入文档，其 DTD 如下:

```
<!ELEMENT chapter (title, section*)>
<!ELEMENT section (title, section*)>
<!ELEMENT title (#PCDATA)>
```

下面是 books.xml 的数据

```
<chapter>
    <title>Data Model</title>
    <section>
        <title> Syntax For Data Model</title>
    </section>

    <section>
        <title>XML</title>
        <section>
            <title>Basic Syntax</title>
        </section>
        <section>
                <title> XML and Semistructured Data</title>
        </section>
    </section>
</chapter>
```

（4）查询 Q9 使用名为 prices.xml 的输入文档，其 DTD 如下:

```
<!ELEMENT prices (book*)>
<!ELEMENT book (title, source, price)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT source (#PCDATA)>
<!ELEMENT price (#PCDATA)>
```

下面是 prices.xml 的数据：

```
<prices>
    <book>
        <title>Advanced Programming in the Unix environment</title>
        <source>www.amazon.com</source>
        <price>65.95</price>
    </book>

    <book>
        <title>Advanced Programming in the Unix environment </title>
        <source>www.bn.com</source>
        <price>65.95</price>
    </book>

    <book>
        <title> TCP/IP Illustrated </title>
        <source>www.amazon.com</source>
        <price>65.95</price>
    </book>

    <book>
        <title> TCP/IP Illustrated </title>
        <source>www.bn.com</source>
        <price>65.95</price>
    </book>

    <book>
        <title>Data on the Web</title>
        <source>www.amazon.com</source>
        <price>34.95</price>
    </book>

    <book>
        <title>Data on the Web</title>
        <source>www.bn.com</source>
        <price>39.95</price>
    </book>
</prices>
```

2. 查询及结果集

（1）Q1。列出 Addison Wesley1991 年后出版的书籍的出版时间和书名。

```
<bib>
    <book year="1994">
        <title>TCP/IP Illustrated</title>
    </book>
    <book year="1992">
        <title>Advanced Programming in the Unix environment</title>
```

```
    </book>
</bib>
```

（2）Q2。创建 result 元素存储 title-author 数据：

```
<results>
        <result>
                <title>TCP/IP Illustrated</title>
                <author><last>Stevens</last><first>W.</first></author>
        </result>
        <result>
                <title>Advanced Programming in the Unix environment</title>
                <author><last>Stevens</last><first>W.</first></author>
        </result>
        <result>
                <title>Data on the Web</title>
                <author><last>Abiteboul</last><first>Serge</first></author>
        </result>
        <result>
                <title> Data on the Web</title>
                <author><last>Buneman</last><first>Peter</first></author>
        </result>
        <result>
                <title>Data on the Web</title>
                <author><last>Suciu</last><first>Dan</first></author>
        </result>
</results>
```

（3）Q3。列出书目集中每本书的 title 和 authors，聚合存放在“result”元素中。

```
<results>
    <result>
        <title>TCP/IP Illustrated</title>
        <author><last>Stevens</last><first>W.</first></author>
    </result>

    <result>
        <title>Advanced Programming in the Unix environment</title>
        <author><last>Stevens</last><first>W.</first></author>
    </result>

    <result>
        <title>Data on the Web</title>
        <author><last>Abiteboul</last><first>Serge</first></author>
        <author><last>Buneman</last><first>Peter</first></author>
        <author><last>Suciu</last><first>Dan</first></author>
    </result>
</results>
```

（4）Q4。列出书目集中每个作者的名字及其所写的所有书的书名，聚合存放在 result 元素中。

```
<results>
    <result>
        <author><last>Stevens</last><first>W.</first></author>
        <title>TCP/IP Illustrated</title>
        <title>Advanced Programming in the Unix environment</title>
    </result>

    <result>
        <author><last>Abiteboul</last><first>Serge</first></author>
        <title>Data on the Web</title>
    </result>

    <result>
        <author><last>Buneman</last><first>Peter</first></author>
        <title>Data on the Web</title>
    </result>

    <result>
        <author><last>Suciu</last><first>Dan</first></author>
          <title>Data on the Web</title>
    </result>
</results>
```

（5）Q5。列出在 bn.com 和 amazon.com 都能找到的书的书名，并标明不同网站上的价格。

```
<books-with-prices>
    <book-with-prices>
      <title>TCP/IP Illustrated</title>
      <price-amazon>65.95</price-amazon>
      <price-bn>65.95</price-bn>
    </book-with-prices>

    <book-with-prices>
      <title>Advanced Programming in the Unix environment</title>
      <price-amazon>65.95</price-amazon>
      <price-bn>65.95</price-bn>
    </book-with-prices>

    <book-with-prices>
      <title>Data on the Web</title>
      <price-amazon>34.95</price-amazon>
      <price-bn>39.95</price-bn>
    </book-with-prices>
</books-with-prices>
```

（6）Q6。列出每本书的书名和前两名作者，如果有超过两名的作者，用 et-al 空元素标记。

```
<bib>
    <book>
        <title>TCP/IP Illustrated</title>
```

```
        <author><last>Stevens</last><first>W.</first></author>
    </book>

    <book>
        <title>Advanced Programming in the Unix environment</title>
        <author><last>Stevens</last><first>W.</first></author>
    </book>

    <book>
        <title>Data on the Web</title>
            <author><last>Abiteboul</last><first> Serge</first></author>
            <author><last>Buneman</last><first>Peter</first></author>
            <et-al/>
    </book>
</bib>
```

（7）Q7。按字母表顺序列出 Addison Wesley1991 年后出版的书籍的出版时间和书名。

```
<bib>
    <book year="1992">
        <title>Advanced Programming in the Unix environment</title>
    </book>
    <book year="1994">
        <title>TCP/IP Illustrated</title>
   </book>
</bib>
```

（8）Q8。在 books.xml 文档中查找所有包含 XML 的部分和章节，不管它的存放层次。

```
<results>
        <title>XML</title>
        <title>XML and Semistructured Data</title>
</results>
```

（9）Q9。在 prices.xml 文档中找出每本书的最低价格，将书名和相关属性存放在 minprice 元素中。

```
<results>
    <minprice title="Advanced Programming in the Unix environment"> 65.95 </minprice>
    <minprice title="TCP/IP Illustrated"> 65.95 </minprice>
    <minprice title="Data on the Web"> 34.95 </minprice>
</results>
```

（10）Q10。对于有作者的书，列出其书名和作者；对于有编辑人员的书，列出 reference（包括书名和编辑人员的从属关系）。

```
<bib>
        <book>
            <title>TCP/IP Illustrated</title>
            <author><last> Stevens </last> <first> W.</first></author>
        </book>

        <book>
```

```
        <title>Advanced Programming in the Unix environment</title>
        <author><last>Stevens</last><first>W.</first></author>
    </book>

    <book>
        <title>Data on the Web</title>
        <author><last>Abiteboul</last><first>Serge</first></author>
        <author><last>Buneman</last><first>Peter</first></author>
        <author><last>Suciu</last><first>Dan</first></author>
    </book>

    <reference>
        <title>The Economics of Technology and Content for Digital TV</title>
        <org>CITI</org>
    </reference>
</bib>
```

（11）Q11。查找相同作者（可是多人）所著的不同的书，列出书名。

```
<bib>
    <book-pair>
        <title> TCP/IP Illustrated </title>
        <title> Advanced Programming in the Unix environment </title>
    </book-pair>
</bib>
```

14.4 小　　结

通过本章的学习，读者可以明确 XML 在数据库领域的巨大作用和光明前景；掌握用 XML 来解决数据抽取、传送、合并、查询、建模等一系列数据库问题；对基于 XML 的结构化和半结构化数据进行全面而广泛的设计应用。

第十五章　XML 中的矢量图形处理技术

本章着重介绍基于 XML 矢量图形处理技术。通过 XML 来实现网络矢量图形处理可以满足大画面、低数据量、无级缩放等需求。该技术以符合 XML 规范的文本存放构成图形的直线、基本形状、曲线以及颜色的数学描述，图形文件相当简洁，而且扩展性很好，并可以任何分辨率显示。因此基于 XML 的矢量图形能在保证良好视觉效果的前提下，最大限度缩小文件的体积，以获得更短的下载时间和更灵活的显示能力。SVG 和 VML 作为两种有很大希望成为主流应用的技术在本章将重点讨论。

本章包括以下内容：

- 可伸缩的矢量图形 SVG
- 矢量标记语言 VML

15.1　可伸缩的矢量图形 SVG

15.1.1　概述

用 HTML 创建一个精美的网站往往是困难的，网站开发和管理人员在自己的开发和应用环境下布置好的网页在用户的显示器上看上去非常凌乱，或者因为小小地调整一下浏览窗口而使网页面目全非，不同的操作系统、不同的浏览器、不同的字体、不同的屏幕分辨率，使得在 Web 上创作精确定位的图形界面难上加难，网站开发和管理人员不得不在每一个网页上都写上“建议采用 XXX 浏览器，在 XXX 分辨率下观看”。然而 HTML 在图形方面的欠佳表现将被 SVG 彻底地改观。

从浏览器的核心技术来看，其对于图形图像的支持还仅仅局限于对图像的简单显示，随着应用的逐渐深入，图像技术自身的一些缺点日益突出，从某种程度上讲也限制了 Web 浏览技术的进一步发展。矢量图形提供了比当前在 HTML 里标准化的图像格式更好的东西，包括表示简洁和可以任意缩放，减少了终端用户的下载时间。为此，众多业内人士针对 Web 浏览器图像图形功能支持较弱的缺点提出了改进措施——SVG。

SVG 的全称是可伸缩矢量图形（Scalable Vector Graphics），它是 W3C 组织为适应 Internet Web 应用飞速发展的需要而制定的一套基于 XML 语言的可缩放矢量图形语言描述规范。近年来，HTML 作为在 Internet/Intranet 网上进行数据浏览和数据交换的一种先进的文件格式规范，以浏览器为依托凭借着其丰富而强大的功能打破了传统的数据浏览方式，给人以耳目一新的感觉，也极大地推动了 Internet 的发展。规范化的 HTML 4.0 语言定义了众多的标记（Tag），支持链接、表格、图像、窗体、帧、层叠样式表（CSS），并允许内嵌 JavaScript、VBScript、ActiveX、Plug-in 以及声音、动画等。但是，随着 HTML 应用的不断深入，其不足之处也逐渐暴露出来，主要存在于两点：一是采用的标记固定、有限且无内涵；二是不支持矢量图形。这两大缺陷越来越成为限制 Web 应用的障碍，作为一种改进，W3C 于 1998 年 2 月 10 日发布了 XML 1.0

规范，并于 2000 年 10 月 6 日发布了 XML1.0 标准的 Second Edition（第二版）。XML 以其元标记的特性解决了 HTML 在标记上的不足，使得 Internet 技术大大前进了一步，但它仍不支持矢量图形。与此同时，各大软件厂商和组织纷纷推出自己的图形规范，主要代表有 Adobe Systems Inc.制定的 PGML（Precision Graphics Markup Language）、CCLRC 提出的 Web Schematics、Autodesk Inc 和微软等提出的 VML（Vector Markup Language），正是在这种情况下，为统一标准，结束目前的混乱局面，W3C 组织于 1998 年 8 月专门成立了 SVG 工作组致力于图形标准的制定工作，并于 1999 年 2 月 11 日发布了第一个讨论草案，后几经修订于 1999 年 8 月 12 日发布最终草案。

SVG 采用的是像 Macromedia 的 Flash 技术一样的矢量图形格式，矢量图形最大的特点是图像可以任意放大或缩小而不损失任何细节。然而，SVG 所带来的不只是这些，它可以对图形元素精确定位、内嵌字体、增加防锯齿功能，以及添加各种效果滤镜，Web 图形制作高手再也不用把他们精心设计好的作品存成 JPEG 或 PNG 等点阵格式，不用再担心分辨率的不同，缩放自如的 SVG 可以保证它们在客户端绘制出来之后都是相同的效果，它给 Web 上的图形设计带来的将是一场革命。

SVG 并非仅仅是一种图像格式，由于它是一种基于 XML 的语言，也就意味着它继承了 XML 的跨平台性和可扩展性。SVG 可以内嵌于其他的 XML 文档中，而 SVG 文档中也可以嵌入其他的 XML 内容，各个不同的 SVG 图形可以方便地组合，构成新的 SVG 图形，从而在图形可重用性上迈出了一大步。SVG 同样支持使用层叠样式表 CSS 或可扩展样式语言 XSL 来定义它们的样式，比如填充颜色和线型等，只需要通过修改几个 CSS 就可以用同一个 SVG 文档创造出各种不同的表现。

SVG 具有设计完善的 DOM 接口，使各种编程语言和脚本语言可以方便地对它的每一个元素进行操作。SVG 内置了对于 JavaScript 的支持，利用 JavaScript，可以为 SVG 添加引人入胜的交互程序，比如将某个图形元素移动或变换，而不是整幅图像的切换，这些将会为动态网站设计和网络游戏的发展提供更加有利的工具。

SVG 内的文字都以文本的形式出现在 XML 文件中，这些信息可以为搜索引擎所用，而以往搜索引擎通常无法搜索到写在点阵图像中的文字。这些文本信息还可以帮助视力有残疾而无法看到图形的人，可以通过其他方式（如声音）来传送这些信息。

目前，SVG 的最新文本是第 7 版工作草案，SVG 的制订工作受到各大公司如 IBM, Microsoft, Apple, Xerox, Sun Microsystems, Hewlett-Packard, Netscape, Corel, Adobe, Quark, Macromedia 等的支持，Internet Explorer 和 Netscape 都将在今后的版本中提供对于 SVG 的支持，Corel 公司的 CorelDraw 10，Adobe 公司的 Illustrator 9 都正在为提供 SVG 的输出插件做 Beta 测试工作，大多数开发者将可以仍旧使用他们熟悉的工具来创作 SVG，当然，也可以只用文本编辑器，就像当初写 HTML 一样。

15.1.2 SVG 标准与规范

SVG 规范定义了 SVG 的特征和语法，包括模块化的 XML 名字空间（namespace）和 SVG 文档对象模型（DOM）。

SVG 是一种用于 XML 的用来描述二维矢量图形和矢量/点阵混合图形的语言，它提供了 4 种类型的图形对象：矢量图形（vector graphic shape）、图像（image）、渐变填充（gradient filling）、

文本（text），图形对象还可进行分组、添加样式、变换、组合等操作，特征集包括嵌套变换（nested transformations）、剪切路径（clipping paths）、alpha 蒙板（alpha masks）、过滤器效果（filter effects）、模板对象（template objects）和其它扩展（extensibility）。

SVG 的绘图可以通过动态和交互式方式进行，在实际操作中，则是以嵌入方式或脚本方式来实现的。对脚本语言的支持，使得高级用户仅仅进行简单的 Scripts 编程，访问 SVG DOM 的元素和属性，响应特定的事件，从而提高了 SVG 的动态和交互性能。

1. SVG 优点

由于 SVG 具备许多特点，使得它有一些独特的优点，特别是由于矢量技术的引入。

- 跨平台：同 XML 规范的无缝连接以及标记语言的平台无关性，赋予了 SVG 跨平台的优点。
- 集成 SVG 的 XML 文档将更小、下载浏览的速度更快，并具有更强的交互性。矢量图形采用点和线的描述来绘制图形，而不必再像过去那样用庞大的点阵图像（目前的浏览器都不支持图形，因此即使某些 Web 制作工具，如 NetObjects Fusion，声称支持图形，其最终也是将图形转换成图像，最多不过是将图形保存在中间文件中）。矢量指令的简单高效，使得集成了 SVG 的 XML 文档大小进一步缩小，由于代替了图像，因而下载速度也会大大提高，也正是基于此，SVG 可以作为解决目前网上浏览瓶颈的最佳方案。
- 更广泛的硬件支持：从较低分辨率的便携式计算机到较高分辨率的台式机再到高分辨率的打印机，这将大大提高 Web 应用。SVG 图像在屏幕上总是边缘清晰，并且可以使用用户打印机的分辨率进行打印。不论是 300 dpi，600 dpi 还是更高，用户都不会体验到难看的锯齿的点阵效果。这也是由于矢量技术所带来的技术上的突破。
- 支持图形的无极缩放：矢量技术使得当将图形按任意比例缩放时保持图形显示的高效，可以任意放大图形显示，但不会牺牲锐利度、清晰度、细节等。这一点是目前的点阵式图像所无法比拟的。
- 交互性和智能化：因为 SVG 是基于 XML 的，它提供无可匹敌的动态交互性。SVG 图像可对用户的动作通过高光显示、工具技巧、特殊效果、声音和动画进行反应和显示。
- 方便的图形定位与检索：通过同 XML 灵活的标记特性相结合，可以方便地进行图形的定位与检索。

2. SVG 对图形图像的特殊支持

SVG 除了支持 HTML 中常用的标记，如文本、图像、链接、交互性、CSS 的使用、脚本（Script）外，还提供了大量针对图形、图像、动画的特定标记等。

（1）SVG 中对矢量图形的支持。众所周知，矢量图形用点和线来描述，可以大大减小文件长度，提高传输效率。更重要的是，它将对图形效果的显示由服务器端移到客户端，可以充分利用客户端的资源，减轻服务器端的负担。

SVG 中有专门用于矢量图形描述的标记，包括矩形<rect>、圆<circle>、椭圆<ellipse>、直线<line>、折线<polyline>和多边形<polygon>。此外，SVG 还支持图形绘制中的常用的、由 Bezier

曲线定义的路径描述和操作，其元素标记为<path>。有了以上定义，就可对相应路径进行勾勒、填充、裁剪、蒙版和合成等一系列操作。

对于<path>、<text>元素和前述形状元素，可以利用 Stroke 和 Fill 属性进行勾勒和填充，其中填充可以使用某种颜色，或使用<lineargradient>、<radialgradient>定义的渐变色，或是使用<pattern>定义的底纹填充模式。而对于<path>、<line>、<polyline>、<polygon>等元素，只要将其放入<marker>标记对中，即可按所定义的路径绘制箭头。

同样，对于<path>、<text>元素和各种形状元素，也可以对其轮廓内的区域作裁剪、蒙版和合成操作。<clippath>利用上述各种元素描述裁剪路径，<mask>标记则支持单通道、三通道和 alpha 通道操作。

SVG 还支持图形中成组的概念，以上操作均可以对一组图形进行操作。

（2）SVG 中对图像过滤操作的支持。目前网上传输的图像主要采用 GIF、JPEG 和 PNG 三种格式。尽管它们具有高压缩比、低容量的优点，但即便要将其做一点微小改动，也必须利用图像软件将其重新制作重新存储，非常烦琐。SVG 支持对于图像的一系列常用过滤器操作，使得图像效果调整的任务可以在客户端进行。

使用标记<filter>可以定义过滤器效果，在其中按照要施加的过滤操作的顺序依次罗列相应的标记。例如，要定义一个阴影过滤操作，在<filter></filter>标记对中应依次写入<fegaussianblur>——高斯滤波、<feoffset>——平移、<fediffuselighting>——扩散和<fecomposite>——合成。

3. SVG 中对动画的支持

目前，网页中播放的动画多为 GIF 格式，它也存在着与网上传输的图像格式相同的问题，即修改在服务器端实现，而不是在客户端实现。SVG 中提供了专门的动画元素，可以描述一个图形图像元素的实时变化。

SVG 中用标记<animatemotion>描述元素的移动效果，用<animateflipbook>描述元素的弹跳效果，用<animatetransform>描述元素的放缩、旋转、偏斜等变换效果，用<animatecolor>描述元素颜色的改变，还可以用<animate>描述元素的淡入淡出效果。

4. SVG 基本图形元素

SVG 包括下面的基本图形元素：

- rectangles（矩形，包括可选择的圆角）
- circles（圆）
- ellipses（椭圆）
- lines（线段）
- polylines（折线）
- polygons（多边形）

（1）矩形（rect 元素）。Rect 元素根据用户的作图坐标定义一个矩形，圆角可以通过设定 rx、ry 属性而实现。其定义如下：

```
<!ENTITY % rectExt "" >
<!ELEMENT rect (%descTitleMetadata;,(animate|set|animateMotion|animateColor|animateTransform
        %geExt;%rectExt;)*) >
```

```
<!ATTLIST rect
   %stdAttrs;
   %testAttrs;
   %langSpaceAttrs;
   externalResourcesRequired %Boolean; #IMPLIED
   class %ClassList; #IMPLIED
   style %StyleSheet; #IMPLIED
   %PresentationAttributes-FillStroke;
   %PresentationAttributes-Graphics;
   transform %TransformList; #IMPLIED
   %graphicsElementEvents;
   x %Coordinate; #IMPLIED
   y %Coordinate; #IMPLIED
   width %Length; #REQUIRED
   height %Length; #REQUIRED
   rx %Length; #IMPLIED
   ry %Length; #IMPLIED >
```

例 15.1

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000802//EN"
   "http://www.w3.org/TR/2000/CR-SVG-20000802/DTD/svg-20000802.dtd">
<svg width="12cm" height="4cm">
   <desc>Example rect01 - rectangle expressed in physical units</desc>

   <rect x="4cm" y="1cm" width="4cm" height="2cm"
         style="fill:yellow; stroke:navy; stroke-width:0.1cm" />
</svg>
```

在支持 SVG 的浏览器中的显示如图 15-1。

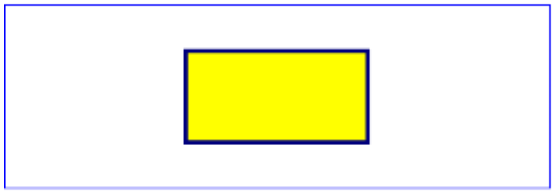


图 15-1

例 15.2

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000802//EN"
   "http://www.w3.org/TR/2000/CR-SVG-20000802/DTD/svg-20000802.dtd">
<svg width="12cm" height="4cm" viewBox="0 0 1200 400">
   <desc>Example rect02 - rounded rectangles expressed in user coordinates</desc>

   <rect x="1" y="1" width="1198" height="398"
         style="fill:none; stroke:blue"/>
```

```
  <rect x="100" y="100" width="400" height="200" rx="50"
        style="fill:green;" />

  <g transform="translate(700 210) rotate(-30)">
    <rect x="0" y="0" width="400" height="200" rx="50"
          style="fill:none; stroke:purple; stroke-width:30" />
  </g>
</svg>
```

在支持 SVG 的浏览器中的显示如图 15-2。

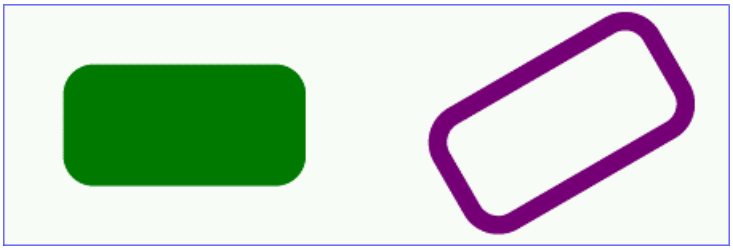


图 15-2

（2）圆（circle 元素）。Circle 元素根据用户定义的圆心和半径作一个圆。其定义如下：

```
<!ENTITY % circleExt "" >
<!ELEMENT circle (%descTitleMetadata;,(animate|set|animateMotion|animateColor|animateTransform
                  %geExt;%circleExt;)*) >
<!ATTLIST circle
  %stdAttrs;
  %testAttrs;
  %langSpaceAttrs;
  externalResourcesRequired %Boolean; #IMPLIED
  class %ClassList; #IMPLIED
  style %StyleSheet; #IMPLIED
  %PresentationAttributes-FillStroke;
  %PresentationAttributes-Graphics;
  transform %TransformList; #IMPLIED
  %graphicsElementEvents;
  cx %Coordinate; #IMPLIED
  cy %Coordinate; #IMPLIED
  r %Length; #REQUIRED >
```

例 15.3

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000802//EN"
  "http://www.w3.org/TR/2000/CR-SVG-20000802/DTD/svg-20000802.dtd">
<svg width="12cm" height="4cm">
  <desc>Example circle01 - circle expressed in physical units</desc>

  <circle cx="6cm" cy="2cm" r="1cm"
          style="fill:red; stroke:blue; stroke-width:0.1cm" />
```

</svg>

在支持 SVG 的浏览器中的显示如图 15-3。

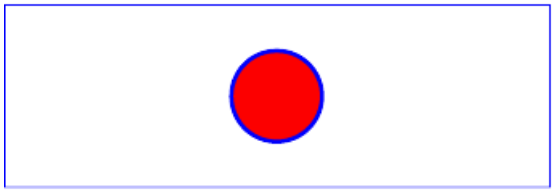


图 15-3

（3）椭圆（ellipse 元素）。Ellipse 元素根据用户定义的中心点和长短轴作一个椭圆。其定义如下：

```
<!ENTITY % ellipseExt "" >
<!ELEMENT ellipse (%descTitleMetadata;,(animate|set|animateMotion|animateColor|animateTransform
                    %geExt;%ellipseExt;)*) >
<!ATTLIST ellipse
  %stdAttrs;
  %testAttrs;
  %langSpaceAttrs;
  externalResourcesRequired %Boolean; #IMPLIED
  class %ClassList; #IMPLIED
  style %StyleSheet; #IMPLIED
  %PresentationAttributes-FillStroke;
  %PresentationAttributes-Graphics;
  transform %TransformList; #IMPLIED
  %graphicsElementEvents;
  cx %Coordinate; #IMPLIED
  cy %Coordinate; #IMPLIED
  rx %Length; #REQUIRED
  ry %Length; #REQUIRED >
```

例 15.4

```
    <?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000802//EN"
  "http://www.w3.org/TR/2000/CR-SVG-20000802/DTD/svg-20000802.dtd">
<svg width="12cm" height="4cm" viewBox="0 0 1200 400">
  <desc>Example ellipse01 - ellipses expressed in user coordinates</desc>

  <g transform="translate(300 200)">
    <ellipse rx="250" ry="100"
          style="fill:red" />
  </g>

