

## 第1章 XML 简介

本章我们将简要讨论 XML 的历史起源，并理解与 XML 相关的技术的关键部分是如何协同工作的。沿着这条思路，我们还将研究 XML 的基本特征，以及此项技术对过去及未来的 Web 体系结构产生的冲击。我们希望这些能够为大家深入挖掘本书其他部分的精华提供一个坚实的基础。

### 1.1 标记语言

早在印刷出版技术出现时，作者就曾经在草稿上做出各种说明，指导印刷工人如何处理版面排放以及其他制作问题。这些说明被称作标记，而协调一致用来定义整套语法和文法的标记集合则被称作语言。例如，校对者就是用手写的标记语言（markup language，ML）与作者交流文字的正确与否。即使是现代的标点符号的使用也是某种形式的标记，因为它告诉读者如何对一段文本进行断句。大多数 ML 都非常特殊，以便使标记与其涉及的文本区分开来。校对者的标记使用的是草体的手写文字和特殊符号，这就与印刷体的文本有着明显的不同。同样，标点符号本身就非常特殊，不可能与代表文本内容的数字与字母相混淆。由于某些标点符号对于理解和排放印刷文字来说是不可或缺的，所以被包含进了 ASCII 码字符集——用于几乎所有现代计算机的基本字符集。因此，这些符号也成为了现代编程语言语法的一部分，应该说，符号的标准化使其又一次占据了重要的地位，不再是仅仅发挥语言标点的作用。

ASCII 标准中还定义了一组用来标记数据传输结构的符号（“C0 控制字符”，其十六进制值为从 00 到 1F）。这些符号中只有一小部分被广泛接受，而且它们的使用也曾经常常出现矛盾。这其中最常见的就是用来界定文档中一行文本的字符。

电传打字机使用的是基于物理动作的字符对 CR-LF（回车，换行），此后的 MS-DOS 和 MS-Windows 也承袭了这种习惯。与此相反的是，Unix 只用一个 LF 字符，MacOS 只用一个 CR 字符来界定一行文本。由于存在这些冲突和 ASCII 码的非标准使用，这些系统之间的文档交换就必需有一个转换步骤——即便是简单的文本文件如果没有经过这个过程也不能被共享——这仅仅是最简单的标记问题，它甚至没有涉及到如何组成一“行”文本的问题。大多数数字处理程序都淘汰了文本“行”的概念，而代之以“段落”，将行结束标记当作段落尾对待，ASCII 码句号-空格（“.”），句号-空格-空格（“.”）则被用来界定语句（虽然这种方法并不完美）。

有各式各样的分隔符用来定义