  <ellipse transform="translate(900 200) rotate(-30)"
        rx="250" ry="100"
        style="fill:none; stroke:blue; stroke-width: 20" />
```

</svg>

在支持 SVG 的浏览器中的显示如图 15-4。

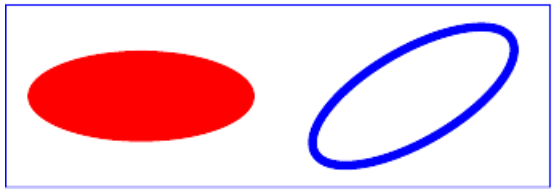


图 15-4

（4）线段（line 元素）。Line 元素定义一条起于一点终于另一点的线段，其定义如下：

```
<!ENTITY % lineExt "" >
<!ELEMENT line (%descTitleMetadata;,(animate|set|animateMotion|animateColor|animateTransform
                %geExt;%lineExt;)*) >
<!ATTLIST line
   %stdAttrs;
   %testAttrs;
   %langSpaceAttrs;
   externalResourcesRequired %Boolean; #IMPLIED
   class %ClassList; #IMPLIED
   style %StyleSheet; #IMPLIED
   %PresentationAttributes-FillStroke;
   %PresentationAttributes-Graphics;
   %PresentationAttributes-Markers;
   transform %TransformList; #IMPLIED
   %graphicsElementEvents;
   x1 %Coordinate; #IMPLIED
   y1 %Coordinate; #IMPLIED
   x2 %Coordinate; #IMPLIED
   y2 %Coordinate; #IMPLIED >
```

例 15.5

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000802//EN"
   "http://www.w3.org/TR/2000/CR-SVG-20000802/DTD/svg-20000802.dtd">
<svg width="12cm" height="4cm" viewBox="0 0 1200 400">
   <desc>Example line01 - lines expressed in user coordinates</desc>

   <g style="fill:none; stroke:green">
      <line x1="100" y1="300" x2="300" y2="100"
            style="stroke-width:5" />
      <line x1="300" y1="300" x2="500" y2="100"
            style="stroke-width:10" />
      <line x1="500" y1="300" x2="700" y2="100"
            style="stroke-width:15" />
      <line x1="700" y1="300" x2="900" y2="100"
            style="stroke-width:20" />
```

```
    <line x1="900" y1="300" x2="1100" y2="100"
          style="stroke-width:25" />
  </g>
</svg>
```

在支持 SVG 的浏览器中的显示如图 15-5。

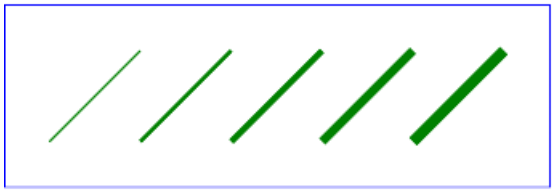


图 15-5

（5）折线（polyline 元素）。Polyline 元素表示一组相交的直线段，通常 Polyline 元素定义开放性的图形。其定义如下：

```
    <!ENTITY % polylineExt "" >
<!ELEMENT polyline (%descTitleMetadata;,(animate|set|animateMotion|animateColor|animateTransform
                %geExt;%polylineExt;)*) >
<!ATTLIST polyline
  %stdAttrs;
  %testAttrs;
  %langSpaceAttrs;
  externalResourcesRequired %Boolean; #IMPLIED
  class %ClassList; #IMPLIED
  style %StyleSheet; #IMPLIED
  %PresentationAttributes-FillStroke;
  %PresentationAttributes-Graphics;
  %PresentationAttributes-Markers;
  transform %TransformList; #IMPLIED
  %graphicsElementEvents;
  points %Points; #REQUIRED >
```

例 15.6

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000802//EN"
  "http://www.w3.org/TR/2000/CR-SVG-20000802/DTD/svg-20000802.dtd">
<svg width="12cm" height="4cm" viewBox="0 0 1200 400">
  <desc>Example polyline01 - increasingly larger bars</desc>

  <rect x="1" y="1" width="1198" height="398"
        style="fill:none; stroke:blue"/>
  <polyline style="fill:none; stroke:blue; stroke-width:10"
            points="50,375
                    150,375 150,325 250,325 250,375
                    350,375 350,250 450,250 450,375
                    550,375 550,175 650,175 650,375
                    750,375 750,100 850,100 850,375
```

```
                950,375 950,25 1050,25 1050,375
                1150,375" />
</svg>
```

在支持 SVG 的浏览器中的显示如图 15-6。

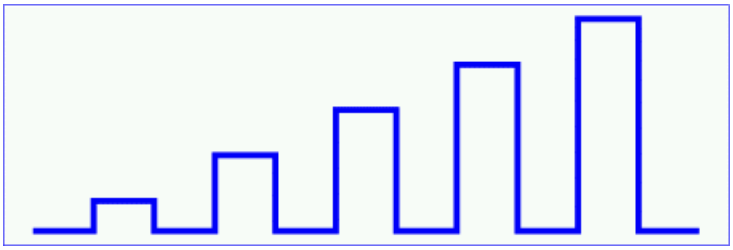


图 15-6

（6）多边形（polygon 元素）。Polygon 元素定义由一组相交的直线段构成的闭合图形，其定义如下：

```
<!ENTITY % polygonExt "" >
<!ELEMENT polygon (%descTitleMetadata;,(animate|set|animateMotion|animateColor|animateTransform
                %geExt;%polygonExt;)*) >
<!ATTLIST polygon
  %stdAttrs;
  %testAttrs;
  %langSpaceAttrs;
  externalResourcesRequired %Boolean; #IMPLIED
  class %ClassList; #IMPLIED
  style %StyleSheet; #IMPLIED
  %PresentationAttributes-FillStroke;
  %PresentationAttributes-Graphics;
  %PresentationAttributes-Markers;
  transform %TransformList; #IMPLIED
  %graphicsElementEvents;
  points %Points; #REQUIRED >
```

例 15.7

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000802//EN"
  "http://www.w3.org/TR/2000/CR-SVG-20000802/DTD/svg-20000802.dtd">
<svg width="12cm" height="4cm" viewBox="0 0 1200 400">
  <desc>Example polygon01 - star and hexagon</desc>

  <polygon style="fill:red; stroke:blue; stroke-width:10"
          points="350,75   379,161 469,161 397,215
                  423,301 350,250 277,301 303,215
                  231,161 321,161" />
  <polygon style="fill:lime; stroke:blue; stroke-width:10"
          points="850,75   958,137.5 958,262.5
                  850,325 742,262.6 742,137.5" />
```

```
</svg>
```

在支持 SVG 的浏览器中的显示如图 15-7。

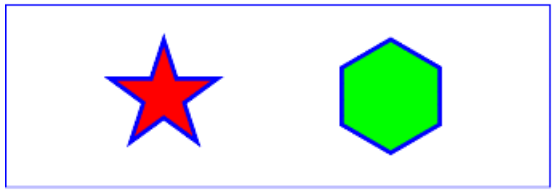


图 15-7

当然 SVG 还有相当多的特性和元素，这里我们不一一详述了，有兴趣的读者可以参阅相关文档和 WWW.W3.ORG 的 SVG 规范草案。

15.1.3 SVG 应用实例

1. SVG 基本应用

下面是 SVG 同 XML 语法相结合的简单例子，在该示例中，使用了 SVG 草案中的一些典型标记。本实例演示了如何用 SVG 规范绘制一个经过梯度变换填充的矩形图形。

```
<?xml version="1.0"?>

<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG April 1999//EN"
"http://www.w3.org/Graphics/SVG/svg-19990412.dtd">
<svg width="1.5in" height="1.0in">

 <title>Sample SVG graphic</title>

 <desc>
 A single rectangle with an orange to red sunburst fill
 </desc>

 <g>
  <radialgradient id="burst">
  <gradientstop offset="0" color="#FF3"/>
  <gradientstop offset="1" color="#C33"/>
  </radialgradient>
  <rect x="2" y="2" height="20" width="33" style="fill:url(#burst); stroke:#FEFEFE;"/>
 </g>

</svg>
```

2. 用图形饰点一个矩形

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000802//EN"
   "http://www.w3.org/TR/2000/CR-SVG-20000802/DTD/svg-20000802.dtd">
<svg width="8cm" height="3cm">
   <desc>Example mask01 - blue text masked with gradient against red background
```

```
  </desc>
  <defs>
    <linearGradient id="Gradient" x1="0cm" y1="0cm" x2="8cm" y2="0cm">
      <stop offset="0" style="stop-color:black; stop-opacity:0"/>
      <stop offset="1" style="stop-color:black; stop-opacity:1"/>
    </linearGradient>
    <mask id="Mask">
      <rect x="0cm" y="0cm" width="8cm" height="3cm" style="fill:url(#Gradient)" />
    </mask>
    <text id="Text" x="4cm" y="2cm"
          style="font-family:Verdana; font-size:1cm; text-anchor:middle">
      Masked text
    </text>
  </defs>

  <!-- Draw a pale red rectangle in the background -->
  <rect x="0cm" y="0cm" width="8cm" height="3cm" style="fill:#FF8080"/>

  <!-- Draw the text string twice. First, filled blue, with the mask applied.
       Second, outlined in black without the mask. -->
  <use xlink:href="#Text" style="fill:blue; mask:url(#Mask)"/>
  <use xlink:href="#Text" style="fill:none; stroke:black; stroke-width:.02cm"/>
</svg>
```

在支持 SVG 的浏览器中的显示如图 15-8。

图 15-8

3. 演示“opacity”属性效果

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000802//EN"
  "http://www.w3.org/TR/2000/CR-SVG-20000802/DTD/svg-20000802.dtd">
<svg width="12cm" height="3.5cm" viewBox="0 0 1200 350">
  <desc>Example opacity01 - opacity property</desc>

  <rect x="1" y="1" width="1198" height="348"
        style="fill:none; stroke:blue"/>

  <!-- Background blue rectangle -->
  <rect x="1cm" y="1cm" width="10cm" height="1.5cm" style="fill:#0000ff" />
```

```
  <!-- Red circles going from opaque to nearly transparent -->
  <circle cx="2cm" cy="1cm" r=".5cm" style="fill:red; opacity:1" />
  <circle cx="4cm" cy="1cm" r=".5cm" style="fill:red; opacity:.8" />
  <circle cx="6cm" cy="1cm" r=".5cm" style="fill:red; opacity:.6" />
  <circle cx="8cm" cy="1cm" r=".5cm" style="fill:red; opacity:.4" />
  <circle cx="10cm" cy="1cm" r=".5cm" style="fill:red; opacity:.2" />

  <!-- Opaque group, opaque circles -->
  <g style="opacity:1">
    <circle cx="1.825cm" cy="2.5cm" r=".5cm" style="fill:red; opacity:1" />
    <circle cx="2.175cm" cy="2.5cm" r=".5cm" style="fill:green; opacity:1" />
  </g>
  <!-- Group opacity: .5, opacity circles -->
  <g style="opacity:.5">
    <circle cx="3.825cm" cy="2.5cm" r=".5cm" style="fill:red; opacity:1" />
    <circle cx="4.175cm" cy="2.5cm" r=".5cm" style="fill:green; opacity:1" />
  </g>
  <!-- Opaque group, semi-transparent green over red -->
  <g style="opacity:1">
    <circle cx="5.825cm" cy="2.5cm" r=".5cm" style="fill:red; opacity:.5" />
    <circle cx="6.175cm" cy="2.5cm" r=".5cm" style="fill:green; opacity:.5" />
  </g>
  <!-- Opaque group, semi-transparent red over green -->
  <g style="opacity:1">
    <circle cx="8.175cm" cy="2.5cm" r=".5cm" style="fill:green; opacity:.5" />
    <circle cx="7.825cm" cy="2.5cm" r=".5cm" style="fill:red; opacity:.5" />
  </g>
  <!-- Group opacity .5, semi-transparent green over red -->
  <g style="opacity:.5">
    <circle cx="9.825cm" cy="2.5cm" r=".5cm" style="fill:red; opacity:.5" />
    <circle cx="10.175cm" cy="2.5cm" r=".5cm" style="fill:green; opacity:.5" />
  </g>
</svg>
```

在支持 SVG 的浏览器中的显示如图 15-9。

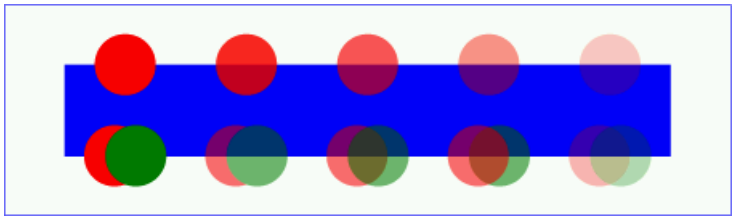


图 15-9

4. SVG 动画元素应用演示（Animation elements example）

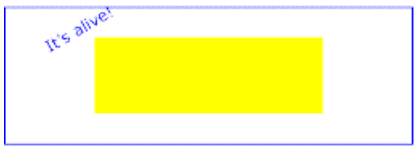
```
<?xml version="1.0" standalone="no"?>
```

```
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000802//EN"
   "http://www.w3.org/TR/2000/CR-SVG-20000802/DTD/svg-20000802.dtd">
<svg width="8cm" height="3cm"    viewBox="0 0 800 300">
   <desc>Example anim01 - demonstrate animation elements</desc>

   <!-- The following illustrates the use of the 'animate' element
           to animate a rectangles x, y, and width attributes so that
           the rectangle grows to ultimately fill the viewport. -->
   <rect id="RectElement" x="300" y="100" width="300" height="100"
           style="fill:rgb(255,255,0)" >
      <animate attributeName="x" attributeType="XML"
                  begin="0s" dur="9s" fill="freeze" from="300" to="0" />
      <animate attributeName="y" attributeType="XML"
                  begin="0s" dur="9s" fill="freeze" from="100" to="0" />
      <animate attributeName="width" attributeType="XML"
                  begin="0s" dur="9s" fill="freeze" from="300" to="800" />
      <animate attributeName="height" attributeType="XML"
                  begin="0s" dur="9s" fill="freeze" from="100" to="300" />
   </rect>

   <!-- Set up a new user coordinate system so that
           the text string's origin is at (0,0), allowing
           rotation and scale relative to the new origin -->
   <g transform="translate(100,100)" >
      <!-- The following illustrates the use of the 'set', 'animateMotion',
            'animateColor' and 'animateTransform' elements. The 'text' element
            below starts off hidden (i.e., invisible). At 3 seconds, it:
               * becomes visible
               * continuously moves diagonally across the viewport
               * changes color from blue to dark red
               * rotates from -30 to zero degrees
               * scales by a factor of three. -->
      <text id="TextElement" x="0" y="0"
              style="font-family:Verdana; font-size:35.27; visibility:hidden" >
         It's alive!
         <set attributeName="visibility" attributeType="CSS" to="visible"
               begin="3s" dur="6s" fill="freeze" />
         <animateMotion path="M 0 0 L 100 100"
               begin="3s" dur="6s" fill="freeze" />
         <animateColor attributeName="fill" attributeType="CSS"
               from="rgb(0,0,255)" to="rgb(128,0,0)"
               begin="3s" dur="6s" fill="freeze" />
         <animateTransform attributeName="transform" attributeType="XML"
               type="rotate" from="-30" to="0"
               begin="3s" dur="6s" fill="freeze" />
         <animateTransform attributeName="transform" attributeType="XML"
               type="scale" from="1" to="3" additive="sum"
```

```
            begin="3s" dur="6s" fill="freeze" />
    </text>
  </g>
</svg>
```

在支持 SVG 的浏览器中的显示如图 15-10。

初始时



三秒时

六秒时

九秒时

图 15-10

5. SVG 滤波元素应用演示（Filter elements example）

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 03December 1999//EN"
                    "http://www.w3.org/Graphics/SVG/SVG-19991203.dtd">
<svg width="7.5cm" height="5cm" viewBox="0 0 200 120">
  <title>Example filters01.svg - introducing filter effects</title>
  <desc>An example which combines multiple filter primitives
        to produce a 3D lighting effect on a graphic consisting
        of the string "SVG" sitting on top of oval filled in red
        and surrounded by an oval outlined in red.</desc>
  <defs>
    <filter id="MyFilter">
      <desc>Produces a 3D lighting effect.</desc>
      <feGaussianBlur in="SourceAlpha" stdDeviation="4" result="blur"/>
      <feOffset in="blur" dx="4" dy="4" result="offsetBlur"/>
      <feSpecularLighting in="blur" surfaceScale="5" specularConstant="1"
                          specularExponent="10" style="lighting-color:white"
                          result="specOut">
        <fePointLight x="-5000" y="-10000" z="20000"/>
      </feSpecularLighting>
      <feComposite in="specOut" in2="SourceAlpha" operator="in" result="specOut"/>
      <feComposite in="SourceGraphic" in2="specOut" operator="arithmetic"
                   k1="0" k2="1" k3="1" k4="0" result="litPaint"/>
      <feMerge>
        <feMergeNode in="offsetBlur"/>
```

```
        <feMergeNode in="litPaint"/>
      </feMerge>
    </filter>
  </defs>
  <rect x="1" y="1" width="198" height="118" style="fill:#888888; stroke:blue"/>
  <g style="filter:url(#MyFilter)">
      <g>
      <path style="fill:none; stroke:#D90000; stroke-width:10"
            d="M50,90 C0,90 0,30 50,30 L150,30 C200,30 200,90 150,90 z" />
      <path style="fill:#D90000"
            d="M60,80 C30,80 30,40 60,40 L140,40 C170,40 170,80 140,80 z" />
      <g style="fill:#FFFFFF; stroke:black; font-size:45; font-family:Verdana">
        <text x="52" y="76">SVG</text>
      </g>
    </g>
  </g>
</svg>
```

在支持 SVG 的浏览器中的显示如图 15-11。

图 15-11

下面我们分步来观察滤波的过程，我们把上例分解为 6 步，这有助于更好地理解。

```
filter id="MyFilter">
            <desc>Produces a 3D lighting effect.</desc>
1           <feGaussianBlur in="SourceAlpha" stdDeviation="4" result="blur"/>
2           <feOffset in="blur" dx="4" dy="4" result="offsetBlur"/>
3           <feSpecularLighting in="blur" surfaceScale="5" specularConstant="1"
                        specularExponent="10" style="lighting-color:white"
                        result="specOut">
    <fePointLight x="-5000" y="-10000" z="20000"/>
  </feSpecularLighting>
4           <feComposite in="specOut" in2="SourceAlpha" operator="in" result="specOut"/>
5           <feComposite in="SourceGraphic" in2="specOut" operator="arithmetic"
                k1="0" k2="1" k3="1" k4="0" result="litPaint"/>
  <feMerge>
    <feMergeNode in="offsetBlur"/>
6           <feMergeNode in="litPaint"/>
  </feMerge>
</filter>
```

在支持 SVG 的浏览器中的分解显示如图 15-12。

源图像　　第一步处理结果　　第二步处理结果

第三步处理结果　　第四步处理结果

第五步处理结果　　第六步处理结果

图 15-12

15.1.4　前景

目前 SVG 工作组正紧锣密鼓地收集各方面的意见和建议，对现行草案进行修订整理，预计近期即可推出正式的标准规范。由于 SVG 规范尚未成为正式的推荐标准，因此目前的浏览器尚不支持。作为一个开放的标准，SVG 不属于任何公司，它是工业界先驱们同心合作的产物。因为 SVG 标准对任何想使用它们的公司或个人都是开放的，已经有很多公司的软件支持 SVG 的创建、编辑和浏览。去年年底 PGML 建议案的提出者 Adobe 即与 IBM 合作展示了 PGML 浏览器。Adobe 计划在整个产品线上需要的地方会加上支持 SVG 的功能。Adobe 也开发了一个 SVG viewer。Adobe 期望更多的公司支持 SVG 的创建、编辑和浏览。

综上所述，SVG 有着不可忽视的优势：

- 一个 SVG 图像是由基于 XML 的文字命令组成的，可以和 JavaScript 或 XML 完全一致，它是非常精练的编码。
- 作为基于文字的格式，在 SVG 图像内的文字可以让用户通过搜索引擎在浏览器内进行搜索。
- SVG 也可以通过任何 scripting 语言进行创建，如 JavaScript，Perl 或 Java。
- SVG 完全支持 DOM，因而是完全可编写的。SVG 图像或其部分都可以对鼠标点击或移动给出反应，引发图形本身的变化或链接到其它 HTML 页面或其它图形上。
- SVG 对于不同的平台、不同的输出分辨率、不同的颜色空间、不同的带宽等，都能很好地工作。

鉴于 SVG 出色的优点，我们有理由相信在不久的将来基于 SVG 技术的浏览技术必将成为 Web 技术上一个新的热点。

15.2 矢量标记语言 VML

15.2.1 概述

VML（Vector Markup Language，矢量标记语言）是一种基于 XML 的标记语言，XML 是新兴的、灵活的、开放的且基于文本的扩展的 HTML 的语言，但 VML 并未得到 W3C 的正式认可。目前对该语言的描述是非正式的，今后还会继续变化。W3C 成员 Autodesk 有限公司、Hewlett-Packard 公司、Macromedia 有限公司和 Visio 公司，还有微软公司，把矢量标志语言规范提交给 W3C。VML 以现有 HTML 能力为基础并允许矢量图形信息和 HTML 页中的文本和其他数据集成。

VML 是 XML 的一个应用程序，它使创建引人注目的网页变得更快也更容易，它允许用户和作者利用应用程序剪贴矢量图，而不会有任何质量损失或者丧失任何编辑能力。VML 也定义了一套转换方式，可以推动在 Microsoft Office 中创建的文档中的商业图表的描述。用 XML 语句包装矢量图形可以让图形更加灵活。使用基于 XML 的矢量图形语言可以在描述图形之外增加与其相关的各种信息。例如，一个矢量图形文件可以包括版权信息、打印的实际大小或使用费用等等。所有这些信息均可以包含在一个头（Header）中，目前矢量图形格式就是这样描述这些元数据的。并且因为这些信息可以被组织成元数据，所以既不会明显地增大整个文件的大小，也不会延长下载的时间。

XML 本身是基于 HTML 的，所以 VML 的核心也是基于 HTML 的。VML 将图形信息与文本信息（或者 HTML 页面中可以描述的其他数据信息）结合在一起。而且 VML 目前是 W3C 站点上的一个开放标准，完全建立在其他 W3C 认可的开放标准（例如 HTML 4.0、XML 1.0 和 CSS Level 2）之上。这意味着 VML 可以与现有的浏览器向后兼容。

VML 的设计目的是提供一种用文本方式描述矢量图形的语言。因为文本是可以剪切和拷贝的，所以能够在不同的创作工具之间剪切和拷贝矢量图形。VML 在很多方面与 HTML 类似。VML 也使用 XML 的语法并使用 CSS Level 2 标准描述矢量图形的布局。HTML 中记录文本文件的位置信息以及其他信息，然后依赖于操作系统提供的函数处理这些信息。VML 中记录矢量路径的位置信息以及像位图这样的相关对象的信息，然后依赖于操作系统提供的函数如 Win32 GDI 或 Macintosh QuickDraw 等来画出图形。VML 描述的对象也是可以编辑的。在 HTML 的术语中，这些可编辑的对象称作段、表格或表。VML 将其称为形状（Shape）或形状集合。

VML 支持矢量图形。VML 使用封闭和开放的路径描述直线的序列。有形状的基础曲线均是三次 Bezier 曲线，这些曲线还可以被填充或点描。

VML 描述文本的方式类似于 HTML 或 XML。它使用 CSS Level 1 作为字体规范的机制，而且遵循 CSS Level 1 的标准使用 RGB 定义颜色值。这样它可以支持真彩色，而且在必要时可以提供描述索引颜色的方法。它对透明处理、模板处理和模糊处理的支持是缺省的。除此之外，VML 还支持光栅数据和矢量数据的混合使用。它可以在矢量路径中插入光栅数据，这样就可以将两者结合起来。

目前业界对 VML 的支持最广泛，它最有希望以产品的形式实现而成为一种“标准”。AutoDesk、HP、Macromedia、Microsoft 和 Visio 等公司均支持 VML。Microsoft 甚至声称今年年底将会在它的产品中支持 VML。同 PGML 一样，VML 基于 XML 并且支持像 CSS 这样的

Web 标准。PGML 和 VML 的目标是一致的，即在 Web 上建立高效、灵活和简洁的图形。VML 的基本规则：

单个的元素被定义为形状。大多数形状是由矢量路径描述的。例如，通过定义直线和三次 Bezier 曲线的顺序可以定义一个形状的轮廓。这些单个的形状又可以组织成组（Group）。为了创建起来更容易，VML 还提供了一些预定义的形状，包括直线、多边形和曲线。使用这些预定义好的形状描述新的形状要方便一些，只需要指明该路径即可。VML 还允许所有的形状引用一种叫作 Shapetype 的形状原型。

VML 形状既可以独立地产生，也可以是多个形状相关的。因为各个形状本身都包含了自身的特征比率信息，所以整个组可以扩展而不会影响到其中包含的内容。这意味着 VML 图形可以在任意标准的文本或 WYSIWYG（What You See IS What You Get，所见即所得）的编辑器中重置大小或重新编排形状而不会影响精确度或图形质量。

最终的结果是 VML 可以定义完整集成并且可扩展的 Web 图形。因为这些图形使用基于三次 Bezier 曲线的路径，可以以很小的存储量保存光滑的曲线。这种简洁的表示方法和灵活的可重用形状设计导致了图形下载时间的缩短。因为 VML 图形可以完全集成在 HTML 文档中，它们还可以与页面上的其他组成部分进行交互和扩展，而且 VML 让增加超链到图形元素的工作也变得更加简单。

15.2.2 VML 基础及应用

1. 第一个 VML 图形

VML 描述的图形有：矩形、椭圆、圆、三角形、梯形等。每种图形都可表述为由一组相连的曲线和直线构成的轨迹。VML 以各种元素及其属性来描述每种图形的轮廓、填充色、位置和其他一些特性。图 15-13 是一个用 VML 实现的矢量图形，在 IE5 中可以浏览。

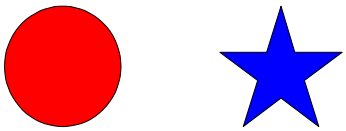


图 15-13

VML 实例：

```
<html xmlns:vml="urn:schemas-microsoft-com:vml">
<head>
  <title>VML sample</title>
  <object id="VMLRender"
    classid="CLSID:10072CEC-8CC1-11D1-986E-00A0C955B42E">
  </object>
  <style>
    vml\:* { behavior: url(#VMLRender) }
  </style>
</head>

<body>
  <div>
    <vml:oval
      style="width:200px; height: 200px"
      stroke="true"
```

```
        strokecolor="red"
        strokeweight="2">
      </vml:oval>

      <vml:polyline
        style="width: 250px; height: 250px"
        stroke="false"
        fill="true"
        fillcolor="blue"
        points="8pt, 65pt, 72pt, 65pt, 92pt, 11pt, 112pt, 65pt,
                174pt, 65pt, 122pt, 100pt, 142pt, 155pt, 92pt,
                121pt, 42pt, 155pt, 60pt, 100pt">
      </vml:polyline>
    </div>
    <hr></hr>
  </body>

</html>
```

2. VML 元素

整个 VML 的结构主要由三个基本元素：形状（shape）元素、形状类型（shapetype）元素和组（group）元素来定义。

（1）形状（shape）元素。shape 是 VML 的基本元素，它描述任意二维闭合曲线。大多数 shape 元素的参数可以由各种属性来定义，属性定义如下：

```
<!entity %coreattrs
  id          id      #implied --此元素唯一的 XML 名称--
  class       cdata   #implied --该图的 CSS 类 --
  style       cdata   #implied --  该图所用 CSS 参数  --
  title       cdata   #implied --图形名--
  href        cdata   #implied --单击图形时的链接 URL --
  target      cdata   #implied --加载框架名称--
  alt         cdata   #implied --   不能显示图形时的替代文本  --
  coordsize   cdata   #implied --   本地坐标表示的图形框高度和宽度  --
  coordorigin cdata   #implied --   图形左上角的坐标角  --
  wrapcoords  cdata   #implied --   指定密集文本如何围绕一个 shape --
>
```

另外，shape 元素还有一些用来控制图形修饰的属性：

```
<!entity %shapeattrs
  opacity       cdata #implied --   图形透明度，介于 0.0（不透明）和 1.0（全透明）之间  --
  chromakey     cdata #implied –  图形透明背景色  --
  stroke        cdata #implied --   是否绘制图形的轨迹  --
  strokecolor   cdata #implied --   画图轨迹所用颜色  --
  strokeweight  cdata #implied --   画图轨迹宽度  --
  fill          cdata #implied --   是否填充  --
  fillcolor     cdata #implied --   填充色  --
```

```
    print          cdata #implied --  打印页面是否打印图形 --
>
```

属性预设缺省值：

```
<shapetype
    adj=null
    path=null
    opacity="100%"
    chromakey="none"
    stroke="true"
    strokecolor="black"
    strokeweight="0.75pt"
    fill="true"
    fillcolor="white"
    print="true"
    id=null
    class=null
    style='visibility: visible'
    title=null
    href=null
    target=null
    alt=null
    coordsize="1000, 1000"
    coordorigin="0, 0"
    wrapcoords=null
/>
```

shape 元素还有一些子元素用来对图形的某些层面进行更细微的控制。如果子元素设置与某一属性设置相背，则采用子元素的设置。子元素如下：

```
<!entity %shape.elements
    (path | formulas | handles | fill | stroke | shadow | textbox | textpath | imagedata |
    %extensions;)
>
```

（2）形状类型（shapetype）元素。shapetype 元素定义可重复利用的图形，方法是在同一文档的后端用 shape 元素指向 shapetype 元素，除了不能指向其它 shapetype 元素以外，shapetype 元素与 shape 元素是一样的。下面列举一个多个 shape 元素复制一个 shapetype 元素的实例。

Shapetype 实例：

```
<html xmlns:vml="urn:schemas-microsoft-com:vml">
    <head>
        <title>shapetype sample</title>
        <object id="VMLRender"
            classid="CLSID:10072CEC-8CC1-11D1-986E-00A0C955B42E">
        </object>
        <style>
            vml\:* { behavior: url(#VMLRender) }
        </style>
    </head>
```

```
<body>
  <vml:shapetype id="fred"
      coordsize="21600,21600"
      fillcolor="blue"
      path="m@0,0l0,21600,21600,21600xe">
    <vml:formulas>
      <vml:f eqn="val #0"/>
      <vml:f eqn="prod #0 1 2"/>
      <vml:f eqn="sum @1 10800 0"/>
    </vml:formulas>
  </vml:shapetype>

  <vml:shape type="#fred" style="width:50px; height:50px" />
  <vml:shape type="#fred" style="width:100px; height:100px"/>
  <vml:shape type="#fred" style="width:150px; height:150px"/>

 </body>
</html>
```

（3）组（group）元素。顶级元素 group 将形状元素和其它一些顶级元素结合在一起，它有自己的局部坐标空间，用来存放一些子图形，因此由各图形所构成的集合可作为一个单元移动摆放。组元素只具有形状元素的核心属性（如：id , class , style , title , href , target , alt , coordorigin 和 coordsize 等）。举例来说，如果我们要画一个有外接圆的五角星，可通过在一个 group 中结合 oval 元素和 polyline 元素来进行创建。

```
<vml:group style="width:6cm;height:6cm "
coordorigin="0 0 "    coordsize="250 250">
<vml:oval style =
"position:absolute;top:15;left:15;width:200;height:200 "
stroke="true " strokecolor="black " strokeweight="2 "
fill="true " fillcolor="red ">
</vml:oval>
<vml:polyline style =
"position:absolute;top:25;left:25;width:200;height:200 "
stroke="true " strokecolor="black " strokeweight=”5 "
fill="true " fillcolor="blue "
points="8,65,72,65,92,11,112,65,174,65,122,
100,142,155,92,121,42,155,60,100 ”>
</vml:polyline>
</vml:group>
```

这里的 coordsize 属性和 coordorigin 属性定义了组中所含元素的局部坐标系统。coordsize 属性定义了沿包容块宽度的单元数目，coordorigin 属性定义包容块的左上角坐标。

Group 元素的子元素定义如下：

```
<!element group
   (group | shape | shapetype | line | polyline | curve | rect | roundrect | oval | arc | image)*
>
```

属性设置见 shape 元素设置中对应项：id, class, style (top, left, width, height, rotation, z-index,

position, visibility), title, href, target, alt, coordsize, coordorigin。

Group 元素属性预设缺省值：

```
<group
  id=null
  class=null
  style='visibility: visible'
  title=null
  href=null
  target=null
  alt=null
  coordsize="1000, 1000"
  coordorigin="0, 0"
  wrapcoords=null
/>
```

VML 的其它元素还很多，这里不一一赘述了，感兴趣的读者可参考相关文档和规范。

我们以一个绘制翻转的五角星图形的例子来结束本节的内容：

```
<html xmlns:vml="urn:schemas-microsoft-com:vml">

  <head>
    <title>Flip sample</title>
    <object id="VMLRender"
      classid="CLSID:10072CEC-8CC1-11D1-986E-00A0C955B42E">
    </object>
    <style>
      vml\:* { behavior: url(#VMLRender) }
    </style>
  </head>

  <body>
     <div>
      <vml:polyline
        style="width: 250px; height: 250px; flip: y"
        stroke="true"
        strokecolor="black"
        strokeweight="5"
        fill="true"
        fillcolor="blue"
        points="8pt, 65pt, 72pt, 65pt, 92pt,11pt, 112pt, 65pt,
                174pt, 65pt, 122pt,100pt, 142pt, 155pt, 92pt,
                121pt, 42pt, 155pt, 60pt, 100pt, 8pt, 65pt">
      </vml:polyline>
    </div>
   </body>

</html>
```

图 15-14

15.2.3 VML 前景

VML 格式最初是通过文本和 WYSIWYG 页面编辑器生成的。随着 VML 的发展，Microsoft 的 Office 和其他的一些应用程序将会实现通过简单的“另存为”命令产生 VML 文件的功能。

因为 VML 在 HTML 文件中嵌入矢量数据，所以现有的页面布局工具应该能够处理这些数据，而内容创建工具应能移入 VML 文件并无缝地嵌入特殊内容的信息。这需要对现有的应用程序做一些改动，不过改动不会很大。

所有兼容 VML 的应用程序应该能够编辑 VML 元素，即使这些元素是由别的应用程序生成的也不例外。其最终目标是创建这样的图形，它可以嵌入 HTML 文件中在 Web 上发布，可以被重新打开编辑而不会丢失细节或导致图形质量下降。

对浏览器端而言，现在确切地描述 VML 的实现过程还为时过早。唯一可以确定的是浏览器中必须带有 XML 语法分析器。IE 5.0 带有内置的模块以支持 VML，其他的浏览器也应该带有相应的插件。VML 还规定允许内容生成器生成位图图形以达到与不识别 VML 的浏览器兼容的目的。虽然这不能显示矢量图形系统的优势，但可以确保图形对所有的用户都是可识别的。

尽管目前 Microsoft 正在全力推出支持 VML 的下一代产品，距 W3C 的可扩展矢量图形工作小组做出推荐使用的决定还会有大约一年的时间，而且现有几种矢量图形格式现阶段也难分高下。可能的情况是从 PGML、XML 和 Web Schematics 中抽取出一种全面的基于 XML 的矢量图形标准。

与之有关的各个厂家也纷纷表示希望能够进行简单的结合，Adobe 和 Microsoft 声明会紧随 W3C 的标准化过程，完全支持最终的规范。但在该规范还未确定之前，Microsoft 还是会支持 VML。那么，就让我们拭目以待下个版本的 Office 和 IE 中对 VML 的更加有力的支持吧！

15.3 小　　结

XML 规范的推广和应用为网络矢量图形处理提供了标准和很强的技术支持。随着技术的发展，未来的网络浏览器将会内置对基于 XML 的矢量图形文件的支持功能，从而使它更适合于 Web 应用，并进而发展成为未来 Web 图形的主流。

第十六章　WML——无线接入的 XML

本章介绍基于 XML 的无线标记语言 WML，WML 技术是用来定义窄带设备中使用的内容和用户接口，这些窄带设备包括移动电话和数字寻呼。我们将在初步了解 WAP 协议的 基础上，学习 WML 的语言规范，进而应用它进行网页设计和实现 Web 服务。

本章包括以下内容：

- 可伸缩的矢量图形 SVG
- 矢量标记语言 VML

16.1　无线应用协议——WAP

16.1.1　WAP 入门

WAP 是 WIRELESS APPLICATION PROTOCOL（无线应用协议）的简称，是在数字移动电话、因特网或其他个人数字助理机（PDA）、计算机应用之间进行通讯的开放全球标准。它是开发移动网络上类似互联网应用的一系列规范的组合。WAP 协议与现在通行的互联网协议类似，但专为小屏幕、窄带的用户装置（如移动电话）优化。WAP 协议是公开的、全球性的标准，由有兴趣参加 WAP FORUM 的成员共同讨论、制定和拥有，它使无线装置可以轻易、实时地交流信息和服务。

1.WAP 形成

摩托罗拉、诺基亚、爱立信和美国的软件公司 PHONE.COM（前 UNWIRED PLANET）于 1997 年中联合发起设立无线应用协议（WAP）标准。它定义了一系列将互联网内容过滤和转化为适用移动通信的标准，使内容可以更容易地在移动终端上显示。

WAP FORUM 成立于美国网络营运商 OMNIPOINT，于 1997 年发出征求移动信息服务的标书之后，多个不同的供应商都参与竞标，提出各自的专用无线信息传输方案，如诺基亚的 SMART MESSAGING 和 PHONE.COM 的 HDML。OMNIPOINT 告知投标人它不打算采用个别公司的专利方案，并建议不同的投标人合作开发一种通用标准，毕竟，不同方案之间并没有本质的区别，可以合组提炼出有用的标准。这类事件是 WAP 开发最初的促进因素。

WAP 专注于用户服务器的研究。它采用一种相对简单的浏览器，仅占用移动电话上很少的资源，使 WAP 适用于单薄的终端设备和早期的智能电话。WAP 赋与 WAP 网关强大的功能而只需在移动电话上增加一个微浏览器，服务和应用暂时都在服务器端进行处理。WAP 致力于将普通的移动电话改造成为具有网络功能的智能电话。正如发起人之一的 PHONE.COM 公司的代表宣称，WAP 背后的意义是尽可能少地利用手持设备的有限资源，通过丰富的网络功能来弥补。

2. WAP 组成

WAP 体现为一种全面的和可扩展的协议，可用于：

- 任何具有类似智能电话的单线展示的移动电话。
- 任何现有或设计中的无线服务，如 SMS，CSD，USSD 和 GPRS。
- 移动网络标准如 CDMA，GSM 或 UMTS。WAP 可用于全部的蜂窝标准并得到主要的无线巨头的支持。支持多种输入终端，如手写板、键盘、触摸屏和笔。

WAP 的重要性在于为应用开发人员和营运商在不同类型的网络、数据、终端上的服务提供了革命性的新途径。WAP 标准在设计时将应用要素与使用的传输数据类型独立开来，使一些应用的转移（如从 SMS、CSD 转为 GPRS）成为可能。

WAP 是由一系列协议组成，用来标准化无线通信设备。WAP 将移动网络和 Internet 以及公司的局域网紧密地联系起来，提供一种与网络类型、运行商和终端设备都独立的移动增值业务。

通过这种技术，无论在何地、何时只要需要信息，用户就可以打开 WAP 手机，享受无穷无尽的网上信息或者网上资源。如：综合新闻、天气预报、股市动态、商业报道、当前汇率等。电子商务、网上银行也将逐一实现。用户还可以随时随地获得体育比赛结果、娱乐圈趣闻以及幽默故事，为生活增添情趣，也可以利用网上预定功能，把生活安排得有条不紊。

WAP 协议包括以下几层：

- Wireless Application Environment (WAE)
- Wireless Session Layer (WSL)
- Wireless Transaction Layer (WTP)
- Wireless Transport Layer Security (WTLS)
- Wireless Transport Layer (WDP)

其中，WAE 层含有微型浏览器、WML、WMLSCRIPT 的解释器等功能。WTLS 层为无线电子商务及无线加密传输数据时提供安全方面的基本功能。

WAP 协议的诞生是 WAP 论坛成员多年努力的结果。它是针对不同的协议层定义了一系列协议，这些协议使得各方面的厂商和公司可以协同工作，开发无线通信网络的应用。目前有超过 100 个成员加入 WAP 论坛，包括有终端和基础设备的制造商移动通信的网络运营商、业务提供商软件公司以及网络内容提供商等，共同为移动设备开发服务和应用。

但是，目前由于无线网的带宽等等因素的限制，WAP 手机在多媒体上的应用，如：可视会议、多媒体教学等等，还需一段时日。

WAP 规范还在不断地完善，WAP 论坛成员们在加紧开发功能完善的 WAP 设备，这无疑加快其在无线因特网综合服务领域的扩展速度。

3. WAP 体系与结构

WAP 是一个用于向无线终端进行智能化信息传递的无需授权、不依赖平台的协议。WAP 论坛成立于 1998 年初，是一个由 Nokia，Ericsson，Motorola，Unwired Planet 等四家公司发起组成，现拥有 100 多个公司和机构的行业协会，它致力于开发用于数字移动电话和其他无线终端设备的无线信息与电话服务在事实上的全球标准。论坛的目标是将无线行业价值链各个环节上的公司联合在一起以保证产品的互操作性和无线市场的发展。

WAP 提供一种以安全迅速、灵活、在线和交互的方式连接服务、信息和其他用户的媒介。用户可以从通过移动电话、寻呼机或其他无线设备实现的对相关 Internet / Intranet 信息的方便安全的访问。还可以得到消息通知与呼叫管理、电子邮件、电话增值服务与联合消息发送、地图与定位服务、天气与交通预报、新闻、体育信息服务、电子商务交易与银行服务、在线地址簿与目录服务以及企业内联网应用等多项服务。

无线应用协议（WAP）可以建立在 GSM-900，GSM-1800，GSM-1900，CDMA IS-95，TDMA，IS-136（即 DAMPS），第三代系统——IMT-2000，UMTS，W-CDMA，宽带 IS-95，FLEX 寻呼系统，CDPD 等系统上。

（1）WAP 的模型。WAP 编程模型与 WWW 程序模型类似，协议制定者尽可能地参考已有的标准，并作为 WAP 技术的起点，使应用开发者可利用熟悉的编程模型、可靠的体系、现有的工具，而从中受益。针对无线环境的特点，WAP 技术进行了一些优化，增加了几种扩展名。WAP 内容和应用供应商使用基于 WWW 的内容格式，内容传送也使用基于 WWW 通讯协议的一系列通讯协议，管理用户界面的微浏览器也与标准的网络浏览器类似。WAP 定义了允许移动终端和网络服务器之间通讯的标准，包括：

- 标准名字模型——WWW 标准的 URL 同样用来界定 WAP 内容和来源服务器。
- 内容类型——WAP 内容有与 WWW 类型一致的特定类型。
- 标准内容格式——WAP 内容格式基于 WWW 技术，包括显示标识、日历、图形和脚本语言等。
- 标准通讯协议——移动终端与网络服务器之间的请求传送。

WAP 内容种类和协议已经为大部分手持设备优化过了。WAP 规范使用标准的 Web 代理技术将无线网络与 Web 连接到网关中，WAP 大大减少了手机上的操作负载，为手机实现普及提供了基础。例如，一个 WAP 网关一般可以使用所有的 DNS 服务来解析 URL 中使用的域名，因此就不再需要手机来完成这个计算任务。另外，网络还可以利用 WAP 网关来为用户提供各种服务，并且可以帮助网络服务商防止诈骗和服务利用。WAP Proxy 连接无线域和 WWW，主要有以下功能：

- 协议网关：协议网关将来自 WAP 协议栈的请求翻译到 WWW 协议（HTTP 和 TCP/IP）中。
- 内容编码器和解码器：内容编码器将 Web 内容翻译成压缩编码的格式，以减少通过无线数据网络传输的数据包的大小和数量。

这个结构使移动终端用户可以浏览各种 WAP 内容和应用，而不管它们使用的是什么类型的无线网络。应用开发者能够创建网络和终端独立的内容服务和应用，使这些应用可以被尽可能多的用户使用和访问。使用 WAP 代理，内容和应用可以放在标准的 WWW 服务器上，开发者可以继续使用通用的 Web 技术如 CGI 编程来进行开发。

WAP 网关还可以将来自不同 Web 服务器上的数据聚合起来，并且对经常使用的信息进行缓冲处理，从而减少手持设备的应答时间。

WAP 网关还可以与用户的数据库接口，使用来自无线网络的信息如位置信息来为某一组用户动态定制 WML 页面。

（2）WAP 体系结构。层叠排列的 WAP 协议体系为无线装置的应用开发提供了可扩展、可延伸的环境。每一层协议或其它服务和应用程序可与它下一层协议直接对话。通过精心设置

的一系列接口，外围服务和应用程序可以利用 WAP 体系提供的各种功能，包括直接使用会话层、交易层、安全层、传输层等。

在 WAP 的协议栈中的协议层次如图 16-1。

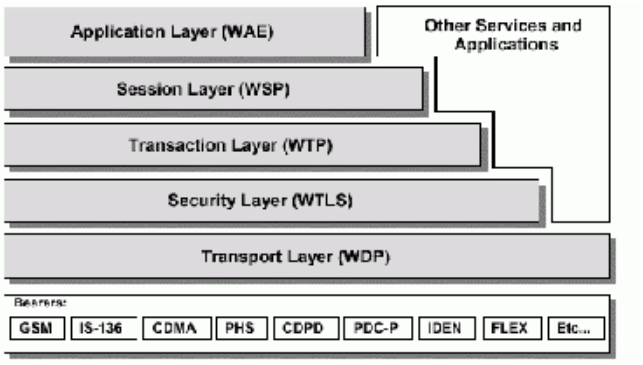


图 16-1

Wireless Application Environment（WAE），无线应用环境 WAE 是基于移动技术与 WWW 结合基础之上的应用环境，目的是为营运商、服务提供商的服务和应用程序建立一个交互操作环境，使他们可以灵活地接入不同平台。WAE 包含一个微型浏览器，具有以下功能：

无线标记语言 WML：一种轻型标记语言，类似 HTML，专为手持终端做了优化；

- WMLScript：轻型的脚本语言，类似于 JavaScript。
- 无线电话应用：电话服务及其编程接口。
- 内容格式：精心设计的数据格式，包括图像、电话号码簿和日历信息。

Wireless Session Protocol（WSP），无线会话协议 WSP 向 WAP 应用层提供两个会话服务的统一接口，一个是在传输层协议 WTP 之上的面向连接的服务，另一个是在安全或非安全数据报服务（WDP）之上的非连接服务。WSP 系列协议针对低效率、长等待时间的网络数据载体进行了优化，它现在由浏览应用的服务组成（WSP/Browsing）,WSP/B 允许使用 WAP Proxy 连接 WSP/B 的客户端和标准的 HTTP 服务器，具有以下功能：

- 无线编码中的 HTTP/1.1 功能和语法。
- 较长时间的会话状态。
- 会话随着会话者移动而暂停或继续。
- 建立一个传送可靠或不可靠的数据的通用设备。
- 协议的协商。

Wireless Transaction Protocol（WTP），无线处理协议 WTP 运行于数据服务之上，提供了一个适用于“轻体”客户（移动终端）的面向传输的轻型协议。WTP 可有效地运行于安全或非安全的无线数据网络，具有以下功能：

- 三个等级的传输服务。
- 不可靠的单向请求。
- 可靠的单向请求。
- 可靠的双向请求-回答传输。
- 可选择的用户-用户连接，WTP 用户自行确认每一收到的消息。

- 可选择的带外数据确认。
- PDU 连接和延时确认，以减少传送的消息数量。
- 异步传输。

Wireless Transport Layer Security（WTLS），无线传输层安全 WTLS 是基于工业标准——TLS（以前称为 SSL）上的安全协议，同样针对移动通信使用的窄频信道进行了优化，它应与 WAP 传输协议同时使用。应用程序可视自己的安全要求和网络特点，选择启用或不启用 WTLS 功能。WTLS 的功能如下：

- 数据的完整性：WTLS 具有保证终端与服务器间传送的数据前后一致且不会损毁。
- 传输的保密性：WTLS 保证端到端的数据的保密性，并可为数据传输过程的中介方读取。
- 认证：终端至服务器的校验。
- “拒绝服务”保护：检验和拒绝重复和未正确识别的数据，以保护上层协议。此外，WTLS 也可用于终端和终端之间的安全通讯，如为交换电子交易卡提供认证。

Wireless Datagram Protocol（WDP），无线数据报协议 WDP 是 WAP 体系的传输层协议，WDP 可运行于各种网络的数据载体。作为一种通用的传输协议，它将传输端口根据底层数据载体进行改造，从而为 WAP 体系中的上层协议提供统一的接口，使会话层、应用层独立于底层的无线网络。如能保持传输层接口和基本功能的一致性，就可通过中介网关使广泛的交互操作得以实现。

BEARER，承载器 WAP 协议在设计时的目的是使它可独立运作于各种不同的数据载体之上，如 SMS（短消息、CSD、封包数据等）。由于数据载体因承载量、容错率和延迟不同而有不同的质量，WAP 协议就需要补偿或容忍这些特点。WDP 集中处理体系中其它层次协议与数据载体的交流，除现在已以可支持的数据类型外，随着新的数据服务在移动市场出现，它也会不断发展以支持更多的数据类型。

其他服务和应用 通过一系列精心设置的接口，WAP 协议还可以支持其他服务和应用程序使用 WAP 提供的功能。外围应用程序可直接切入会话层、交易层、安全层和传输层，虽然这类有价值的应用未在 WAP 标准中界定。WAP 协议可用来开发如电子邮件、日历、电话号码本、手写板、电子商务和黄页、白页等各类服务等。

WIRELESS TELEPHONY APPLICATION（WTA），无线电话应用 WAP 标准也定义了一个名为 WTA 的协议。它是一个面向通讯的技术，使 WAP 能够融合电信网络中先进服务，比如智能网络（Intelligent Networks）。在融合基于浏览器的 WAP 用户界面，WTA 能够不用修改终端就直接享受基于智能网络的服务。

4. WAP 的现状和将来

WAP 论坛在 1998 年 4 月发布了 WAP v1.0，v1.1 在 1999 年 5 月前获得通过。WAP v1.1 与 WAP v1.0 具有相同的功能性，并在为商业应用准备的新版本中增加了来自第一批用户的详细反馈。WAP 使用了许多 Internet 标准，如 XML，UDP 和 IP。许多预留的协议是基于像 HTTP、TLS 这类 Internet 标准的，但为适应无线环境的特殊限制而进行了优化。未来的 WAP 将包括端到端安全性、智能卡接口、面向连接的传输协议、持续存储、计费接口和推送技术，并且将向支持多媒体移动服务发展。

16.1.2 WAP 规范详解

WAP 的协议规范（Specification）有三种：已批准（Approved）规范，提议中的（Proposed）规范和原型（Prototype）规范。我们在这里涉及的是 WAP 规范套件 1.2（WAP 1.2 Specification Suite）。

下面将逐一对 WAP 规范套件的文件内容进行简单介绍。

（1）SPEC-PAP-19991108，此文件定义了一个 PUSH 访问协议（Push Access Protocol，PAP）。有时人们需要从某个 Internet 上的服务器向移动设备如手机上主动发送某些信息，比如每天的天气预报、电子邮件通知、每日新闻等。WAP 规定这一过程如图 16-2 所示。

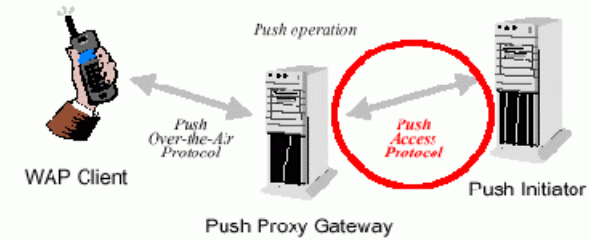


图 16-2

由图 16-2 所示，WAP 规定在只有从源服务器（Push Initiator，PI）到 Push 代理网关的这段过程才需要满足 PAP 协议，而由网关到客户端（Client）的这一过程，则需满足 PushOTA（Push Over The Air）协议。

（2）SPEC-PPGService-19990816，文件定义了一个 Push 代理网关服务的规范（Push Proxy Gateway Service Specification，PPG）。在 WAP 的 Push 过程中，必须要有一个代理网关服务器将有线网络和无线网络连接起来。PPG 规定的，就是这个服务器在完成 Push 操作时所必须满足的一些功能，如图 16-3。

图 16-3

（3）SPEC-PushArchOverview-19991108，文件对 WAP 中 Push 操作的整体流程和结构进行了说明。

（4）SPEC-PushMessage-19990816，文件对如何在源服务器上定义 Push 消息进行了具体说明。

（5）SPEC-PushOTA-19991108，PushOTA 是 Push On The Air 的缩写。文件对 Push 在代理网关到客户端（手机）之间的数据传输过程所必须满足的一些协议进行了规定。

（6）SPEC-ServiceInd-19991108，WAP 允许 Push 服务器不同步地向移动终端发一些服务

指示（Service Indication，SI），比如新邮件通知、股票价格变化通知和重要的新闻通知等，用户可以选择是进一步阅读更详细的信息，还是另选时间再行阅读。文件对这一过程进行了定义。

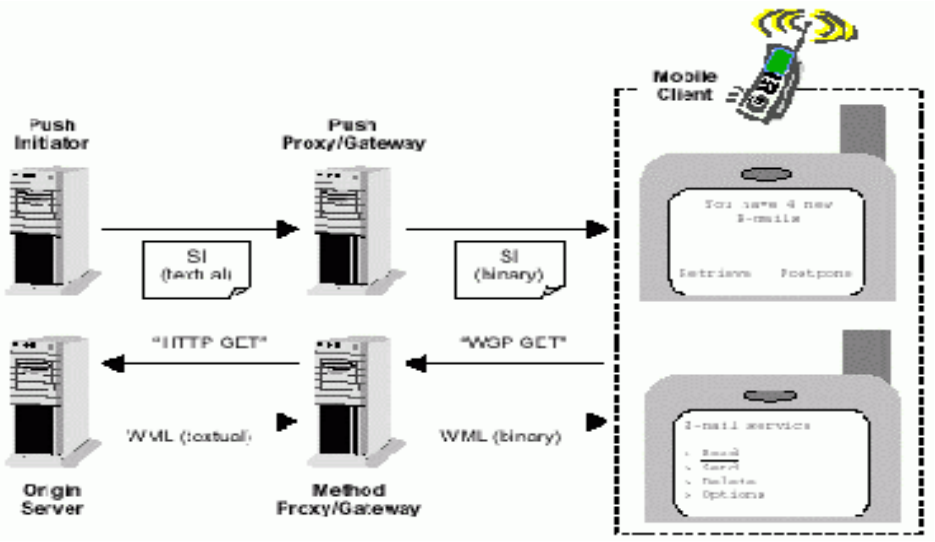


图 16-4

从图 16-4 中可知，当 Push 代理网关接收到源服务器以文本格式发来（PAP 方式）的 SI 之后，将其转换成二进制格式发送（OTA 方式）到移动终端（手机），用户对指示信息进行反应。如果选择阅读进一步的信息，则向 Push 代理网关发出 WSP（Wireless Session Protocol）会话请求，网关将其转换成 HTTP 请求发还源服务器，源服务器收到请求后再将指示信息的细节内容用 WML（Wireless Markup Language）的格式（通过网关转换成二进制后）送到终端，进行显示。文件不仅仅规定了上述过程的基本内容，还对 SI 的其他处理方式进行了定义，如删除、修改、过期处理等等。

（7）SPEC-ServiceLoad-19991108，有时候直接将服务送到移动终端（手机）并进行执行并不合适，特别是终端正在处理另一个服务的时候，因为有些终端的内存和处理能力不能同时处理多个服务，而且，如果将信息直接送入，则可能会中断前一个服务的处理过程。为了处理这个难题，WAP 定义了另外一种内容类型：服务载入（Service Loading，SL）。

（8）SPEC-UAProf-19991110，文件对 WAP 终端的所必须满足的一些条件（User Agent Profile）进行了定义和说明。

（9）SPEC-WAEOverview-19991104，

文件对无线应用环境（Wireless Application Protocol，WAE），即进行 WAP 编程所需的知识，与 WAP 其他技术之间的联系等进行了整体性的说明和描述。

（10）SPEC-WAESpec-19991104，文件对 WAE 的文档结构进行了说明，即 WAE 结构包括哪些组件，这些组件都有些什么特征等等。

（11）SPEC-WAPArch-19980430，文件对 WAP 协议的整体构架和组成进行了说明，是了解 WAP 整体状况的出发点和纲领性文档。

（12）SPEC-WAPCachingMod-19990211，文件对 WAP 终端的缓冲模式所必须满足的条件

进行了定义和说明。

（13）SPEC-WAPOverGSMUSSD-19990715，文件在对 GSM USSD（Unstructured Supplementary Service Data）进行了简单的描述后，定义了 WAP 到它的映射关系。

（14）SPEC-WBXML-19991104，当 WAP 的数据在无线网上以 XML 格式进行传输时，为了减少传输的数据量，提高传输的效率，必须将 XML 文档以压缩后的二进制格式发送。文件对这种二进制 XML 内容格式（Binary XML Content Format，WBXML）进行了定义和说明。

（15）SPEC-WCMP-19990804，WAP 构架中的传输层协议包括无线事务协议（Wireless Transaction Protocol，WTP）和无线数据报文协议（Wireless Datagram Protocol，WDP）。文件对 WDP 的报错机制即无线消息控制协议（Wireless Control Message Protocol，WCMP）进行了定义和说明。

（16）SPEC-WDP-19991105，作为一种通用的传输服务，WAP 构架中的 WDP 层为其上的其他 WAP 协议提供统一的服务，并与其下的各种承载器（Bearer）进行透明的通讯。文件对 WDP 的构架，WDP 层与其他层次之间的通讯方式等内容进行了定义和说明。SPEC-WDP-WCMPAdapt-19991105 文件对移动终端和 WAP 代理网关的 WDP、WCMP 如何通过无线数据网关进行访问进行了定义和说明。

（17）SPEC-WIM-19991105，WAP 的安全功能包括无线传输层安全（Wireless Transport Layer Security）和应用程序的安全层，而 WAP 身份认证模块（WAP Identity Module，WIM）就用于执行这两个安全层。文件对 WIM 的构架、格式和组件等内容进行了详细的说明和阐释。

（18）SPEC-WML-19991104，无线置标语言（Wireless Markup Language，WML）是一种基于 XML 的置标语言，在 WAP 中用于表现内容和对用户接口进行描述。文件对 WML 的语法进行了详细的定义和说明。

（19）SPEC-WMLScript-19991104，WMLScript 由 WAP 通过对 ECMAScript 的改进后得到的，它很好地符合了移动通讯中带宽窄、客户端功能不足等特点，用于协助 WML 提高客户端的处理能力。文件对 WMLScript 的特点、语法等内容进行了详细的说明和定义。

（20）SPEC-WMLScriptCrypto-19991105，文件提供了一个 WMLScript 的加密函数库，以便向 WAP 客户端提供加密功能，同时，文件还定义了一个加密数据格式，以便在 WAP 设备之间传送加密数据。

（21）SPEC-WMLScriptIntent-19980430，文件介绍了在其他文件中漏掉的一些 WMLScript 特性。

（22）SPEC-WMLScriptLibs-19991104，文件对 WMLScript 的标准函数库进行了定义和说明。

（23）SPEC-WSP-19991105，无线会话协议（Wireless Session Protocol，WSP）是 WAP 的会话层协议，用于对上层的应用程序层提供两种会话服务的统一接口：一种是建立在 WTP 基础上的面向连接的（connection-oriented）服务，另一种是建立在 WDP 基础上的无连接服务。文件对 WSP 的构架、组件、数据结构等进行了详细的定义和说明。

（24）SPEC-WTA-19991108，无线通讯应用程序（Wireless Telephony Application，WTA）框架用于提供无线通讯服务，WTA 终端则是标准 WML 终端的一种扩展，以便使其能够执行所在无线网络所提供的其他通讯服务，但是，文件中所介绍的协议则仅限于无线终端之上。

（25）SPEC-WTAI-19991108，文件介绍了 WAP 为支持 WTA 而作的扩展，即无线通讯应

用程序接口（Wireless Telephony Application Interface，WTAI）。WTAI 是通过在 WML 和 WMLScript 中加入 WATI 函数库而达到这种扩展目的的。所以用户在阅读此文件之前，最好能够对 WML 和 WMLScript 有比较深入的了解。

（26）SPEC-WTAIGSM-19991108，文件对在其他文件中漏掉的 WTAI 在 GSM 网络中的接口函数进行了定义和说明。

（27）SPEC-WTAIIS136-19991108，文件对在其他文件中漏掉的 WTAI 在 IS-136 网络中的接口函数进行了定义和说明。

（28）SPEC-WTAIPDC-19991108，文件对在其他文件中漏掉的 WTAI 在 PDC 网络中的接口函数进行了定义和说明。

（29）SPEC-WTLS-19991105，WTLS（Wireless Transport Layer Security，无线传输层安全）是 WAP 在另一个工业标准 TLS（Transport Layer Secureity，传输层安全）协议的基础上制定的一个安全协议。其制定的目的就是在窄带宽的通讯频道上安全地使用 WAP 的传输协议。文件对 WTLS 的构架，组件，加密计算等内容进行了说明和定义。

（30）SPEC-WTP-19990611，WTP（Wireless Transaction Protocol，无线事务协议）运行于数据报文服务之上，并提供部分的面向事务（transaction-oriented）协议（因为移动数字终端属于“瘦”客户端，即它无法支持大数据量的事务服务）。文件对 WTP 中的事务类别、操作类别、协议特性等内容进行了定义和说明。

以上内容仅对 WAP 规范套件中各个文件的内容做了一个大致的描述，只能作为读者在阅读文档之前的一个参考材料，而不至于在数十个文件中迷失方向。但要了解其具体细节，还必须请读者自己去下载并仔细阅读。

16.2 WAP 网页设计——WML 编程

本节主要介绍如何编写包含 WAP 协议的手机(以下的手机均指该类手机)所专用的网页，从普通浏览器的角度来介绍 WML 编程语言，使读者能够更好地理解。

16.2.1 Web 服务器的配置

WAP 网络架构由三部分组成，即 WAP 网关、WAP 手机和 WAP 内容服务器，这三方面缺一不可！其中 WAP 网关起着协议的“翻译”作用，是联系 GSM 网与万维网的桥梁；WAP 内容服务器存储着大量的信息，以提供 WAP 手机用户来访问、查询、浏览等。图 16-5 表明了 WAP 网络的基本架构。当用户从 WAP 手机键入要访问的 WAP 内容服务器的 URL 后，信号经过无线网络，以 WAP 协议方式发送请求至 WAP 网关，然后经过“翻译”，再以 HTTP 协议方式与 WAP 内容服务器交互，最后 WAP 网关将返回的内容压缩、处理成 BINARY 流返回到客户的 WAP 手机屏幕上。编程人员所要做的是编写 WAP 内容服务器上的程序或 WAP 网页。

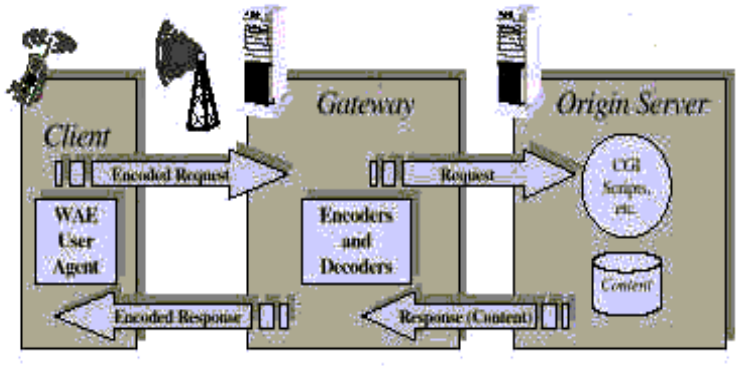


图 16-5

在设计 WAP 网页时不论用户使用的 WAP 开发工具是 UP.SDK 或 NOKIA WAP TOOKIT 或者是 ERICSSON WAPIDE，都必须进行 WEB 服务器设置，这里我们将常见的几种 WEB 服务器的设置先介绍一下。

1. Windows NT 平台的 IIS

（1）启动菜单-->程序-->Windows NT Option Pack -->IIS4.0。

（2）右击 Internet Information Server 的子项计算机名，在快捷菜单上选择“属性”。

（3）在属性页面的下部，有一个“文件类型（F）”按钮，单击此按钮，会出现文件类型界面。

（4）单击“新增类型”按钮，然后在相关的扩展名栏中填写.wml，在内容类型（MIME）栏中填写 text/vnd.wap.wml。

（5）单击“确定”按钮。

（6）重复 3）、4）、5）三步，再增加其他的 MIME 类型。

2. Windows 98 的 PWS

（1）单击“开始”、“运行”，键入 regedit，打开注册表编辑器程序。

（2）在 HKEY_CLASSES_ROOT 下新建一个主键.wml。

（3）在 HKEY_CLASSES_ROOT\.wml 下新建一个字符串，名称为 Content Type，值为 text/vnd.wap.wml。

（4）在 HKEY_LOCAL_MACHINE\Software\CLASSES\MIME\Database\Content Type 新建一个主键 text/vnd.wap.wml。

（5）在 text/vnd.wap.wml 主键下新建一个字符串，名称为 Extension，值为.wml。

（6）如果需要增加其它的 MIME 类型，重复（2），（3），（4），（5）四步。

（7）重新启动 Windows。

3. Apache Web Server on NT or Solaries or LINUX or OTHER UNIX

（1）不管是 NT 还是 UNIX 或 LINUX，都是修改 Apache 安装目录下的 conf/mime.types 文件。

（2）在该文件中增加以下内容： text/vnd.wap.wml .wml image/vnd.wap.wbmp .wbmp application/vnd.wap.wmlc .wmlc text/vnd.wap.wmls.wmls application/vnd.wap.wmlsc .wmlsc。

（3）存盘。

（4）重新启动 Apache Web Server 即可。

16.2.2 WML 语言基础

WML 编程前的知识准备：

（1）理解 Internet 基本知识，了解 HTTP 协议及其内涵。

（2）熟悉 Web 服务器，并理解 Web 服务器与浏览器之间的交互原理、关系。

（3）了解 HTML 语言规范，熟悉静态网页的设计。

（4）熟悉 JavaScript、XML 语言的设计将更有助于 WML 语言的学习。

目前手机页面可以用 HDML 和 WML 两种语言来描述，在此主要介绍 WML，它本质上是用 XML 1.0 来定义的。学习 WML 的最好方法就是观察例程，我们先以一个简单例子来认识 WML。

由于 WML 语言继承于 XML，所以一个有效的 WML 文档必须包含一个 XML 声明和一个文件类型声明。以下就是一个最常用的声明，由于 WML 语法要求非常严格，为了避免出错，制作者可以直接拷贝粘贴到制作文档。

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
                    "http://www.wapforum.com/DTD/wml_1.1.xml">
```

注意 <?xml version="1.0"?>语句必须出现在一个 Deck 的首行，而且必须顶头写，插入任何字符哪怕是空格都会造成语法错误。

一个简单的 WML 文件：

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <template>
      <do type="prev" label="back">
          <prev/>
          <!--provide a button you can clink to back a step-->
      </do>
  </template>
<card id="friends" title="Hot link">
  <p>
      <a href="http://wap.sian.com.cn/">Sina WAP</a><br/>
      <a href="#nextcard">Next Card</a>
  </p>
</card>
<card id="nextcard">
  <p>
      this is the second card.
```

```
  </p>
</card>
</wml>
```

通过以上示例我们应该了解到以下内容：

（1）语法：WML 的语法与 HTML 相似，仍然是一种标记语言，而且延续了 XML 语法规则。

（2）元素：在 XML 和 WML 语言中，语言的基本元素称之为“标签”。

标签必须被“<”和“>”括起来。大多数标签都包括“起”、“止”两部分，例如： … 某些特殊标签可以只有一个标签，但是必须有结束标记，例如：<prev/>。

（3）属性：XML 语言的标签可以包含很多属性，给标签提供必要的附加信息。

属性内容通常在起始标签内使用，属性只作为参数为标签提供必要的信息，不会被浏览器显示，属性的值需要被引号括起来，可以是单引号或者双引号，引号可以成对嵌套使用。

例如：<card id="friends" title="Hot link">

（4）注释：注释内容是方便制作者阅读源代码，不会被浏览器显示，WML 不支持注释嵌套。

例如：<!-- This is a comment. -->

（5）文档结构：WML 文档是由 Card 和 Deck 构成的，一个 Deck 是一个或多个 Card 的集合。在得到客户终端的请求之后，WML 从网络上把 Deck 发送到客户的浏览器，访问者可以浏览 Deck 内包含的所有 Card，而不必从网上单独下载每一个 Card。

在上面的示例中我们还有一些没有涉及到的基本内容，列举如下：

（1）大小写敏感，无论是标签元素还是属性内容都是大小写敏感的，这一点继承了 XML 的严格特性，任何大小写错误都可能导致访问错误，这是 WML 制作者必须注意的问题。

（2）避开语法检查的方法——CDATA，CDATA 内的数据内容都会被当作文本来处理，从而避开语法检查，直接作为文本显示。

示例：

< ! [CDATA][this ia <b> a test]] >

显示结果为：

this ia <b> a test

（3）定义变量，WML 可以使用变量供浏览器和 Script 使用，通过在 Deck 中的一个 Card 上设置变量，其他 Card 不必重新设置就可以直接调用。

XML 是一种语法非常严格的语言，WML 也继承了这种规则，这为我们深入学习 WML 提供了一个很好的基础。

1. WML 变量

WML 使用 XML 文档字符集，目前支持 Unicode 2.0。WML 的所有标签、属性和规定的可接收值必须小写，CARD 的名字和变量也是区分大小写的；对于连续的空字符，只显示一个空格。标签内属性的值必须用" "或者' '括起来，属性名，“＝”和值之间不能有空格。对于不成对出现的标签，必须在 > 前加 / ，比如
。

在对保留字符的处理中，对应的取代字符有：

```
<       &lt;
```

> >
? ' '
" "
& &
$ $$
空格
-

这里要指出的是在 URL 的传递过程中，用来连接参数的 & 必须转化为 &amp。

2. 变量格式

WML 定义变量，可以让页面设计更简单和富有逻辑性。变量格式如下显示：

```
$identifier
$(identifier)
$(identifier:conversion)
```

圆括号在变量带有空格时使用，第 3 种格式本节后面说明。

变量的优先权最高，所以当出现与变量符号相同的字符时，它将被认为是变量的标志。因此如果想在 WML 页面中显示$符号时必须在其后面再跟一个$符号。如下例：

```
You account has $$1650.00 in it.
```

变量名是由 US-ASCII 码、下划线和数字组成，并且只能以 US-ASCII 码开头。变量名大小写敏感。WML 变量没有类型，均表示为字符串。变量没有被赋值的时候，内容为空，即空字符串。可以在运行过程中改变它的值。

3. 创建变量

创建变量最简单的方法是使用<setvar>元素，语法如下：

```
<setvar
    name="string"
     value="string" />
```

<setvar>只能在<go>、<prev>和<refresh>中使用（具体操作见后）。下例定义了一个叫 vNAME 的变量并赋值为 Jeff：

```
<setvar name="vNAME" value="Jeff" />
```

另外，还可以在<input>、<select>和<postfield>中定义变量，参考任务。

4. 替换文本

变量可以用作替换用途，但只能在文本类型（如显示字符、URL 等）中使用。任何元素和属性都不能使用变量来替代。例如：

```
Hello, $vNAME.
```

将显示：

```
Hello, Jeff.
```

5. ESCAPE 转换

前面说过变量可以用作替换用途，但是在 URL 中使用时，变量的内容必须遵守[RFC2396]标准。这个标准规定某些特殊字符在 URL 里使用的时候必须用特殊表示方法，即 ESCAPE 八

进制表示。例如：

list.asp?id=3

在 URL 中要表示为：

list.asp%3fid=3

所以变量有可能需要标志为是否对其内容进行 ESCAPE 转换，变量定义就有如下几种特殊方式：

效果　表达方式 1　表达方式 2　表达方式 3

对变量中 ESCAPE 字符进行转换　$(var:e)　$(var:E)　$(var:escape)

不进行 ESCAPE 转换　$(var:u)　$(var:U)　$(var:unesc)

变量中没有 ESCAPE 字符　$(var:n)　$(var:N)　$(noesc)

给变量进行 ESCAPE 转换是 WML 的默认方式

6. WML 桌面（DECK）

由于网络带宽以及某些 WAP 服务器 DECK 传输的限制，所以 DECK 越小越好，最好不要超过 1.2K。如果需求很复杂，最好分成几个 DECK 来完成。

一个完整的 WML 文档结构如下：

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
    "http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
    <head>
      .
      . 头信息.....
      .
    </head>
    <template>
      .
      . 模板定义.....
      .
    </template>
    <card>
      .
      . 内容.....
      .
      </card>
      .
      . 其他 card（可有可无）.....
      .
</wml>
```

7. 类型声明

前面已经讲过，DECK 开头必须指明以下的 XML 类型声明：

```
<?xml version="1.0"?>
```

```
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
    "http://www.wapforum.org/DTD/wml_1.1.xml">
```

第一行指出 XML 版本，紧跟的文档类型（DOCTYPE）声明指出所引用的语言标准定义。

8. 标签注释

（1）<wml>。

语法：

```
<wml xml:lang="STRING">
```

<wml>标签和 HTML 中的<html>标签一样，用来表明这是一个 WML 的 DECK。xml:lang 属性来指定文档的语言，比如<wml xml:lang="zh">表示文档语言为中文。

（2）<head>。<head>标签包含了该 DECK 的相关信息。<head>标签之间可以包含一个<access>标签和多个<meta>标签。

<access>语法：

```
<access
      domain="STRING"
      path="STRING" />
```

<access>相当于 HTML 中的<BASE>标签，指定该 DECK 的访问控制信息，它的两个属性是可选的。

domain：指定域，默认值为当前域。

path：指定路径，默认值为"/"，即根目录。

<meta>语法：

```
 <meta
      http-equiv="STRING" | name="STRING"
      forua="true|false"
      content="STRING"
      scheme="STRING" />
```

和 HTML 中<META>的类似，<meta>提供了该 DECK 的 meta 信息。包括三种情况：

- name="name" UP.Link Server 忽略 meta 数据。
- http-equiv="name" UP.Link Server 将 meta 数据转为 HTTP 响应头（同 HTML）。
- user-agent="agent" UP.Link Server 直接将 meta 数据传给手机设备。

content 属性也是必选的，其内容根据属性而定。scheme 属性目前尚不支持。forua 为可选属性，指定在该 wml 文件传到客户端之前，<meta>标签是不是被中间代理删除（因为传输的协议可能改变），默认值为 false。

目前支持的 meta 数据有：

<meta http-equiv="Cache-Control" content="max-age=3600"/>指定 DECK 在手机内存缓存中的存储时间段，默认的为 30 天(除非内存耗尽)，在该期间，手机对于访问过的 DECK 直接从缓存里调用。如果信息是对时间敏感的，可以用 max-age 指定 DECK 在缓存里的生存期，最小单位是秒，如果指定为 0，则每次都需通过连接服务器来调用该 DECK。

<meta user-agent="vnd.up.markable" content="false"/>和<meta user-agent="vnd.up.bookmark" content="指定的 URL"/>类似于普通浏览器的书签功能。当用户将一个 CARD 做了书签后，手机浏览器首先用一个标记记录该 CARD，这个标记默认的是<card>标签中的 title 属性（以后会

讲到），然后当用户选择了该书签以后，浏览器就会打开被记录的 URL。但是因为在默认的情况下，手机会记录所有的 DECK，所以，一般<meta>被用来使手机不要记录当前的 URL，即<meta user-agent="vnd.up.markable" content="false"/>。此外，如果要为书签指定不同于当前 DRECk 的 URL，用<meta user-agent="vnd.up.bookmark" content="指定的 URL"/>。

（3）<template>。<template>元素声明一个 DECK 级的事件/请求，对 DECK 页面中所有 CARD 有效，当然某个 CARD 可以通过定义同名的事件来替代<template>声明中的事件处理。

语法：

```
<template
      onenterforward="STRING"
      onenterbackward="STRING"
      ontimer="STRING" />
```

onenterforward：当用户通过<go>进入 CARD 时调入的链接。

onenterbackward：当用户通过<prev>退回 CARD 时调入的链接。

ontimer：<timer>事件激活时调入的链接。

实例（其中涉及的其他命令参考本节其他部分）：

```
<wml>
<template>
    <do type="options" name="do1" label="default">
    <prev/>
    </do>
</template>
<card id="first">
  <!-该卡片将自动套用模块中定义的事件处理过程-->
            …
          </card>
      <card id="second">
    <!-使用空操作（noop）来屏蔽模块中定义的事件处理过程->
      <do type="options" name="do1">
        <noop/>
        </do>
          …
      </card>
      <card id="third">
    <!-该卡片使用同名的事件处理替代模块中提供的事件处理->
      <do type="options" name="do1" label="options>
        <go href="/options"/>
        </do>
      </card>
      </wml>
```

（4）<card>。一个 DECK 可以包含多个 CARD，每个 CARD 的内容可能不止一屏显示，注意 DECK、CARD 和屏幕显示范围的关系。一个 CARD 用<card>和</card>包含。

语法：

```
<card
```

```
id="STRING"
title="STRING"
newcontext="true|false"
ordered="true|false"
onenterforward="STRING"
onenterbackward="STRING"
ontimer="STRING"
xml:lang="STRING">
```

每个 CARD 元素可以有一个标号（ID）和标题（TITLE）。当然，这都不是必须的。

id：CARD 的名字，在 DECK 中唯一，可用作 URL 已进行跳转。

title：CARD 的标题，用户 BOOKMARK 一个 CARD 时的名字。该属性在某些用户终端上会显示出来。

newcontext：用来指示当跳转到本 CARD 时，用户终端（手机、模拟器等）是不是要清除以前保留的信息如变量、堆栈历史记录、终端状态等。默认值为 FALSE。

ordered：表明该 CARD 里的内容是按固定的顺序显示，还是按用户的选择来显示。默认值是 TRE。这点和 HTML 不同，CARD 页面里的内容可以按一定的顺序显示，默认的是按线性顺序显示，即按代码的顺序，但是，要注意的是，以下三个标签必须按以下顺序来写<onevent> <timer> <do>，（这和以后要讲的“事件”有关）。这样做是为了方便填表单，当 ordered 设置为 true 时，如果一个表单的内容不能在一屏里显示完，就分成多屏显示；当 ordered 设置为 false 时，手机可以显示一个概要 CARD 来总结有效的选项，用户可以从中选取表单选项来填写。

onenterforward：当用户通过<go>进入 CARD 时调入的链接。

onenterbackward：当用户通过<prev>退回 CARD 时调入的链接。

ontimer：<timer>事件激活时调入的链接。

9. 显示内容

CARD 里可以显示像文本、图像这样的内容。像 HTML 一样，WML 也提供一些标记对内容进行排版。

（1）段落与换行标记。<p>：段落标记，用来对段落进行分段。

语法：

```
<p
  aligh="left|right|center"
  mode="wrap|nowrap";
  xml:lang="STRING" >
```

aligh：文字对齐方式，左（left）、右（right）、居中（center）。

mode：文字超出屏幕时是否折行，各种终端处理方式不同。

xml:lang：显示语言编码。

：行分隔标记，产生回车效果。

语法：

```
<br
  xml:lang="STRING" />
```

xml:lang：显示语言编码。

段落和换行的例子：

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
   "http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card title="Paragraphs">
<p>
This is a paragraph
</p>
<p>
This is a another<br/>with a line break
</p>
</card>
</wml>
```

（2）文字样式标记。WML 提供了一系列文字样式的标记，如下表所示。不过 WML 鼓励用户尽量使用和标记，因为某些 WAP 终端会忽略其他标记。

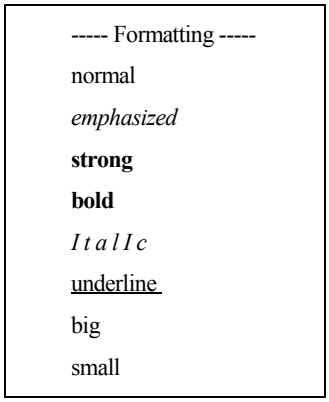
标记	字体样式
<b>	粗体
<big>	大字体
	强调字体
<i>	斜体
<small>	小字体
	加重强调字体
<u>	下划线字体

每个文字样式标记语法都一样，都有一个标志语言代码的 xml:lang 属性。下面是一个例子：

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
    "http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card title="Formatting">
    <p>
      normal<br/>
      <em>emphasized</em><br/>
      <strong>strong</strong>
      <br/> <b>bold</b> <br/>
      <i>italic</i><br/>
      <u>underline</u><br/>
      <big>big</big><br/>
      <small>small</small>
    </p>
```

```
</card>
</wml>
```

显示结果如下：



（3）表格。WML 支持简单的表格标记<table>、<tr>和<td>。语法如下：

```
<table
    align="L|R|C"
    columns="NUMBER"
    TITLE="STRING"
  <tr>
    <td> 内容... </td>
     其他列...
  </tr>
     其他行...
</table>
```

在 WML 里定义一个表格必须先指定列数，即 columns 属性。而在表格里就必须有相应数量的<td></td>标记对。应该注意的是，align（对齐）属性的内容只能是 L（左对齐）、R（右对齐）和 C（居中），跟其他标记不同。

下面的例子演示了表格的基本功能：

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
    "http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card title="SALARY">
<p>
<table columns="2">
<tr>
<td>NAME</td>
<td>PAYED</td>
</tr>
<tr>
<td>TOM</td>
<td>$8888.88</td>
```

```
</tr>
<tr>
<td>Mike</td>
<td>$6666.99</td>
</tr>
</table>
</p>
</card>
</wml>
```

（4）图像。相对于多媒体丰富的 HTML 网站，在资源紧张的 WAP 设备上显示图像就有点问题。但是 WML 还是提供了图像显示的支持，毕竟一个设计精巧的图像会比一段话表达意思更清楚，或许占用空间更小。

WML 支持 WBMP（Wireless Bitmap）格式的图像，需要用特殊工具制作。显示图像使用<img>标记，语法如下：

```
<img
  alt="STRING"
  src="STRING"
  align="top|middle|bottom"
  height="NUMBER"
  width="STRING"
  hspace="NUMBER"
  vspace="STRING"
  localsrc="STRING" />
```

alt：图像无法显示时的替换文字。

src：图像的 url。

localsrc：储存于 ROM（或 RAM）中图像的名字，各种终端支持不同。

align：上下对齐方式。

height：图像显示高度。

width：图像显示宽度。

hspace：图像左右的空白，以 pixel 数或百分比表示。

vspace：图像上下的空白，以 pixel 数或百分比表示。

上面的属性只有 alt 和 src 是必须的。其他附加属性基本上仅仅用来指示用户终端，大多数情况下会被忽略。下面是个例子：

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
   "http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card title="Congratulation">

<p>
WML 图片 <img src="wonderful.wbmp" alt="wonderful"/>!
</p>
</card>
```

```
</wml>
```

10. WML 任务（TASK）

前面我们已经讲过如何在 WML 中显示内容。不过任何程序员都知道，没有结构和进程就没有程序。在 WML 中定义进程有很多方法，最简单的就是任务。

WAP1.1 定义了几种类型的任务，任务通过对事件（有关事件的详细解释见下一章事件）的响应改变程序的运行顺序。有四种 WML 任务：<noop>、<prev>、<refresh>和<go>。

（1）<noop>。这个任务不做任何事情，一般用于屏蔽 DECK 级事件（参见桌面和事件），语法非常简单：

```
<noop/>
```

（2）<prev>。当用户激活该任务时，终端就转回上次用户访问过的 URL。语法如下：

```
<prev>
    <setvar>
    ......
</prev>
```

如果<prev>中包含了<setvar>元素，就会优先处理。下面例子定义了一个只有 Back 按钮的 DECK，按下以后会返回前面看过的页面。

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card>
  <p>
    <anchor>
      Back
      <prev/>
    </anchor>
  </p>
</card>
</wml>
```

（3）<refresh>。当用户激活该任务时，就执行一个刷新过程。如果这个任务里使用<setvar>定义了变量，变量值将被重新设置。语法如下：

```
<prev>
    <setvar>
    ......
</prev>
```

如果当前 CARD 含有<timer>元素，那么在刷新时<timer>优先启动。下面的例子定义在屏幕刷新时重设 firstname，lastname 和 age 变量。

```
<do type="refresh">
  <refresh>
    <setvar name="firstname" value="david">
    <setvar name="lastname" value="smith">
    <setvar name="age" value="29">
```

```
    </refresh>
</do>
```

（4）<go>。当用户激活该任务时，就引导用户去 WML 中指定 URL，可以是服务器上其他的 DECK，也可以是本 DECK 中其他的 CARD。语法如下：

```
<go
    accept-charset="STRING"
    href="URL"
    method="post|get" sendreferer="true|false" >
    <postfield>, <setvar>.....
</go>
```

href：必选属性，指向一个合法 URL。如果是其他的 DECK，则该 DECK 的第一个 CARD 会显示出来。如果是本 DECK 中的其他 CARD，而历史堆栈里保存的是最新数据的话，则堆栈保持不变，直接调入该 CARD。

sendreferer：如果为 true，用户主体信息（USER AGENT）必须传送给 WAP 网关。传送时使用 HTTP 的提交头信息，即尽可能简短的相对 URL。这个属性可以用来给服务器控制存取 URL 的权力。默认值为 false。

Method：值必须为 get 或 post。分别用来产生 HTTP 的 GET 和 POST 请求。若为 get，则在 URL 中列出参数，例如：“http://www.wap86.net/bob.cgi?argone=one”；若为 post，则数据在请求内部传送，不需要在 URL 中列出。

Accept-charset：指定字符集名称列表，服务器在接受<go>的时候必须接受这个编码规则。默认值为 unknown。具体内容这里不作解释。

下面是一个简单的例子：

```
 <go href="../topic.wml" sendreferer="true">
```

（5）其它——<postfield>。<postfield>并不是一个任务，但是跟<go>任务有关，所以在这里介绍。它用来定义“名称/值”对以便通过<go>向服务器发送 HTTP 请求。语法如下：

当用户激活该任务时，就引导用户去 WML 中指定 URL，可以是服务器上其他的 DECK，也可以是本 DECK 中其他的 CARD。语法如下：

下面是一个简单的例子：

```
<postfield name="STRING" value="STRING" />
```

当一个含有<postfield>的任务被执行的时候，终端要完成这样一个过程：

- 识别“名称/值”对并准备参数变量。
- 参数变量转换成正确的字符集。
- 根据 URL 的 ESCAPE 规则对参数进行 ESCAPE 转码，编译成 application/x-www-form-urlencoded 的 MIME 类型。
- 根据 method 指定的请求模式提交任务。

下面的例子演示 get 模式的用法：

```
 <go href="../news.asp" sendreferer="true" method="get">
    <postfield name="newstype" value="technology"/>
    <postfield name="newstext" value="wml"/>
</go>
```

服务器将收到这样的 get 请求：

```
GET ../news.asp?newstype=technology&newstext=wml HTTP/1.1
.
. 其他 HTTP 头信息:
.
```

如果把前面的请求模式改成 post，则同样的<go>任务产生的这样的 post 请求：

```
POST../news.asp HTTP/1.1
content-type="xxx-urlencoded" .
. 其他 HTTP 头信息:
.
newstype=technology&newstext=wml
```

11. 事件（EVENT）

任务不能在真空中生存，它们必须绑定到某个事件上才能做一些有用的事情。事件发生→任务执行，这才是完整的进程控制。有 4 个元素可以帮助用户完成事件对任务的绑定：<anchor>、<onevent>、<timer>和<do>。

<anchor> 链接 <onevent> 固有事件 <timer> 计时器 <do> 用户触发事件 DECK 级事件

（1）<anchor>链接。像 HTML 一样，WML 也可以定义到其他程序的链接。在 HTML 中，链接通常用下划线和特殊颜色的方式表示跟其他内容的区别。在 WAP 终端上则没有严格的规则说如何表示一个链接，一般采用反白的显示方式。

<anchor>的语法如下：

```
<anchor
      title="STRING"
      xml:lang="STRING"
>
   <br>, <go>, <img>, <prev>, <refresh>, TEXT
</anchor>
```

title：链接的标题；

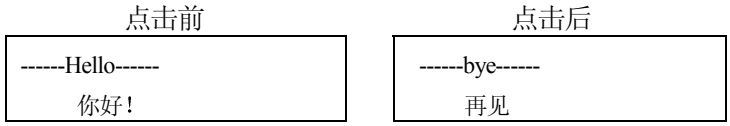
xml:lang：语言编码。

在这种定义方式中，实际上链接一般是通过<anchor>中包括的<go>元素完成的。比如下面的例子定义了 2 个 CARD，点击第一个 CARD 里的 click me 链接会跳转到第 2 个 CARD：

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
   "http ://www.wapforum.com/DTD/wml_1.1.xml">

<wml>
<card id="Hello" title="Hello">
   <p>你好!
   <anchor>click me
      <go href="#bye"/>
   </anchor>
   </p>
</card>

<card id="bye" title="Bye">
```

```
    <p>再见</p>
</card>
</wml>
```

演示效果如下：



链接还有一种短格式，语法如下：

```
<a
      href="STRING"
      title="STRING"
      xml:lang="STRING"
>
    <br>, <img>, TEXT
</a>
```

除了跟原来一样的两个属性以外，多了 href 属性，可以对它直接指定 URL。例如下面的例子定义了一个带有图像的链接：

```
<a title="HotBars" href="www.wap86.net/HotBars.wml">
    <img src="hot.wbmp" alt="Hot Bars" />
</a>
```

（2）<onevent>固有事件。WML 定义了 4 种由用户终端触发的固有事件：

- oneventforward：当用户被<go>任务或其他机制（如一个 SCRIPT 过程）引导到一个 CARD 时触发。
- oneventbackward：当用回被<prev>或其他机制（如在设备上按 BACK 按钮）引导到一个 CARD 时触发。
- ontimer：当 TIMER 计时结束时触发。定义 TIMER 计时器可以使用<timer>元素，见下一节。
- onpick：当用户按下一个<option>选项时触发（可以是选中或取消选择）。

把这些事件绑定到一个任务要使用<onevent>元素，它的语法如下：

```
 <onevent
    type="oneventforward | oneventbackward | ontimer | onpick"
>
    <go>, <noop>, <prev>, <refresh>
</onevent>
```

下面的例子演示了如何使用固有事件在 CARD 调入时清空变量。

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
   "http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card id="card1">
    <onevent type="oneventforward">
```

```
      <refresh>
        <setvar name="firstname" value=""/>
        <setvar name="lastname" value=""/>
      </refresh>
      </onevent>
      <p>
        You have no name!
      </p>
    </card>
    </wml>
```

（3）<timer>计时器。看名字就知道，这是个计时器，它在经过规定的计时时间以后产生一个任务。<timer>计时器只在所属 CARD 里有效：当进入 CARD 时，计时器开始工作；时间一到，触发任务；如果离开 CARD，计时器停止。语法如下：

```
<timer
  name="STRING"
   value="NUMBER"
/>
```

name：可选。指定一个包含计时时间的变量，在计时器开始工作以后，变量的值会逐渐减少。如果这个变量在<timer>之前就已经存在并且赋值为一个非负整数，那后面 value 属性的值会被忽略，直接使用变量里的值作为计时时限。

value：必选。指定计时时间，以 1/10 秒为单位。

下面的例子每隔 1 秒显示不同的文字（当然，读者如果有兴趣改成图像也可以）。

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
    "http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card id="Hello" title="Hello">
    <onevent type="ontimer">
      <go href="#card2">
    </onevnet>
    <timer value="10"/>
    <p>Hello!</p>
</card>

<card id="Take care">
    <onevent type="ontimer" title="Take care">
      <go href="#card3">
    </onevnet>
    <timer value="10"/>
    <p>R U tired?</p>
    </card>

<card id="Rest">
    <onevent type="ontimer" title="Rest">
```

```
        <go href="#card1">
    </onevnet>
    <timer value="10"/>
    <p>Take a rest!</p>
</card>

</wml>
```

还有一个例子，通常用来做网站主界面，显示一段文字以后进入正式内容。

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
    "http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card id="Welcome to BHP" ontimer="main.wml">
    <timer value="30"/>
    <p>
      Hello!
      Welcome to BHP .
    </p>
</card>

</wml>
```

（4）do>用户触发事件。每个 WAP 终端都预定义了一系列用户界面部件，如手机上的按钮、触摸屏上的图表、声音指令或者其他一些很容易识别的部件。WML1.1 定义了下面一些 WAP 兼容终端必须支持的部件。但是要说明的是，只有 prev 有预先定义的功能，其他的只是概念上的定义，需要根据开发者激活并赋予一定的动作。

部件	功能
accept	确认，接受输入
prev	退回历史页面访问堆栈里上一个 CARD
help	上下文关联的帮助信息
reset	重设设备状态
options	上下文关联的选项或附加操作
delete	删除当前内容或选择
unknown	由开发者自己定义

当用户激活这些部件的时候会产生相应的事件。可以使用<do>元素捕获这些事件并对其做出反应。下面是<do>的语法：

```
<do
  type="accept | prev | help | reset | options | delete | unknown"
   lebel="STRING"
   name="STRING"
   optional="true | false"
```

```
    xml:lang="STRING"
>
<go> | <noop> | <prev> | <refresh>
</do>
```

type：必选属性，内容只能是 7 种。

label：用户接口部件的显示标签。如果终端不能显示则被忽略。WML1.1 建议此属性长度限制在 6 个英文字符以内。

name：标志“事件/任务”绑定关系的唯一名称（在 CARD 范围内）。CARD 级的<do>事件替换同名的 DECK 级<do>事件（见下一节）。如果不指定 name 属性或 name 为空字符串，则 name 默认为 type 的类型。

optional：告诉终端此软按钮在屏幕中是否显示。如果此值设为 true，则忽略本<do>元素。默认值为 false。

xml:lang 任务：语言代码。

跟其他“事件/任务”绑定关系一样，<do>里的任务定义必须是<go>、<noop>、<prev>或<refresh>中的一个。

下面例子演示了<do>的使用方法。用户按了不同的按钮会被导向不同的 CARD。

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
    "http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card id="Card1" title="Test Links">
    <do type="accept" label="Links" optional="false">
        <go href="#Links"/>
    </do>
    <do type="help" label="Help" optional="false">
        <go href="#Help"/>
    </do>
   <p> WAP86's perfect links</p>
</card>

<card id="Links" title="Test Links">
    <p> Select one:<br><br>
      <a href="www.bhp.com.cn">希望出版社</a><br>
      <a href="www.microsoft.com">Microsoft</a>
    </p>
</card>
<card id="help" title="Help">
    <do type="accept" label="Links" optional="false">
      <go href="#Links"/>
    </do>
    <p>
        Select "Links" button to view the links.
    </p>
```

```
</card>

</wml>
```

（5）DECK 级事件。前面（DECK 部分，关于<template>）已经讲过可以用<template>元素定义 DECK 级事件，做法跟 CARD 级事件一样，只要在<template>元素里包含<do>或<onevent>事件就可以。这种做法可以定义一些在每个 CARD 里都需要定义的事件，而不需要重复说明。比如下面的例子给所有的 CARD 定义了 BACK 按钮以便返回上一页：

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
    "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>

<template>
    <do type="prev" label="BACK" optional="false">
      <prev/>
    </do>
</template>

<card id="Links" title="My Links">
    <do type="help" label="Help" optional="false">
      <go href="#Help"/>
    </do>
    <p> Select one:<br><br>
      <a href="www.bhp.com.cn">希望出版社</a><br>
      <a href="www.microsoft.com">Microsoft</a>
    </p>
</card>

<card id="help" title="Help">
    <do type="accept" label="Links" optional="false">
      <go href="#Links"/>
    </do>
     <p>
       Select "Links" button to view the links.
   </p>
</card>

</wml>
```

12. 数据输入（Data Input）

程序允许用户输入数据才能发挥更多的作用，就是简单如选项选择也会给用户带来很大的方便。WML 也提供了各种各样的数据输入机制：

<input> 简单文本输入 <fieldset> 复合数据输入 <select>、<option> 选择 <optgroup>复选框

（1）<input>简单文本输入。输入数据最简单的途径是<input>，它允许用户输入字符串（可以进行一些规格化操作），并把输入结果保存到一个变量里。用户具体的输入方式跟所使用的终端有关。

<input>的语法如下：

```
<input
    emptyok="true | flase"
    format="STRING"
    maxlength="NUMBER"
    name="STRING"
    size="NUMBER"
    tabindex="NUMBER"
    title="STRING"
    type="text | password"
    value="STRING"
    xml:lang="STRING"
/>
```

emptyok：如果设为 true，则用户可以不输入任何字符。一般情况下带有 format 属性的 input 要求用户必须按照一定格式输入数据（有时不允许输入空字符串，比如日期），所以此属性默认值为 false。

format：输入掩码，用来规定用户输入格式，具体见下表。

maxlength：输入字符串的最大长度，默认为无限。

name：必选。存储输入数据的变量名称。如果该变量已经定义并且其值符合输入格式，则变量的值直接作为 input 的默认值，否则取 value 属性的值作为默认值。

size：输入区的宽度。

tabindex：CARD 中输入元素的 tab 顺序号。

title：输入区的提示字符串。

type：输入类型，默认为 word。如果为 password，则用户输入字符显示成“＊”。

value：input 的默认值，如果 name 指定的变量可用，本属性被忽略。

xml:lang：语言编码。

掩码表

掩码	允许输入的字符
A	大写字母和标点符号
a	小写字母和标点符号
N	数字
X	大写字母、标点符号和数字
x	小写字母、标点符号和数字
M	所有字符，但是所有字母转换成大写
m	所有字符，但是所有字母转换成小写
\c	直接显示 c 所代表的字符，不需输入
*f	f 代表上面掩码中的一个，字符数不限
nf	f 代表上面掩码中的一个，重复 n 次，
n	取值 1~9

下面的例子演示了简单的用户输入：

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card title="User Login">
```

```
  <p>
    User Name:
    <input name="user" maxlength="10" tabindex="1" /><br/>
    Country:
    <input name="country" maxlength="2" emptyok="true"
      value="CN" tabindex="2" /><br/>
    Password:
    <input name="password" maxlength="8"
      type="password" tabindex="3" /><br/>
  </p>
</card>

</wml>
```

例 2 演示了掩码的用法：

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card title="User Login">
  <p>
    用户名:
    <input name="user" maxlength="10" tabindex="1"
      format="X*M" /><br/>
    出生日期:
    <input name="birthday" maxlength="2" emptyok="true"
      tabindex="2" format="NNNN\-NN\-NN" value="1900-00-00"/><br/>
    密码:
    <input name="password" maxlength="8"
      type="password" tabindex="3" format="8m" /><br/>
  </p>
</card>

</wml>
```

演示效果如下：

输入前

```
--- User login ----
  用户名:[]
  出生日期:
  [1900-00-00]
  密码:[]
```

输入后

```
--- User login ----
  用户名:[John]
  出生日期:
  [1980-06-01]
  密码:[ ******]
```

（2）<fieldset>复合数据输入。一般来说，我们很少在一个 CARD 里放很多数据输入元素。因为大多数 WAP 终端屏幕很小，不能在一个屏幕上显示很多输入区。所以，终端必须决定如何处理用户输入界面，比如元素是否应该显示成上下滚动的列表。也许一个页面上显示一个输入是最直观的方式。每个用户终端有不同的方式处理这个问题。其实 WML 已经提供了两种途径来达到最佳效果。

第一个途径就是 CARD 元素的 ordered 属性。通常，ordered 设为 true，表示 CARD 里的内容表现为一个线性列表，也许会显示很多页，就像前面那个例子。但是如果我们把 ordered 设为 false，则某些终端会把输入列表先列出来，如果输入项目不是很多的话（比如 email，包括地址、主题和内容），可以在一个屏幕上显示出来。但是非常可惜，目前国内市面销售的手机几乎对此都不支持。

如果碰到更复杂的情况，那就要用到第二种方法：<fieldset>元素。它可以帮助用户组织一系列文字和输入框，可以形成多个组。它告诉用户终端这些元素之间的关系以便更好地显示并引导用户输入。这种做法看起来有点像虚拟的页面，可以让用户分层输入。

<fieldset>的语法如下：

```
<fieldset
    title="STRING"
    xml:lang="STRING"
    <a>, <anchor>, <do>, <img>, 数据输入元素, 文本
</fieldset>
```

title：用来在用户终端上显示的组名称。

xml:lang：语言编码。

<fieldset>有两点需要注意：（1）可以嵌套。（2）只能使用上面所列出的几种元素类型。

下面例子演示了使用<fieldset>进行数据输入：

```
 <?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card ordered="false">

  <do type="accept">
    <go href="register.asp" method="post">
      <postfield name="fname" value="$fname"/>
      <postfield name="lname" value="$lname"/>
      <postfield name="address" value="$address"/>
      <postfield name="city" value="$city"/>
      <postfield name="province" value="$province"/>
      <postfield name="country" value="$country"/>
      <postfield name="postno" value="$postno"/>
      <postfield name="phone" value="$phone"/>
      <postfield name="fax" value="$fax"/>
      <postfield name="email" value="$email"/>
     </go>
  </do>

  <p>

    <fieldset title="Name">
    First Name:
```

```
        <input name="fname" maxlength="10" emptyok="false"/><br/>
      Last Name:
        <input name="lname" maxlength="10" emptyok="false"/><br/>
    </fieldset>

      <fieldset title="Address">
      Address:
       <input name="address" maxlength="50" emptyok="false"/><br/>
      city:
       <input name="city" maxlength="30" emptyok="false"/><br/>
      province:
       <input name="province" maxlength="30" emptyok="false"/><br/>
      country:
       <input name="country" maxlength="30" emptyok="false"/><br/>
      Post NO:
       <input name="postno" maxlength="6" emptyok="false"/><br/>
      </fieldset>

      <fieldset title="Contact">
      Phone:
       <input name="phone" maxlength="30" emptyok="false"/><br/>
      Fax:
       <input name="fax" maxlength="30"/><br/>
      Email:
       <input name="email" maxlength="30" emptyok="false"/><br/>
      </fieldset>

    </p> </card>
</wml>
```

（3）<select>、<option>选择。在 WML 中可以定义输入选项。选项可以嵌套，可以定义默认值，也可以在用户做出选择时触发任务。基本的选项列表用<select>定义，语法如下：

```
<select
      iname="STRING"
      ivalue="STRING"
      multiple="true | false"
      name="STRING"
      tabindex="NUMBER"
      title="STRING"
      value="STRING"
      xml:lang="STRING"
>
      <option>, <optgroup>
</select>
```

iname：保存用户选择索引值的变量。索引值表示选项在<select>中的顺序，从 1 开始计算。0 代表用户没有选择。

ivalue：保存默认选择索引值的变量。只有 iname 属性没有指定的时候，这个属性才起作

用。

name：保存用户选择文本的变量。

value：保存默认选择文本的变量。只有 name 属性没有指定的时候，这个属性才起作用。

multiple：是否允许多重选择。如果为 true，用户就可以一次选择多个选项。这时候<select>也会在 name 指定的变量里保存多个选择，用分号（；）分开，同样也适用于 iname。

title：用来显示的标题。

tabindex：CARD 中输入元素的 tab 顺序号。

xml:lang：语言编码。

选项内的单独选择项用<option>定义，语法如下：

```
<option
    onpick="URL"
    title="STRING"
    value="STRING"
    xml:lang="STRING"
>
    文本, <onevent>
</option>
```

onpick：<option>被选择或取消选择时转向的地址。对于多重选择来说，所有<option>都被选择或取消选择时激发 onpick 事件。

title：用来显示的标题。

value：返回给<select>中 name 属性的文字。

xml:lang：语言编码。

下面的例子演示了<select>、<option>的用法：

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card id="maincard" title="About you">
  <do type="accept">
    <go href="register.asp" method="post">
      <postfield name="sex" value="$sex" />
      <postfield name="income" value="$income" />
      <postfield name="hobbies" value="$hobbies" />
    </go>
  </do>
  <p>
    Tell me about you:
      <select>
      <option onpick="#sexcard">Sex: $sex</option>
      <option onpick="#incomecard">Income: $income</option>
      <option onpick="#hobbiescard">Hobbies: $hobbies</option>
    </select>
  </p>
```

```
</card>

<card id="sexcard" title="Sex">
  <p>
    What's your sex?
    <select name="sex">
      <option value="Male">Male</option>
      <option value="Female">Female</option>
    </select>
  </p>
</card>

<card id="incomecard" title="Income">
  <p>
    How much do you earn?
    <select name="income">
      <option value="0-1000">0-1000</option>
      <option value="1001-5000">1001-5000</option>
      <option value="5001-10000">5001-10000</option>
      <option value="over 10000">over 10000</option>
    </select>
  </p>
</card>

<card id="hobbiescard" title="Hobbies">
  <p>
    What's your hobbies?
    <select name="hobbies" multiple="true">
      <option value="Computer">Computer</option>
      <option value="Sport">Sports</option>
      <option value="Book">Book</option>
      <option value="Entertainment">Entertainment</option>
    </select>
  </p>
</card>
</wml>
```

（4）<optgroup>复选框。<optgroup>更进一步，可以对<option>进行分组，语法如下：

```
<optgroup
    title="STRING"
    xml:lang="STRING"
>
    <optgroup>, <option>
</optgroup>
```

title：用来显示的标题。

xml:lang：语言编码。

下面的例子把前面例子里的 hobbiescard 作了改动，提供更多的选择内容：

```
<card id="hobbiescard" title="Hobbies">
<p>
        What's your hobbies?
        <select name="hobbies" multiple="true">
          <optgroup title="Computer">
             <option value="Software">Software</option>
             <option value="Hardware">Hardware</option>
          </optgroup>
          <optgroup title="Sports">
             <option value="Swimming">Swimming</option>
             <option value="Skiiing">Skiiing</option>
             <option value="Bicycle">Bicycle</option>
             <option value="Tennis">Tennis</option>
         </optgroup>
         <optgroup title="Book">
             <option value="Literature">Literature</option>
             <option value="Technology">Technology</option>
          </optgroup>
         <optgroup title="Entertainment">
             <option value="Films">Films</option>
             <option value="TV">TV</option>
             <option value="Dancing">Dancing</option>
             <option value="Bar">Bar</option>
          </optgroup>
    </select>
    </p>
</card>
```

13. 速查表

（1）WML 字符实体表。

文本替换字符

显示效果	说明	实体表达式	实体 ASC 编码
&	&(=and)符号	&amp;	&#38;
'	单引号	&apos;	&#39;
>	大于号	&gt;	&#62;
<	小于号	&lt;	&#60;
	空格		
"	双引号	&quot;	&#34;
-	连字号	&shy;	&#173

URL ESCAPE 转换字符（标准：RFC2396）

RESERVED		UNWISE		DELIMITERS	
;	%3b	{	%7b	<	%3c
/	%2f	}	%7d	>	%3e

（续表）

RESERVED		UNWISE		DELIMITERS	
?	%3f	|	%7c	#	%23
:	%3a	\	%5c	%	%25
@	%40	^	%5e	"	%22
&	%26	[	%5b		
=	%3d	]	%5d		
+	%2b	`	%27		
$	%24				
,	%2c				
空格	%20				

（2）WML 标签速查表。

结构相关标签

结构相关标签	语法及属性
<wml>	<wml xml:lang="lang" > content </wml>
<card>	<card id="name" title="label" newcontext="boolean" style="style" onenterforward="url" onenterbackward="url" ontimer="url" > content </card>
<template>	<template onenterforward="url" onenterbackward="url" ontimer="url" > content </template>
<head>	<head> content </head>
<access>	<access domain="domain" path="path" />
<meta>	<meta name="name"|http-equiv="name" content="value"

任务相关标签

任务相关标签	语法及属性
<timer>	<timer name="variable" value="value" />
<setvar>	<setvar name="name" value="value" />
<anchor>	
<a>	 task text
<do>	<do type="type" label="label" name="name" optional="boolean" > task </do>
<onevent>	<onevent type="type" > task </onevent>
<go>	<go href="url" sendreferer="boolean" method="method" accept-charset="charset" content</go>
<prev>	<prev>content</prev>
<noop>	<noop/>
<refresh>	<refresh>content</refresh>

输出效果标签

输出效果标签	语法及属性
<img>	
<table>	<table align="alignment" title="label" columns="n"/>
<td>	<td>content</td>
<tr>	<tr><td>content</td></tr>
	text
<big>	<big>text</big>
	text
	text
<p>	<p align="alignment" mode="wrapmode" />
<small>	text
	text
<u>	<u>text</u>

控件相关标签

控件相关标签	语法及属性
<input>	<input name="variable" title="label" type="type" value="value" default="default" format="specifier" emptyok="boolean" size="n" maxlength="n" tabindex="n" />
<select>	<select title="label" multiple="boolean" name="variable" default="default" iname="index_var" ivalue="default" tabindex="n" > content </select>
<option>	<option title="label" value="value" onpick="url" > content </option>
<optgroup> <optgroup title="label" > content </optgroup>	
<fieldset> <fieldset title="label"> content </fieldset>	
	ontimer="url" > content </card>

16.2.3 实现 WAP 服务

1. WMLScript 语言简介

WMLScript 语言和 JavaScript 语言非常相似，不同之处在于 WMLScript 语言必须放在一

个 WMLS 的文件里面，并不像 JavaScript 那样可以镶嵌在 HTML 中，并且 WMLS 文件的大小也不要超过 1.4k。

WMLScript 语法规则如下：

- 每一行程序均以分号结尾。
- 注释方法：“//”或者“/* */”。
- 使用关键字 var 来声明变量。

从上面我们就可看出来，WMLScript 语言和 JavaScript 语言是多么相似，WMLScript 的变量类型有：Boolean，Interger，Floating-point，String，Invalid，WMLScript 的流程控制语言和 JavaScript 的相同，实例如下：

```
// if-else
if (x ==
   x = x * 3.25;
}else{
      x = 0;
}
// for loop
for (var counter = 1; counter < 500; counter
      var i = counter * 1.05;
      somefunction(i);
};
// while loop

while (i >
      i--;
};

// break

for (var counter = 1; counter < 500; counter
      var i = counter * 1.05;
      if (counter == 250) break;
      somefunction(i);
};

//continue
for (var counter = -100; counter < 100; counter
      if (counter == 0) continue;
      var x = 350/counter;
};
```

2. 函数

WMLScript 的函数使用方法如下：

```
extern function identifier(FormatParameterList) Block ;
```

extern 是说明函数是放在另外一个文件当中的。如下：

```
function RunTime(distance,
    var time = distance / speed;
    return time;
};
```

此例输入 distance 以及 speed 两个参数，然后返回 time 的值。

如果调用的是 WMLScript 的内建函数，就必须加上这个函数所属的类名，例如调用 String 类的 length()函数，使用：

```
var a = String.length('just a test');
```

3. 内建函数

WMLScript 有六大内建函数：

- Long：包含数据形态、绝对值、随机数等。
- Float：浮点数处理。
- String：字符串长度（length）、字符位置（charAt）等字串处理。
- URL：getReferer，getHost。
- WMLBrowser：go，prev，next，refresh 等浏览处理。
- Dialogs：弹出（prompt，confirm，alert）等对话框。

首先我们先建立一个 WML 文件，WMLScriptExample.wml，内容如下：

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card id="stuid" title="stuid:">
<do type="accept" label="Results">
    <go href="WMLScriptExample.wmls#checkid($(stuid))"/>
</do>
<p>
    Enter Stu_ID: <input type="text" name="stuid"/>
</p>
</card>

<card id="Results" title="Results:">
<p>
    You entered:<br/>
    Stu_ID: $(stuid)<br/>
</p>
</card>

</wml>
```

让用户输入学号，go 的 href 连接到另外一个.wmls 文件，调用 checkid 函数。再建立一个 WMLScriptExample.wmls 文件，内容如下：

```
extern function
    if (String.length(stuid) !=
```

```
        WMLBrowser.setVar("stuid", "Error: String must be 8 digits long.");
    }
    WMLBrowser.go("WMLScriptExample.wml#Results");
};
```

这个 WMLScript 就是检查 stuid 字串长度是否为 8，并且返回结果。WMLBrowser.setVar() 设定变量的值，WMLBrowser.go() 将手机定位到下一个 WML 文件，也就是 WMLScriptExample.wml 的 Results 卡片。

4. Server 端 ASP 生成 WAP 网页

这里我们将详细解释如何实现 Server 端 ASP 生成 WAP 网页，实现 WAP 服务。

实现步骤：

第一步：先设计数据库 MyDatabase，建立新的表 MyTABLE。

第二步：建立 ODBC 数据源，命名为 MyDatabase。

第三步：ASP 动态生成 WAP 页面。

（1）首先我们要建立一个 ODBC 源 MyDatabase 数据库 ,其中有一个表 MyTABLE，其中有三个字段（title，author，price）。我们用 Access 来建立数据库，具体操作见相关书籍，当然也可以让身边的“高手”帮助完成。

（2）在 ASP 中可以使用多种方式如 ADO，RDO 等等，使 ASP 脚本能访问数据库，我们采用 ADO 通过 ODBC 来访问数据库。因此需要在控制面板里建立 ODBC 源（即 System DSN），命名为 MyDatabase。具体过程不再赘述。

（3）ASP 程序生成 WML 文件。

用服务器端语言产生动态的 WML 页面，和动态产生 Web 网页的编程类似，只是输出的格式按 WAP 网页的规范输出，即遵循 WML 的规范。另外，由于 WAP 手机的内存小，每次输出的页面的大小要有所限制。我们将文件存为 ASP2WML.ASP。

```
<%
<!-- get the wml head -- >
Response.ContentType = "text/vnd.wap.wml;charset=GB2312"
Response.write("<?xml version=""1.0""?><!DOCTYPE wml PUBLIC ""-//WAPFORUM//DTD WML 1.1//EN""
""http://www.wapforum.org/DTD/wml_1.1.xml""><wml><card><p>")
Response.write("<em>ASP2WML</em>")

<!--connect to database and execute the query -- >
set Conn = Server.CreateObject("ADODB.Connection")
Conn.Open( " MyDatabase " )
sql = "select * from MyTABLE "
set RS = Conn.Execute(sql)

%>
查询的结果如下：<i> <%= sql %></i>
<table columns="4">

<%
```

```
i=0
while ( not RS.EOF and i <= 4 )
i=i+1
%><TR>
<TD>
<u><%=RS(1)%></u></TD>
<TD><%=RS(2)%></TD>
<TD><%=RS(3)%></TD>
</TR>
<%RS.MoveNext;
wend

RS.close
Conn.close

Response.write("</table>")
Response.write( "It's the end.")

<!-- get the end tag of WML-- >
Response.write("</p></card></wml>")
%>
```

设置好 Web 服务器，将程序放在服务器上的 Web 目录下，运行 WAP 模拟器，就可以享受方便而时尚的 WAP 服务了。

16.3 小　　结

利用 WAP 实现移动设备访问 Web 和进行信息共享是当前的技术应用趋势，而基于 XML 的 WML 为开发移动 Web 服务提供了一个非常好的技术环境。掌握了 WML 的程序编写规范与方法，就相当于把握住了 Web 发展的未来。

第十七章　XML 与 Java

本章介绍 XML 与 Java 技术的结合，作为网络应用与技术处理的关键技术，XML 与 Java 是并行共生的，代表着 Web 的发展方向。我们首先对 Java 技术进行简单了解，接着讨论 XML 与 Java 的联合开发，并描述了新兴的 JSP 技术与 XML 的协同编程。

本章包括以下内容：

- Web 技术双子星座——XML&Java
- 用 Java 创建 XML 文档
- JSP+XML 平台

17.1　Web 技术双子星座——XML&Java

17.1.1　Java 体验

1. Java 技术背景和发展

1991 年，SUN MicroSystem 公司的 Jame Gosling，Bill Joe 等人，为在电视、控制烤面包箱等家用消费类电子产品上进行交互式操作而开发了一个名为 Oak 的软件（即一种橡树的名字），但当时并没有引起人们的注意，直到 1994 年下半年，Internet 的迅猛发展，万维网 WWW 的快速增长，促进了 Java 语言研制的进展，使得它逐渐成为 Internet 上受欢迎的开发与编程语言，一些著名的计算机公司纷纷购买了 Java 语言的使用权，如 Microsoft，IBM，Netscape，Novell，Apple，DEC，SGI 等，因此，Java 语言被美国的著名杂志 PC Magazine 评为 1995 年十大优秀科技产品（计算机类就此一项入选），随之大量出现了用 Java 编写的软件产品，受到工业界的重视与好评，认为 Java 是八十年代以来计算机界的一件大事，微软总裁比尔·盖茨在悄悄地观察了一段时间后，不无感慨地说：“Java 是长时间以来最卓越的程序设计语言”，并确定微软整个软件开发的战略从 PC 单机时代向着以网络为中心的计算时代转移，而购买 Java 则是他的重大战略决策的实施部署。因此，Java 的诞生必将对整个计算机产业发生深远的影响，对传统的计算模型提出了新的挑战。

SUN MicroSystem 公司的总裁 Scott McNealy 认为 Java 为 Internet 和 WWW 开辟了一个崭新的时代。

万维网 WWW 的创始人 Berners-Lee 说：“计算机事业发展的下一个浪潮就是 Java，并且将很快会发生的”。

Microsoft 和 IBM 两大公司都在 Internet 上销售用 Java 编写的软件。Apple、HP、IBM、Microsoft，Novell，SGI，SCO，Tandem 等公司均计划或已经将 Java 并入各自开发的操作系统，而负责开发并推广 Java 技术的 SunSoft 公司（这是 SUN 下属的一个子公司），将通过颁发许可证的办法来允许各家公司把 Java 虚拟机和 Java 的 Applets 类库嵌入他们开发的操作系

统，这样各类开发人员就能更容易地选择多种平台来使用 Java 语言编程，不同的用户也就可以脱离 Web 浏览器来运行 Java 应用程序，这无疑是很受广大用户欢迎的，也为 Java 语言的应用开拓了极为广阔的前景（当然，各类 JavaOS 之间的兼容性必须得到重视，好在 JavaSoft 已保证将监督这种兼容性）。

2. Java 语言对软件开发技术的影响

Java 语言将对未来软件的开发产生影响，可从如下几个方面考虑：

（1）软件的需求分析：可将用户的需求进行动态的、可视化描述，以提供设计者更加直观的要求。而用户的需求是各色各样的，不受地区、行业、部门、爱好的影响，都可以用 Java 语言描述清楚。

（2）软件的开发方法：由于 Java 语言的面向目标的特性，所以完全可以用面向对象的技术与方法来开发，这是符合最新的软件开发规范要求的。

（3）Java 语言的动画效果远比 GUI 技术更加逼真，尤其是利用 WWW 提供的巨大动画资源空间，可以共享全世界的动态画面的资源。

（4）软件最终产品：用 Java 语言开发的软件可以具有可视化、可听化、可操作化的交互、动画与动作，可以自由停止和播放，而这是在电影与电视播放过程中难以做到的。

（5）其它：使用 Java 语言对开发效益、开发价值都有比较明显的影响。

3. Java 的特点

Java 是一个广泛使用的网络编程语言，它是一种新的计算概念。首先，作为一种程序设计语言，它简单、面向对象、不依赖于机器的结构，具有可移植性、鲁棒性、安全性、并且提供了并发的机制，具有很高的性能。其次，它最大限度地利用了网络，Java 的小应用程序（applet）可在网络上传输而不受 CPU 和环境的限制。另外，Java 还提供了丰富的类库，使程序设计者可以很方便地建立自己的系统。

（1）简单性。Java 语言是一种面向对象的语言，它通过提供最基本的方法来完成指定的任务，只需理解一些基本的概念，就可以用它编写出适合于各种情况的应用程序。Java 略去了运算符重载、多重继承等模糊的概念，并且通过实现自动垃圾收集大大简化了程序设计者的内存管理工作。另外，Java 也适合于在小型机上运行，它的基本解释器及类的支持只有 40KB 左右，加上标准类库和线程的支持也只有 215KB 左右。

（2）面向对象。Java 语言的设计集中于对象及其接口，它提供了简单的类机制以及动态的接口模型。对象中封装了它的状态变量以及相应的方法，实现了模块化和信息隐藏；而类则提供了一类对象的原型，并且通过继承机制，子类可以使用父类所提供的方法，实现了代码的复用。

（3）分布性。Java 是面向网络的语言。通过它提供的类库可以处理 TCP/IP 协议，用户可以通过 URL 地址在网络上很方便地访问其它对象。

（4）鲁棒性。Java 在编译和运行程序时，都要对可能出现的问题进行检查，以消除错误的产生。它提供自动垃圾收集来进行内存管理，防止程序员在管理内存时容易产生的错误。通过集成的面向对象的例外处理机制，在编译时，Java 提示出可能出现但未被处理的例外，帮助程序员正确地进行选择以防止系统的崩溃。另外，Java 在编译时还可捕获类型声明中的许多常见

错误，防止动态运行时不匹配问题的出现。

（5）安全性。用于网络、分布环境下的 Java 必须要防止病毒的入侵。Java 不支持指针，一切对内存的访问都必须通过对象的实例变量来实现，这样就防止程序员使用“特洛伊”木马等欺骗手段访问对象的私有成员，同时也避免了指针操作中容易产生的错误。

（6）体系结构中立。Java 解释器生成与体系结构无关的字节码指令，只要安装了 Java 运行时系统，Java 程序就可在任意的处理器上运行。这些字节码指令对应于 Java 虚拟机中的表示，Java 解释器得到字节码后，对它进行转换，使之能够在不同的平台运行。

（7）可移植性。与平台无关的特性使 Java 程序可以方便地被移植到网络上的不同机器。同时，Java 的类库中也实现了与不同平台的接口，使这些类库可以移植。另外，Java 编译器是由 Java 语言实现的，Java 运行时系统由标准 C 实现，这使得 Java 系统本身也具有可移植性。

（8）解释执行。Java 解释器直接对 Java 字节码进行解释执行。字节码本身携带了许多编译时信息，使得连接过程更加简单。

（9）高性能。和其它解释执行的语言如 BASIC、TCL 不同，Java 字节码的设计使之能很容易地直接转换成对应于特定 CPU 的机器码，从而得到较高的性能。

（10）多线程。多线程机制使应用程序能够并行执行，而且同步机制保证了对共享数据的正确操作。通过使用多线程，程序设计者可以分别用不同的线程完成特定的行为，而不需要采用全局的事件循环机制，这样就很容易地实现网络上的实时交互行为。

（11）动态性。Java 的设计使它适合于一个不断发展的环境。在类库中可以自由地加入新的方法和实例变量而不会影响用户程序的执行。并且 Java 通过接口来支持多重继承，使之比严格的类继承具有更灵活的方式和扩展性。

4. Java 语言基础

如果读者对 Java 非常熟悉，那么这部分内容可以略过。

下面我们先介绍两个简单的 Java 程序，并对其进行分析。在此过程中我们可以了解一些 Java 语言的设计基础知识。

例 17.1

```
public class HelloWorldApp {                    //an application
                public static void main (String args[ ]){
                System.out.println("Hello World!");
                                }
    }
```

本程序的作用是输出下面一行信息:

```
Hello World!
```

程序中，首先用保留字 class 来声明一个新的类，其类名为 HelloWorldApp，它是一个公共类（public）。整个类定义由大括号{}括起来。在该类中定义了一个 main()方法，其中 public 表示访问权限，指明所有的类都可以使用这一方法；static 指明该方法是一个类方法，它可以通过类名直接调用；void 则指明 main()方法不返回任何值。对于一个应用程序来说，main()方法是必需的，而且必须按照如上的格式来定义。Jave 解释器在没有生成任何实例的情况下，以 main()作为入口来执行程序。Jave 程序中可以定义多个类，每个类中可以定义多个方法，但是最多只能有一个公共类，main()方法也只能有一个，作为程序的入口。main()方法定义中，

括号()中的 String args[ ]是传递给 main()方法的参数，参数名为 args，它是类 String 的一个实例，参数可以为 0 个或多个，每个参数用“类名参数名”来指定，多个参数间用逗号分隔。在 main()方法的实现（大括号中），只有一条语句：

```
System.out.println ("Hello World!");
```

它用来实现字符串的输出，这条语句实现与 C 语言中的 printf 语句和 C++中 cout＜＜语句相同的功能。另外，//后的内容为注释。

现在我们可以运行该程序。首先把它放到一个名为 HelloWorldApp.java 的文件中，这里，文件名应和类名相同，因为 Java 解释器要求公共类必须放在与其同名的文件中。然后对它进行编译：

```
C:\>javac HelloWorldApp.java
```

编译的结果是生成字节码文件 HelloWorldApp.class。最后用 java 解释器来运行该字节码文件:

```
C:\>java HelloWorldApp
```

结果在屏幕上显示 Hello World!

我们再来看下面的一个例子：

例 17.2

```
import java.awr.*;
import java.applet.*;
public class HelloWorldApplet extends Applet {          //an applet
    public void paint(Graphics g){
    g.drawString ("Hello World!",20,20);
    }
}
```

这是一个简单的 Applet（小应用程序）。程序中，首先用 import 语句输入 java.awr 和 java.applet 下所有的包，使得该程序可能使用这些包中所定义的类，它类似于 C 中的#include 语句。然后声明一个公共类 HelloWorldApplet，用 extends 指明它是 Applet 的子类。

在类中，我们重写父类 Applet 的 paint()方法，其中参数 g 为 Graphics 类，它表明当前作画的上下文。在 paint()方法中，调用 g 的方法 drawString()，在坐标（20,20）处输出字符串“Hello World!”，其中坐标是用像素点来表示的。

这个程序中没有实现 main()方法，这是 Applet 与应用程序 Application（如例 1）的区别之一。为了运行该程序，首先我们也要把它放在文件 HelloWorldApplet.java 中，然后对它进行编译：

```
C:\>javac HelloWorldApplet.java
```

得到字节码文件 HelloWorldApplet.class。由于 Applet 中没有 main()方法作为 Java 解释器的入口，我们必须编写 HTML 文件，把该 Applet 嵌入其中，然后用 appletviewer 来运行，或在支持 Java 的浏览器上运行。它的＜HTML＞文件如下：

```
<HTML>
    <HEAD>
    <TITLE> An Applet </TITLE>
    </HEAD>

    <BODY>
```

```
<applet code="HelloWorldApplet.class" width=200 height=40>
</applet>
</BODY>
</HTML>
```

其中用<applet>标记来启动HelloWorldApplet, code指明字节码所在的文件，width和height指明 applet所占的大小，我们把这个HTML文件存入Example.html，然后运行：

```
C:\>appleviewer Example.html
```

这时屏幕上弹出一个窗口，其中显示Hello World!。

从上述例子中可以看出，Java程序是由类构成的，对于一个应用程序来说，必须有一个类中定义 main()方法，而对 applet 来说，它必须作为 Applet 的一个子类。在类的定义中，应包含类变量的声明和类中方法的实现。Java 在基本数据类型、运算符、表达式、控制语句等方面与 C、C++基本上是相同的，但它同时也增加了一些新的内容，我们这里只是使大家对 Java 程序有一个初步的了解。具体的语言参考可以查看“浩如烟海”的Java技术书籍。

17.1.2 完美的结合：XML与Java技术

“我们正在利用 XML 和 Java 进行开发，两者真是完美的补充!”

——David Skok, SilverStream 软件公司主席和创始人

“缺少 XML 的 Java 战略是不全面的。”

——Eric Brown，剑桥大学 Forrester 研究中心分析研究员

“XML 不能单独使用；它是对 Java 的完善 Java 为 XML 提供易于使用的代码，XML 为 Java 提供数据。”

——Nancy Lee, Sun 公司 XML 产品经理

推动 Java 的 Sun 公司承认没有 Java 虚拟机 XML 也能与客户端通讯，但是它认为 XML 需要 Java 来发挥它的潜力。

“XML 能在不同平台间交换信息，它不会与跨平台的应用程序混淆不清。”

——Dave Wascha，微软公司 XML 产品经理

XML 能创建不依赖于平台、语言或限制性格式化协定的开放数据。如果广泛地被采用，XML 能变成为内容以及下至客户端对象通讯的广泛标准。这似乎十分类似 Java，至少作为内容平台。XML吸引了对Java不是很感兴趣的微软公司。

XML 在许多方面增强了 Java；然而，XML 也发展了一种对象传输协议，该协议与 Java 声称的跨平台性格格不入。XML 将基于网络的信息置标化，使得开发者和电脑易于辨认。这是有必要的，因为 HTML 除了隐藏的 URL 外缺少有效的途径来说明内容的含义。XML 的目标就为网络的对象添加那些含义，而这曾经就是Java的任务之一。

许多网络开发者得出结论：XML 和 Java 是完美的一对，因为彼此十分相辅相成。XML 有助于独立平台、易提取信息的文档和数据。Java 有助于独立平台、易于处理的面向对象的应用软件解决方案。

能驱动 XML 广泛接受的应用是在 HTML 的限制下难以实现的那些应用。可分为下列的四大类：

- 需要网络客户端协调多种不同类型数据库的应用。

- 力图将比例可观的处理工作从服务器端分布于客户端的应用。
- 要求客户端将同一数据以不同的形式展现给不同的使用者的应用。
- 智能网络代理应个别用户的需要搜索定制信息的应用。

XML 与 Java 技术完美地互补，为开发者创造了一个可能性的新世界。XML 被称为自 Java 技术横空出世以来 Internet 应用领域最大的新闻。

很难想象比它们两者更为互补的技术了：Java 平台提供了在网络上安全而方便地传播代码的基础，XML 技术则为数据提供了同样的能力，一种清晰地、平台独立地表示内容的方法。

在众多编程语言中，Java 是使用 XML 的优秀平台，XML 又是 Java 应用的优秀数据表示方法。XML 和 Java 都与 Internet 关系密切。XML 被设计成为一个优化的、灵活的可读格式，可直接用于 Internet；而 Java 从一开始就支持 socket，HTTP，HTML 和服务器。它们都支持 Unicode，因而很容易实现本地化应用。正如 Java 向程序员提供了表达复杂数据结构和面向对象模型的能力一样，用 XML 表达复杂的层次化数据模型是很理想的。近年来 Java 开发者已从丰富的开发环境中受益，而 XML 的支持者更可以得到丰富的帮助处理 XML 文档的工具。尽管有许多工具和库是基于其他语言的，如 Python，Perl 和 C，但 XML 开发的主流仍然集中于 Java。

用户用 XML 和 Java 创建的应用将依赖于所用的 Java XML 解析器提供的服务。信息本身是存储在一些可持续的引擎中。从这些引擎得出的数据必须先转换为 XML，转换完成后就要有应用来处理它们。

1998 年 2 月 10 日 W3C 发布了 XML1.0 标准。从那时起，XML 技术作为一种网络系统中通用的数据交换格式迅速得到了支持。使用 XML 的实际的好处有：

- 结构化——建立有任何复杂层次的数据模型。
- 可扩展性——根据需要定义新的标志。
- 验证——检查数据在结构上的正确性。
- 独立与媒介——以多种方式发布内容。
- 独立于供应商和平台——使用标准的商业软件甚至文本工具处理任何符合（XML 标准）的文档。

1. XML Java 应用的层次

XML Java 应用的层次结构大致可做如下划分：

- Java 应用层。这一层实际上包括所有的 Java 代码，这些代码利用解析器访问 XML 文档中的数据。
- XML 解析器层。
- XML 源层，它提供应用所需的 XML 数据。
- 可持续的引擎，这里是实际存储数据的地方，XML 源层从这里获取数据。

XML Java 应用可运行在服务器端或客户端，或者同时在服务器和客户端上运行。其界面既可能是图形用户界面，也可能是基于 Web 的。我们大致可将其简单地分成三类：

（1）客户端的图形 Java 应用。最简单的 XML Java 应用就是将信息存储到 XML 文档中的 Java 应用了。用 XML 作为信息表示和存储的方式，可以不必使用专有的或二进制的文件格式，这将使用户的应用具有跨平台、跨应用甚至跨编程语言的互操作性。因为 XML 可以定义任意

类型的标记语言，所以应用也可以用自己的标记语言存储信息。有些商业程序已经允许将其应用数据存储为 XML 文件，例如，Framemaker 可将其产生的文档存为 XML 文件。当然，为了创建这类应用，用户必须定义自己的 DTD。

（2）客户和服务器端的应用服务器。应用服务器将许多不同的网络软件集成在一起，以从多个来源向一组客户端 Java 应用或 Web 浏览器提供信息。XML 则是这些来源的共同基础，使得分离的系统能共享一种只包含纯信息（及其结构关系）的介质。由于使用纯文本编码数据，因此信息交换不需要什么特殊的二进制信息格式转换，同时通过 HTTP 协议在分散的网络上传输数据也非常自然。其他远程信息交换如 RMI 和对象序列化则受太多限制。将来，当可公开获得各行业的标准化 DTD 时，基于 XML 的应用服务器将非常流行。同样，当 XML Schema 仓库被普遍使用时，应用服务器将提供许多现在无法提供的服务。需要共享信息的公司只要统一采用一套 DTD 或 Schema 就可以交换信息，不论它们用什么样的系统存储信息。

（3）基于 Web 的应用。基于 Web 的应用类似于应用服务器，不过它没有自己的客户应用程序，而是使用 Web 浏览器作为客户端。它们用自动生成的 HTML 产生前端，在 Java 环境中，Servlet 最适合于这一工作。基于 Web 的应用可以依赖于其他应用服务器收集信息。用户也可以自己写 Servlet 从远程或本地数据库、XML 文档仓库甚至其他 Servlet 获得信息。基于 Web 的应用可用于封装一个应用服务器，它不需要在客户端安装特定的 Java 虚拟机或其他插件，客户可以通过浏览器访问应用服务器提供的服务。

2. 针对 XML 技术的 Java 技术标准扩展

SUN 通过 Java 平台支持 XML 技术，并正领导着为 XML 定义 Java 技术标准扩展的努力。它将通过 Java Community Process 的业界参与者来开发，以确保稳定性和兼容性。企业可以信赖 XML 标准扩展来获得与 Java 平台的高质量的集成。

第一步是通过 XML 标准扩展提供基础功能，包括读、维护和生成 XML 文本。这些核心功能将形成开发全功能的，基于 XML 技术的应用程序的构造块。

XML 标准扩展将由一个规范，一个参考实现和一个兼容性测试工具组成。根据 SUN 关于对开放过程和工业标准承诺，XML 标准扩展将顺从 XML 1.0 规范，并充分利用已经为 XML 技术开发的 Java API，包括 W3C DOM Level 1 核心建议和 SAX 1.0 API。

根据波士顿 Patricia Seybold Group 的资深顾问 Anne Thomas 的介绍，这个标准扩展是向前迈出的一大步：“针对 XML 的 Java 平台标准扩展将提供生成和处理 XML 的标准类，并且，因为是标准扩展，这些类将在几乎所有的 Java 平台上提供。开发者不再需要自己开发这些类，并且 XML 文档不会显得很累赘，因为我们不需要在应用程序的代码中包含这些类。这些类将会驻留在目标系统中。”

3. 企业平台支持

在 Sun，这一新技术的最大支持者也许是 Jon Bosak，他还是 W3C XML 协调组的主席，通常被认为是 XML 之父。Bosak 说：“XML 和 Java 是厂商独立程序的阴和阳。把它们集成在一起，用户能获得完整的、平台独立的、基于 Web 的计算环境。”

XML 技术还会被使用在 SUN Java 企业平台的一些关键领域。Java 2 平台企业版产品线经理 Bill Roth 指出：“XML 是我们下一代企业计算平台：Java 2 平台企业版计划的基础。我们

将通过它来使 Enterprise JavaBeans 组件更便于使用。我们还将使它成为传送企业关键任务数据的标准。”

Sun 已经宣布它正在将基于 XML 技术的标准扩展加入下一个版本的 Enterprise JavaBeans 架构，以响应客户对提高 EJB 组件的适用性的要求（这里所说的是 EJB 2.0，它已经发布了）。

XML 技术被期望给面向网络的应用带来革命性的影响，特别是在数据交换领域。Java 与 XML 一起使得在诸如电子商务和企业应用集成这样领域的新一代 Web 应用成为可能。

目前，几乎所有 Internet 技术的主要参与者都承诺支持 XML 技术。除了 Sun 以外，像 IBM，Oracle, Fujitsu, Novell, Webmethods, Ariba, Bluestone, CommerceOne, Vervet, NetPost 等公司正在开发将 XML 和 Java 一起使用的产品和技术。

4. “聪明的数据”

Patricia Seybold Group 的 Anne Thomas 解释说：“把 Java 和 XML 技术组合在一起产生了轻便的‘聪明’的数据。XML 提供了普遍适用的格式化的数据格式，同时 Java 技术提供了普遍适用的代码。因为用 Java 语言写的代码可以嵌入用 XML 语言写的文档中，我们可以创建包含自己的数据处理程序的数据结构。这是伟大的组合。”

Java 平台确实是使用 XML 语言工作的开发人员的首选技术。例如，有很多解析器和通用工具是在 Java 平台上开发的。开发人员不仅发现 Java 语言的移植性和吸引人的面向对象特性，他们还被 Java 语言的效率所深深吸引。企业应用集成分析和顾问公司 NC.Focus 的总裁 JP Morgenthal 指出：“使用 Java 语言写他们的工具允许公司和开发人员更快地完成工作。同时，Java 提供字符串处理，对哈希表，URL 的支持，以及其它一些特性使它成为使用开发向 XML 这样的应用的自然工具。最后，共享代码确实容易，这是在这个快速发展的领域中非常重要的一个特性。”

这是一条双向路。利用它的元数据的灵活性和数据移植性，XML 给了 Java 巨大的帮助，使数据通过网络更加容易移植。Java 技术为开发人员提供了相对 C 和 C++的坚实的生产率提高。同时，XML 和 Java 技术直接导致了平台独立的和基于标准的应用程序能被立即开发。

当具有在网络系统上交换信息的需要时，例如电子数据交换（EDI）、电子商务、企业资源计划和工作流应用，XML 和 Java 技术一起成为一种最适宜的选择。

5. 可移植的采购定单

很多观察者相信，XML 和 Java 技术一起将革新我们交换和处理信息的方式，我们将能在收到信息的同时使用建立在 Java 技术上的应用程序，根据我们自己的需要处理它。Sun 的 Bill Smith 解释说：“XML 技术使信息交换成为可能，而 Java 技术使自动处理更灵活。”Bill 是 WWW 协会 XML 连接工作小组的设计师。

例如，用 XML 语言描述的公司采购定单可以包含生动的成分，例如零件和客户编号，它们可以和数据库结合在一起，在不同的程序中自动更新仓库库存和出货记录而不需要重复输入数据。

在这个例子里，一份定单在不同的应用中可以有不同的含义。在采购部的人可能有权利赋予定单号，指定客户代码和修改金额，而供货方将只能证实它和修改金额，收货人只能查看、存储或打印这份文件。但是，在上述每一种情况下，实质上是同一份文档，基于同样的数据，

根据不同的接收者，有不同的行为说明。

或者，同样数据的行为根据处理它的应用程序，甚至应用程序运行的设备的不同而改变。这意味着，举例来说，一个简单的股票市场的数据流可以运行在不同的应用程序中，可以是一个滚动的文本窗口，客户定制的图表或文字和图形混合的 Web 页面。

在文档管理和出版应用中，XML 和 Java 技术可以提供某种突破，比如独立于媒体的出版，独立于设备的表示，客户端处理定制的数据和视图。

这是因为，与 HTML 文档依赖 Web 服务器端的 CGI 描述语言提供功能不同，XML 与 Java 技术可以将更多的应用功能直接提供给客户设备来处理。这提高了用户在客户端对数据的掌握程度，同时又减少了网络处理和流量。

6. 相关文档和规范:

XML Technology Pages onJava.sun.com(http://java.sun.com/xml/)

Java Community Process pages on theJava Developer Connection(http://java.sun.com/jdc/jcp/index.html)

XML andJava Technologies in the News - Search results from ourJava Industry ConnectionSM site. (http://java.sun.com/industry/)

The SGML/XML Web Page by Robin Cover(http://www.oasis-open.org/cover/sgml-xml.html)

Java Project X Technology Release 1 - code for XML technology services.(http://java.sun.com/features/1999/03/xml-side1.html)

Managing Names and Ontologies: An XML Registry and Repository by Robin Cover(http://www.sun.com/981201/xml/))

WDVL.com: The Web Developer's Virtual Library - XML Subsite(http://www.wdvl.com/Authoring/Languages/XML/)

Tutorials for using theJava 2® platform and XML technology(http://developerlife.com)

General XML info: Published by Seybold (http://www.xml.com)

XML FAQ(http://www.ucc.ie/xml/)

17.2 用 Java 创建 XML 文档

17.2.1 关于解析器

XML 应用一般都是围绕解析器建立的。解析器实际上就是一段代码，它读入一个文档并分析其结构。目前已经有多种语言的 XML 解析器和库，包括 Java、C++、Perl 和 Python。由于 Java 与 XML 有着完美的结合关系，因此这里主要介绍基于 Java 的解析器。目前主要的 Java 解析器有 IBM 公司的 XML4J 和 Sun 公司的 Project X，它们都遵循 SAX 1.0 和 DOM 1.0 标准，因此用法几乎是一样的。

1999 年 11 月 9 日，Apache Software Foundation 宣布创建 xml.apache.org 以提供开放源码的 XML 解决方案。IBM 随即宣布将它的 XML4J、XML4C 和 LotusXSL 技术贡献给 xml.apache.org。XML4J 解析器被重新命名为 Xerces，而 LotusXSL 被重新命名为 Xalan。

IBM 正将其 XML 解析技术开发的资源投入 Xerces 解析器，将使用 Xerces 作为 XML4J 和 XML4C 的基础。最新一版 XML4J 3.0.x 就是基于 Apache Xerces codebase 的。它已支持 XML Schema、DOM Level 2 和 SAX version 2。

通常，使用 Parser 需要遵循“三步曲”：

（1）首先创建一个 Parser 对象。

（2）将你的 XML 文档传递给 Parser。

（3）最后对结果进行处理。

解析器可以按不同标准进行分类：

（1）校验的（Validating）和不校验的（non-validating）解析器。我们知道，使用一个 DTD 并遵循 DTD 定义规则的 XML 文档称为有效的文档。遵循基本标记规则的 XML 文档称为格式良好的文档。XML 规范要求解析器一旦发现文档不是格式良好的就立刻报告错误。而不校验的解析器忽略所有的校验错误，即不检查 DTD 中的规则。如此一来，其处理速度和效率就比校验的解析器高得多。因此如果用户确信 XML 文档是有效的，就可以使用不校验的解析器。或者如果只关心找到文档中的标记，那么也可以选用不校验的解析器。

（2）支持 DOM 的解析器。DOM 是 W3C 的官标方准，它定义了一个接口，使程序能访问和更新 XML 文档的样式、结构和内容。支持 DOM 的解析器实现这一接口。使用一个 DOM 解析器，将得到一个包含文档中所有元素的树结构。

（3）支持 SAX 的解析器。SAX API（应用程序接口）是处理 XML 内容的另一种方式，由 XML-DEV 邮件列表的成员开发，它已经是事实上的工业标准。使用 SAX 解析器解析 XML 文档，解析器将在文档中的不同点产生事件。这些点包括文档的起点和终点、在元素中发现字符时以及其他一些点，然后用户可以决定如何处理这些事件以及从解析器得到的信息。

通常在以下情况下应该使用 DOM：需要详细了解文档的结构、需要移动文档的组成部分（如排序）、需要不止一次地使用文档中的信息。

而在下列情况下，应该使用 SAX：只需要从一个 XML 文档中抽取一些元素、没有很多可用的内存或者文档中的信息只用一次。

17.2.2 JDOM 概述

由于 XML 文档是结构化的，因此如果使用 XML 外部处理器，也可正确地取出所需要的数据。但在使用 XML 文档全部应用中，用来组成 XML 处理器的难点在于代价过高。因此对从应用程序调用 XML 处理器的接口做了规定。这种 API 应用程序接口称为 DOM 文档对象模型方法，在用 DOM 方法确定 API 的标记中，可以采用对象管理组 OMG 规定的 CORBA（公用对象需求代理体系）所确定的 IDL 接口定义语言。

DOM 的基本概念是：根据 XML 文档中对于使用的“某某名称”标记所显示的值，即可决定从应用程序中进行调用的规则。在 EC 电子商务领域中，为了实现在企业间自动交换的 Web 自动化，DOM 技术的应用是不可缺少的。

当前 XML 的 API 很多。我们应该把 SAX，DOM 以及 JAXP 放在相同的高度去看待。对 XML 来说，似乎所有必需的工具都已经被其包括在其中了，而 Java 却用一种非标准的方式给我们留下了一些问题：SAX 和 DOM 如何工作，XML 文档中出现的某些问题，如何获得一个解释器。因而，依据开放源代码软件的惯例，我们决定修补这些问题，让用户得到最终的解决

方案：一种新的 API——JDOM！

JDOM 寻求一种以 Java 为中心的，多数情况下能对 SAX 和 DOM 进行比较选择的方案。这种方案不应基于 DOM 或者 SAX，并且它允许用户在不利用 DOM 的特性的条件下处理 XML 文档。同时，它应该具有 SAX 的高效，允许快速解析和输出。对扩充实现的完整支持也必须包含其中。JDOM 本身有具体的（不是抽象的）类构成。因此，创建元素、属性、注释和其它的 JDOM 要素并不需要代理的支持。

像 DOM 和 SAX 这样的标准 API 在 WWW 上和 XML 中已经有了广泛的应用，因此修订它们需要花费大量的时间。但是，现在的软件正往开放源代码和公用接口（以及易维护性）的方向发展，这却呼唤版本的快速升级和 Java.xml 说明书的快速更新。正走向开放源代码模式的 JDOM 就致力于这个方向。比如 JDOM 在所有的文件对象中（即使这份文档是由其它的解析器创建的）已经加入了对 XML 名称域的支持。

我们已经说过 JDOM 是基于 Java 2 的一种完整的 API。它加强了 Java 聚集类的能力。还没有通用的方案把这个 API 导入到其它语言中，但是如果我们能够降低 API 与其它语言通讯的标准，那么它的易用性将会大大提高——这也是我们的目标。基于 Java 2 的核心 JDOM 类允许用户使用聚集类和弱参照。当然，JDK1.1 版也是通用的。

Java 开发者们希望他们开发的 JDOM 在学习和使用的时候能够容易一些，并且符合已经存在的 Java 设计规范。通过直接对象的解释，JDOM 结构（元素、注释、属性等等）得以创建。我们希望读者能从整体上把握 XML 文档（对其它文档也是一样），并且希望在任何时候这些文档都是有用有效的。处理 XML 的移动，修改问题的最直接的方法已经被提了出来，并且 Java 类已经开始用于输入输出（URL、输入流、输出流等等）。

17.2.3 JDOM 生成 XML 文档

获取一个 JDOM 文档有两种方法：如果读者没有可供读取的 XML 文档，我们可以随机创建一个 JDOM 文件对象；如果资料已经存在，可以重构 JDOM 文件对象（JDOM 的文件对象就是一个代表 XML 文档的核心类）。

限于篇幅，我们不列出所有引用的类和规范（所有带下划线的部分）。读者可以在 http://www.jdom.org 得到这里我们使用的完整相关文档、类库和规范说明。

如果在开始的时候不存在 XML 资料，那么利用构造器创建 JDOM 文件对象是一件非常容易的事情：

```
Document do=new Document(new Element("root"));
```

正如我们曾经讲到的那样，JDOM 是一整套具体的类，而不是什么接口，这就意味着有一些在 DOM 中处于必作之列的事情在 JDOM 反而不必要。我们只需要对文件对象进行一些新的操作。在 JDOM 中，我们有一系列的文件可以使用。

文档并没有与解释器捆绑在一起。XML 通常是由空白模板创建，而不是由已存在的 XML 资料源。因此，对于 org.jdom.Document 这个 JDOM 构造器来说用一个根元素作为参数就足够了。例 1 就是使用 JDOM 构建了一个 XML 文档。

例 17.3

```
import org.jdom.Document;
import org.jdom.Element;
```

```
/**
 * <p>
 * Demonstrate building a JDOM Document from scratch.
 * </p>
 *
 * @version 1.0
 */
public class FromScratch {

    /**
     * <p>
     * Build a simple XML document in memory.
     * </p>
     */
    public static void main(String[] args) {
        Document doc = new Document(new Element("root"));
        System.out.println("Document successfully built");
    }

}
```

上例中，程序员使用了一个新元素作为根（元素采用了 root 这个名字）创建了一个 JDOM 文件对象。这个文档可以用作各种用途。它在内存中处理过之后，将会输出到输出流中。

1. 从 XML 构建文档

无数据构建 JDOM 文档是很容易的，但更多的情况却是通过读入数据，构建 JDOM 文档。因为 JDOM 文件可由多种资料源创建。org.jdom.input 包定义了一个构造入口，如下所示：

```
public interface Builder {

    // Create a JDOM Document from an InputStream
    public Document build(InputStream in) throws JDOMException;

    // Create a JDOM Document from a File
    public Document build(File file) throws JDOMException;

    // Create a JDOM Document from a URL
    public Document build(URL url) throws JDOMException;
}
```

程序提供了从各种输入源中创建 JDOM 文件的模块。它能够根据不同的输入格式创建不同的实现。JDOM 提供了两种构建方式：SAXBuilder 和 DOMBuilder。这可以保证标准的解释器能够直接用来构建 JDOM 文件对象，而不需要额外的支持。

在JDOM文档中我们最常做的事就是利用输出流输出XML数据到一个文件或者另一个应用组件之中。这个输出流可能包括控制台输出、文件、URL或者其它可以接受数据的结构。这个任务是在org.jdom.output.XMLOutputter类中处理的。这个类有如下的构造器和输出方法：

```
public class XMLOutputter {
    // Accept defaults: 2 space indent and new line feeds on
    public XMLOutputter(    );
    // Specify indent, accept default for new line feeds (on)
    public XMLOutputter(String indent);
    // Specify the indention to use and if new line feeds should be used
    public XMLOutputter(String indent, boolean newlines);
    // Output a JDOM Document to a stream
    public void output(Document doc, OutputStream out)throws IOException;

}
```

我们用缺省的构造器为例来说明问题。下面的例子用的是SAXTest类，它将文档打印到标准的输出流中：

```
import java.io.File;
import java.io.IOException;

import org.jdom.Document;
import org.jdom.Element;
import org.jdom.JDOMException;
import org.jdom.input.SAXBuilder;
import org.jdom.output.XMLOutputter;

public class SAXTest {
    public static void main(String[] args) {
        if (args.length != 1) {
            System.out.println("Usage: SAXTest [filename to parse]");
            return;
        }

        try {
            // Request document building without validation
            SAXBuilder builder = new SAXBuilder(false);
            Document doc = builder.build(new File(args[0]));
            printDocument(doc);
        } catch (JDOMException e) {
            if (e.getRootCause(    ) != null) {
                e.getRootCause().printStackTrace(    );
            }
            e.printStackTrace(    );
        } catch (Exception e) {
            e.printStackTrace(    );
        }
    }

    public static void printDocument(Document doc) throws IOException {
        XMLOutputter fmt = new XMLOutputter(    );
```

```
            fmt.output(doc, System.out);
        }
    }
```

我们应该注意到在使用 SAX 构造器和 DOM 构造器构建 JDOM 文档对象的方法中，没有使用任何数据，一个构建结束的文档会立即等待输出——这就使读写 XML 变得非常容易！

17.2.4 XML Java API 版本 2.0.x

1. XML Java API 版本 2.0.x 要点

Application Server 版本 3 中的 XML 文档结构服务包括：

- XML Java API 版本 2.0.x。
- XML Java API 版本 1.1.x（包括支持在 Application Server 版本 2 下开发的 XML 应用程序）。

这两个 XML Java API 版本都位于 AS_install_root\lib\xml4j.jar 中。对于 API 版本 2.0.x 所要强调的是：

（1）通过提供下列对象的接口提高扩展的模块化体系结构：

- 可插的验证器
- 可插的 DOM 设备
- 可插的目录支持

（2）在 API 版本 2.0.x 中新增的 4 个语法解析器配置如下：

- 无验证 SAX 语法解析器（com.ibm.xml.parsers.SAXParser）
- 验证 SAX 语法解析器（com.ibm.xml.parsers.ValidatingSAXParser）
- 无验证 DOM 语法解析器（com.ibm.xml.parsers.NonValidatingDOMParser）
- 验证 DOM 语法解析器（com.ibm.xml.parsers.DOMParser）

（3）API 版本 1.1.x 中首先支持的两个语法解析器配置为：

- 无验证 TX 兼容语法解析器（com.ibm.xml.parser.Parser）
- 验证 TX 兼容 SAX 语法解析器（com.ibm.xml.parser.Parser）

（4）在按计划修改 DOM 树后对该树进行再次验证的能力。

(5）提高的性能。

（6）对基于 DOM 和 SAX 应用程序的大型文档进行语法分析的能力。

（7）对 XCatalog（API 版本 2.0.x 中新增的 XML 格式目录模块）和 TXCatalog（API 版本 1.1.*中便已提供）的支持。

2. 移植到 XML Java API Version 2.0.x

如果用户使用 API 版本 1.1.x 开发 XML 应用程序，则可以在某种限制下在 Application Server 版本 3 中使用那些应用程序，或者可以将那些应用程序移植到 API 版本 2.0.x 中。对于移植要考虑的是：

API 版本 2.0.x APIs 中存在一些固有的性能改进，但用户可以在数据可托管的应用程序环境中明确地使用无验证语法解析器（替代 API 版本 1.1.x 中的强制验证），以获取其它方面的性能改进。

大多数的 XML 应用程序在一定程度上基于 DOM 或 SAX API。移植到 API 版本 2.0.x 的主要决定因素是应用程序是否必须严格地支持 DOM 或 SAX API。如果要求的话，就需使用 API 版本 2.0.x。如果应用程序使用了一些兼容 API 的方便特性，则可使用 API 版本 1.1.x。

注意　用户不能将版本 2.0.x API 和版本 1.1.x API 混用来创建 XML 应用程序。

3. 更改为 API 版本 1.1.x

Application Server 版本 3 XML4J.JAR 文件中包括下列 TX 兼容的 API 版本 1.1.x 软件包：

- com.ibm.xml.parser 软件包
- com.ibm.xml.xpointer 软件包

如果用户需要 API 版本 1.1.x 中提供而在 API 版本 2.0.x 中未提供的语法解析器功能，则可以使用 TX 兼容的版本 1.1.x 类。使用 API 版本 1.1.x 方法来创建语法解析器，导致该语法解析器读取输入并设置选项。由 TX 兼容类返回的 DOM 树是 API 版本 1.1.x 中的 TX 类实例。

下表概述了 API 版本 1.1.x 类 com.ibm.xml.parser.Parser 的方法（在 API 版本 2.0.x 中不支持或尚未实现）：

方法	状态
Parser.addNoRequiredAttributeHandler	不支持，报告 java.lang.IllegalArgumentException
Parser.getReaderBufferSize	不支持，报告 java.lang.IllegalArgumentException
Parser.setErrorNoByteMark	不支持，报告 java.lang.IllegalArgumentException
Parser.setReaderBufferSize	不支持，报告 java.lang.IllegalArgumentException
Parser.setProcessExternalDTD	未实现，与 API 版本 1.1.x 中的功能不同
Parser.setWarningNoDoctypeDecl	未实现，与 API 版本 1.1.x 中的功能不同
Parser.setWarningNoXMLDecl	未实现，与 API 版本 1.1.x 中的功能不同
Parser.stop	未实现，与 API 版本 1.1.x 中的功能不同

在 API 版本 2.0.x 中，不赞成使用某些版本 1.1.x 的方法。下表概述了一些不赞成使用的方法：

不赞成使用的方法	建议
com.ibm.xml.parser.Parent.addElement(Child)	使用 appendChild()。
com.ibm.xml.parser.EntityDecl.getName()	使用 getNodeName()。
com.ibm.xml.parser.TXNotation.getName()	使用 getNodeName()。
com.ibm.xml.parser.TXElement.getName()	使用 getNodeName() 或 getTagName()。
com.ibm.xml.parser.EntityDecl.getNDATAType()	使用 getNotationName()。
com.ibm.xml.parser.Namespace.getUniversalName()	请参阅 createExpandedName()。
com.ibm.xml.parser.TXElement.getUniversalName()	使用 createExpandedName()。
com.ibm.xml.parser.TXAttribute.getUniversalName()	使用 createExpandedName()。
com.ibm.xml.parser.TXElement.isEmpty()	请参阅 hasChildNodes()。
com.ibm.xml.parser.EntityDecl.isNDATAType()	在以后的发行版中将除去此方法。
com.ibm.xml.parser.TXAttribute.setAttribute(TXAttribute)	使用 setAttributeNode()。
com.ibm.xml.parser.TXNotation.setName(String)	在以后的发行版中将除去此方法。
com.ibm.xml.parser.DTD.setName(String)	在以后的发行版中将除去此方法。
com.ibm.xml.parser.TXText.splice(Element, int, int)	在以后的发行版中将除去此方法。

4. 版本 2.0.x 语法解析器的新方法

下面概述了可在所有经由基本类 XMLParser 的六个 API 版本 2.0.x 语法解析器中使用的新方法：

- setAllowJavaEncodingName 如果设置为真，则允许使用和由 XML 标准定义的编码方案的 Java 语言名称。
- setWarningOnDuplicateAttDef 如果设置为真，则出现重复属性定义的警告。
- setCheckNamespace 如果设置为真，则检查名称空间（namespace）的语法。
- setContinueAfterFatalError 如果设置为真，则在严重出错之后继续处理。
- setDocumentTypeHandler 设置 XMLDocumentHandler。
- setEntityHandler 设置 EntityHandler。
- setValidationHandler 设置 ValidationHandler。
- setDocumentHandler 设置 SAX DocumentHandler。
- setLocale 设置消息中所用的语言环境（即语言）。
- setEntityResolver 设置 SAX EntityResolver。
- setDTDHandler 设置 SAX DTDHandler。
- setErrorHandler 设置 SAX ErrorHandler。

5. 创建版本 2.0.x 语法解析器

API 版本 2.0.x 提供了六个语法解析器配置：

（1）API 版本 2.0.x 中新增的四个语法解析器配置如下：

- 无验证 SAX 语法解析器（com.ibm.xml.parsers.SAXParser）
- 验证 SAX 语法解析器（com.ibm.xml.parsers.ValidatingSAXParser）
- 无验证 DOM 语法解析器（com.ibm.xml.parsers.NonValidatingDOMParser）
- 验证 DOM 语法解析器（com.ibm.xml.parsers.DOMParser）

（2）API 版本 1.1.x 中首先提供的两个语法解析器配置为：

- 无验证 TX 兼容的语法解析器（com.ibm.xml.parser.Parser）
- 验证 TX 兼容 SAX 语法解析器（com.ibm.xml.parser.Parser）

请参阅更改为 API 版本 1.1.x 以获得关于创建版本 1.1.x 语法解析器的实例的信息。

6. 传送语法解析器名称

当 XML 应用程序必须支持在不同的语法解析器中的转换时，这个方法是十分有用的。使用此方法创建语法解析器实例的步骤是：

（1）导入语法解析器、SAX 语法解析器和 SAX ParserFactory 的软件包。

（2）创建包含此语法解析器类的全限定名称的字符串。

（3）要使此语法解析器实例化，将该字符串发送到语法解析器工厂方法。

使用此方法创建一个验证 DOM 语法解析器的示例是：

```
import org.xml.sax.Parser;
import org.xml.sax.helpers.ParserFactory;
import com.ibm.xml.parsers.DOMParser;
```

```
import org.w3c.dom.Document;
...
String parserClass = "com.ibm.xml.parsers.DOMParser";
        String xmlFile = "input_file_URL";
Parser parser = ParserFactory.makeParser(parserClass);
                try {
      parser.parse(xmlFile);
        } catch (SAXException se) {
            se.printStackTrace();
        } catch (IOException ioe) {
            ioe.printStackTrace();
}
// 下一行仅用于 DOM 语法解析器
Document doc = ((DOMParser) parser).getDocument();
...
```

7. 明确地将语法解析器类实例化

如果 XML 应用程序将使用一个特定的语法解析器，则使用此方法创建该语法解析器：

- 导入该语法解析器软件包。
- 实例化语法解析器。

使用此方法创建一个验证 DOM 语法解析器的示例是：

```
import com.ibm.xml.parsers.DOMParser;
import org.w3c.com.Document;
...
        String xmlFile = "input_file_URL";
DOMParser parser = new DOMParser();
                try {
            parser.parse(xmlFile);
        } catch (SAXException se) {
            se.printStackTrace();
        } catch (IOException ioe) {
            ioe.printStackTrace();
}
// 下一行仅用于 DOM 语法解析器
        Document doc = parser.getDocument();
...
```

8. 处理出错信息

创建语法解析器实例时，缺省错误处理器不是自动地注册。因此，当该程序遇到一个错误时，它将失败。要注册一个带有语法解析器的错误处理器，需提供实现 org.xml.sax.ErrorHandler 接口的类。这种需求适用于基于 SAX 和 DOM 的语法解析器。

9. 提高处理时间

API 版本 2.0.x 验证 DOM 语法解析器（com.ibm.xml.parsers.DOMParser）和非验证 DOM

语法解析器（com.ibm.xml.parsers.NonValidatingDOMParser）支持扩展 DOM 树节点的两种方法：

（1）全部扩展。在语法分析结束时创建 DOM 树的所有节点。要调用这种类型的节点扩展，需指定该语法解析器的 setNodeExpansion(full)方法。

（2）延期扩展。仅当访问 DOM 树中的节点时才创建它们。例如，getDocument 返回一个仅包含文档节点的 DOM 树。当应用程序访问该文档节点的子节点时，就会创建子文档。当访问该节点的任何一个子节点时，就会创建该节点的直系节点。这样就缩短了对一个 XML 文件进行语法分析和创建DOM树的时间。尽管此方法减少了对XML文件进行语法分析和创建DOM树所需的时间，但却增加了第一次访问节点所需的时间。创建一个节点之后，它即被缓存，以便加快后继访问。

10. 再验证 DOM 树

用户可以在修改 DOM 树之后对它进行再次验证。对于本地 API 版本 2.0.x，该 DOM 实现阻止了无效节点的插入。因此，对于这样的应用程序再验证是无用的。然而，TXRevalidatingDOMParser 是对 1.1.x 应用程序的一个十分有用的增强。

如需再验证 DOM 树：

- 导入 TXRevalidatingDOMParser 或 RevalidatingDOMParser。
- 调用验证语法解析器的验证方法，以 DOM 节点的名称发送。有效性检查是使用当前文档的 DTD 在此节点根目录下的 DOM 树中的一部分执行的。

在下面的示例中，TXRevalidatingDOMParser 用于读取 XML 文档、对它进行语法分析、通过添加一个无效节点来修改它和再验证已修改的文档。

```
import java.io.IOException;
import com.ibm.xml.parser.TXElement;
import com.ibm.xml.parsers.TXRevalidatingDOMParser;
import org.xml.sax.SAXException;
import org.w3c.dom.Document;
import org.w3c.dom.Node;

public class RevalidateSample {
    public static void main(String args[]) {
        String xmlFile = "input_file_URL";
        TXRevalidatingDOMParser parser = new TXRevalidatingDOMParser();
            try {
            parser.parse(xmlFile);
        } catch (SAXException se) {
            System.out.println("SAX error while parsing: caught "+se.getMessage());
            se.printStackTrace();
        } catch (IOException ioe) {
            System.out.println("I/O Error while parsing: caught "+ioe);
            ioe.printStackTrace();
        }

        Document doc = parser.getDocument();
```

```
            System.out.println("Doing initial validation");
            Node position = parser.validate(doc.getDocumentElement());
            if (position == null) {
                System.out.println("ok.");
            } else {
                System.out.println("Invalid at " + position);
                System.out.println(position.getNodeName());
            }

            // Now insert dirty data
            Node junk = new TXElement("invalid_node");
            Node corruptee = doc.getDocumentElement();
            System.out.println("Corrupting: "+corruptee.getNodeName());
            corruptee.insertBefore(junk,corruptee.getFirstChild().getNextSibling());

            System.out.println("Doing post-corruption validation");
            position = parser.validate(doc.getDocumentElement());
            if (position == null) {
                System.out.println("ok.");
            } else {
                System.out.println("Invalid at " + position);
                System.out.println(position.getNodeName());
            }

        }
    }
```

11. 生成一个 XML 文档

这里提供了利用文档工厂方法或从数据库中抽取数据的方法生成 XML 文档的说明。生成 XML 文档的基本步骤如下：

- 创建文档对象。
- 创建 XML 的说明、根元素、子元素、元素特性和其它文档对象。
- 向根元素和子元素附加对象。

下例使用了 TX 的兼容类来创建一个符合 state DTD 的 state XML 文档。

```
import com.ibm.xml.parser.TXDocument;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Text;
import java.io.PrintWriter;

public class MakeStateDoc
public static void main(String[] argv)
            try
            // Create Document object
```

```
            Document doc = new TXDocument();

            // Make tag as root, and add it
            Element root = doc.createElement("state");
            root.setAttribute("stateid", "MN");

            // Make element and add it
            Element elem = doc.createElement("city");
            elem.setAttribute("cityid", "mn12");
            root.appendChild(elem);

            // Make element and add it
            elem = doc.createElement("name");
            elem.appendChild(doc.createTextNode("Johnson"));
            root.appendChild(elem);

            // Make element and add it
            elem = doc.createElement("population");
            elem.appendChild(doc.createTextNode("5000"));
            root.appendChild(elem);

            // State has city, name, and population
            doc.appendChild(root);

            // Show the XML document
            ((TXDocument )doc).setVersion("1.0");
            ((TXDocument )doc).printWithFormat(new PrintWriter(System.out));

            } catch (Exception e)
            e.printStackTrace();
            }
        }
}
```

MakeStateDoc 类生成一个状态 XML 文档，并在命令提示下显示文档：

```
C:\>java MakeStateDoc
<?xml version="1.0" ?>
<state stateid="MN">
   <city cityid="mn12"/>
   <name>Johnson</name>
   <population>5000</population>
</state>
```

17.3 JSP+XML 平台

17.3.1 JSP 基础

JSP（JavaServer Pages）是由 Sun Microsystems 公司倡导、许多公司参与一起建立的一种动态网页技术标准，其网址为 http://www.javasoft.com/products/jsp。在传统的网页 HTML 文件（*.htm,*.html）中加入 Java 程序片段（Scriptlet）和 JSP 标记（tag），就构成了 JSP 网页（*.jsp）。Web 服务器在遇到访问 JSP 网页的请求时，首先执行其中的程序片段，然后将执行结果以 HTML 格式返回给客户。程序片段可以操作数据库、重新定向网页以及发送 email 等等，这就是建立动态网站所需要的功能。所有程序操作都在服务器端执行，网络上传送给客户端的仅是得到的结果，对客户浏览器的要求最低，可以实现无 Plugin，无 ActiveX，无 Java Applet，甚至无 Frame。

1. JSP 简单示例

JSP 的开发环境设置这里不介绍，需要了解可以查阅相关资料。

JSP 开发工具 JSWDK 中包含的 Web 服务器的文档目录在缺省状态下为 jswdk-1.0.1Webpages，主文档在缺省状态下为 index.html 和 index.jsp。也就是说访问 http://localhost:8080 等于访问 jswdk-1.0.1Webpagesindex.html。

用文本编辑器，如 Windows 中的记事本（Notepad），创建一个文本文件 hello.jsp，保存在 jswdk-1.0.1Webpages 目录下，其内容如下：

```
<html>
<head>
<title>Hello  JSP!</title>
</head>

<body>
<%
String Msg = "This JSP test.";
out.print("Hello World!");
%>
<h2><%=Msg%></h2>
</body>
</html>
```

在浏览器的地址栏中键入 http://localhost:8080/hello.jsp，JSWDK 中的 Web 服务器会执行 JSP 文件中用“〈%”以及“%>”括起来的 Java 程序语句，其中 out.print 是将文字输出到网页，语句“〈%= 变量｜表达式%>”的作用是将 Java Scriptlet 中变量或表达式的值输出到网页。

将变量 Msg 赋值为中文字符串，用“〈%= %>”输出，或者用 out.print 输出中文字符串，则在英文版 NT4 及 Redhat 6.1 下实验运行结果正常，而在中文 NT 4.0 和中文 98 下，则反而会出现乱码。

2. 便于维护的网站界面

JSP 支持服务器端的文件包含，即可以在一个 JSP 文件中插入多个其他文件，用来实现统

一的网站界面。修改上述 hello.jsp 并另存为 mainpage.jsp：

```
<%@ include file="top.htm" %>
<%
String Msg = "This JSP test.";
out.print("Hello World!");
%>

<h2><%=Msg%></h2>
<%@ include file="bottom.htm" %>
```

用可视化 HTML 编辑器，例如 FrontPage、Dreamweaver 等设计网站的框架结构，将设计好的框架结构文件分割成两个部分，上面一半保存为 top.htm，下面一半保存为 bottom.htm，代码如下所示：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>网站主页〈/title>
</head>

<body>
<table border="0" width="100%" cellpadding="4" cellspacing="0" align="center">
<tr>
<td width="100%" colspan="2" bgcolor="#837ED1" align="center"><font face="隶书" color="#FFFF00" size=5>主页标题</font>
</td>
</tr>
<tr>
<td bgcolor="#837ED1" width="15%" valign="top" align="center"><br>
<font color="#FFFFFF">内容</font>
<p><font color="#FFFFFF">内容</font></p>
<p><font color="#FFFFFF">内容</font></p>
<p><font color="#FFFFFF">.....</font></p>
<p>  </p>
</td>
<td width="85%" valign="top">
</td>
</tr>
</table>

</body>
</html>
```

在浏览器的地址栏中键入 http://localhost:8080/mainpage.jsp。

这样网站的界面就能统一起来，而设计者可以集中精力在功能模块上处理用户登录、连接数据库、发送 email 等等。每个 JSP 文件都有如下结构：

```
<%@ include file="top.htm" %>
<%
// 实现某些功能
```

```
%>

<%@ include file="bottom.htm" %>
```

维护网站的界面也相对比较容易，只要修改 top.htm 和 bottom.htm，就能影响到所有网页。

3. 数据库连接

数据库连接对动态网站来说是最为重要的部分，Java 中连接数据库的技术是 JDBC（Java Database Connectivity）。很多数据库系统带有 JDBC 驱动程序，Java 程序就通过 JDBC 驱动程序与数据库相连，执行查询、提取数据等操作。Sun 公司还开发了 JDBC-ODBC bridge，用此技术 Java 程序就可以访问带有 ODBC 驱动程序的数据库，目前大多数数据库系统都带有 ODBC 驱动程序，所以 Java 程序能访问诸如 Oracle，Sybase，MS SQL Server 和 MS Access 等数据库。下面介绍如何用 Access 实现一个动态 FAQ（常见问题及答案）网站。首先建立一个 Access 数据库 faq.mdb，其中的表 faqs 有字段 id（自动增量型，并设为主关键字）、subject（文字型，长度 200）、answers（备注型）。这个表中可以存放一些编程知识的常见问题及答案。

然后，在 Control Panel（控制面板）的 ODBC Datasource 模块中加入 System DSN，取名 faq，并指向 faq.mdb。创建一个 JavaBean，名为 faq.java，并保存在 jswdk-1.0.1WebpagesWeb-INFjsp□eans est 目录下。faq.java 的内容如下：

```
package test;
import java.sql.*;
public class faq
String sDBDriver = "sun.jdbc.odbc.JdbcOdbcDriver";
String sConnStr = "jdbc:odbc:faq";
Connection conn = null;
ResultSet rs = null;

public faq()
try
Class.forName(sDBDriver);
}
catch(java.lang.ClassNotFoundException e)
System.err.println("faq(): " + e.getMessage());
}

}

public ResultSet executeQuery(String sql)
rs = null;
try
conn = DriverManager.getConnection(sConnStr);
Statement stmt = conn.createStatement();
rs = stmt.executeQuery(sql);
}
```

```
catch(SQLException ex)
System.err.println("aq.executeQuery: " + ex.getMessage());
}

return rs;
}

}
```

编译 faq.java 以后，在 jswdk-1.0.1Webpages est 目录下创建 JSP 文件 faq.jsp，其内容如下：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>MY FAQ !</title>
</head>

<body>
<p><b>MY FAQ!</b></p>
<%@ page language="java" import="java.sql.*" %>
<jsp:useBean id="workM" scope="page" class="test.faq" />
<%
ResultSet RS = workM.executeQuery("SELECT * FROM faqs");
String tt;
while (RS.next())
tt = RS.getString("Answer");
out.print("<LI>" + RS.getString("Subject") + "</LI>");
out.print("<pre>" + tt + "</pre>");
}

RS.close();
%>
```

在浏览器的地址栏中键入 http://localhost:8080/test/faq.jsp，faq.jsp 调用 JavaBean，从数据库中读出内容并输出。

17.3.2 JSP 与 XML 协同编程

XML 和 JSP 是这些日子中最热门的事物。本节介绍如何联合这两种技术来为网站的开发做一些基本的准备。还可以同时看一下 DOM，XPath，XSL 和其它 Java-XML 技术的示例代码。

我们在此假设读者已经了解 JavaServer Pages（JSP）和 Extensible Markup Language（XML）。

JSP 的应用很容易，可以用它设计网页，使之看起来似乎和 HTML 一样。唯一的不同是 JSP 是动态执行的。例如，它们可以处理表单 form 和读写数据库。

在本节中，我们可以了解如何使用一种相当先进的方式用 XML 来设计一个系统的一些技术。许多站点有海量数据收集并以一种很标准或很不标准的方式来显示它们。我们打算使用 XML 文件在 Web 服务器上进行存储，并用 JSP 来显示数据。

1. XML 文档

所有人都喜欢照相！他们喜欢展示自己的、亲人的、朋友的、度假时的照片，而 Web 是他们展示的好地方。即使千里之外的亲戚都可以看到。我们将着重于定义一个单独的 Picture 对象。该对象描述了表示一张照片所需要的字段：title，date，一个可选的标题，以及对图片来源的一个指向。

一个图像，需要它自己的一些字段：源文件（GIF/JPEG）的定位，宽度和高度像素（以协助建立<img> 标记。这里可以看到一个很简单的优点，即使用文件系统来代替数据库的时候，用户可以将图形文件存放在与数据文件相同的目录中。

最后，让我们来用一个元素扩展图片记录，该元素定义了一套缩略图来用于内容表或其它地方。这里我们使用了和先前同样定义的图片内容。

一张图片的 XML 表示可以是这样的：

```
<picture>
<title>Alex On The Beach</title>
<date>1999-08-08</date>
<caption>Trying in vain to get a tan</caption>
<image>
<src>alex-beach.jpg</src>
<width>340</width>
<height>200</height>
</image>
<thumbnails>
<image>
<src>alex-beach-sm.jpg</src>
<width>72</width>
<height>72</height>
</image>
<image>
<src>alex-beach-med.jpg</src>
<width>150</width>
<height>99</height>
</image>
</thumbnails>
</picture>
```

注意，通过使用 XML，我们将一张单独图片的全部信息放到了一个单独的文件中，而不是将它分散放入 3~4 个表中。

我们将这称为.pix file，于是文件系统会是这样的：

```
summer99/alex-beach.pix
summer99/alex-beach.jpg
summer99/alex-beach-sm.jpg
summer99/alex-beach-med.jpg
summer99/alex-snorkeling.pix
etc.
```

2. *存放数据方法*

将 XML 数据放到 JSP 中也不止一种办法。这里列举了其中一些方法，（其实，很多其它工具也可以做得同样出色）。

- DOM：可以使用类来调用 DOM 接口对 XML 文件进行分析检查。
- XMLEntryList：可以使用其他人的代码来将 XML 加载到 name-value pairs 的 java.util.List 中。
- XPath：可以使用一个 XPath 处理器（如 Resin）通过路径名在 XML 文件中定位元素。
- XSL：可以使用某种 XSL 处理器将 XML 转换成为 HTML。
- Cocoon：可以使用开放源码的 Cocoon framework。
- 运行自己的 bean：用户可以写一个外壳类，使用某种其它技术来将数据加载到自己定义的 JavaBean 中。

这些技术将和一个从另外来源取得的 XML stream 执行得同样出色，例如一个客户端或者一个应用服务器。

3. 查找文件

当 JSP 启动时，初始化后第一件事情就是查找用户所要的 XML 文件。它是怎么知道在众多文件中用户要找的是哪一个?这来自于一个参数，使用者会在调用 jsp 的 URL 中加入参数：picture.jsp?file=summer99/alex-beach.pix （或者通过 HTML 表单来传递文件参数）。

但是，当 JSP 接受此参数以后，仍然只完成了一半工作，因为还要知道文件系统的根目录在哪里。例如，在 Unix 系统中，实际文件可能在这样的路径：

/home/alex/public_html/pictures/summer99/alex-beach.pix。

JSP 文件在执行状态时没有当前路径概念。所以要为 java.io 包给出一个绝对路径。

Servlet API 可以提供一个方法来将一个 URL 路径，从相对于当前 JSP 或 Servlet 的路径转化成为一个绝对的文件系统路径。方法是：

ServletContext.getRealPath(String)。

每一个 JSP 有一个叫做 application 的 ServletContext 对象。所以代码可以是：

```
String picturefile =
application.getRealPath("/" + request.getParameter("file"));
```

或者是：

```
String picturefile =
getServletContext().getRealPath("/" + request.getParameter("file"));
```

它也可以在 servlet 中工作（必须加上“/”，因为此方法需要传递 request.getPathInfo()的结果）。

这里有一个重要的提示：每当存取本地的资源时，要非常小心地检查输入数据的合法性。黑客或者粗心的用户，可能发送伪造的或错误的数据来破坏用户的站点。例如，请想一下以下的表达会发生什么结果：

如果输入了 file=../../../../etc/passwd。

用户会读到服务器的 password 文件！

4. 缓存

尽管有很多优点，但分析一个 XML 文件总是需要耗费时间。为了改进基于 XML 应用的性能，需要使用某种缓存技术。这种缓存必须在内存中保存 XML 对象，记住它们来自哪一个文件。如果对象被加载以后文件被修改了，那么对象需要重新加载。我们可以开发一个用于这种数据结构的简单方法，供给一个 CachedFS 调用返回功能，使用实际执行 xml 分析的内部类，将文件转为一个对象。Cache 于是可以在内存中存储那个对象。

这里是创建 Cache 的代码，这一对象有 application scope，所以此后的请求可以使用同一对象 cache。把这些代码放到 init.jsp，这样就不必将这些初始化的代码剪贴到其他 JSP 文件中去了。总之，必须在一个公共的地方定义 application-scope 对象。

```
<jsp:useBean id="cache" class="com.purpletech.io.CachedFS" scope="application">
<% cache.setRoot(application.getRealPath("/"));
cache.setLoader( new CachedFS.Loader() {
// load in a single Picture file
public Object process(String path, InputStream in) throws IOException
{
try {
Document doc = DOMUtils.xml4jParse
(new BufferedReader(new InputStreamReader(in)));
Element nodeRoot = doc.getDocumentElement();
nodeRoot.normalize();
Picture picture = new DOMPicture(nodeRoot);
return picture;
}
catch (XMLException e) {
e.printStackTrace();
throw new IOException(e.getMessage());
}
}
});
%>
</jsp:useBean>
```

这部分内容很浅显地讨论了在 JSP 中嵌入 Java 来从 XML node 中展开数据的问题。完成同样工作还可以有另外一种常见的模型：Extensible Stylesheet Language （XSL）。 这一模型和 JSP 模型有着根本的不同。在 JSP 中，主要内容是 HTML，它包含了一些 Java 代码片段；而在 XSL 中，主要内容是 XSL 文档，它包含了一些 HTML 片段。

现在相信读者应该有了一个 JSP-XML 应用及其强大威力的初步结构认识。

其实开发 JSP-XML 应用中最令人烦闷的是为每一个 XML schema 中的元素 element 创建 JavaBean。XML Data Binding 组织正在开发一种技术，可以为每一个给定的 schema 自动生成 Java 类。另外，IBM alphaWorks 最近也推出了 XML Master，或称为 Xmas，这是另一种 XML-Java data binding 系统。另外一种可能性是扩展文件系统的功能，建立一些更加强大的功能，如查询和事务处理。

17.4 小　　结

通过本章的学习，我们可以使用 XML 分析器、Java 对 XML 文档进行处理，这样我们就能够基于 JSP+XML 平台创建 Web 网站并对其功能进行最大程度的扩展。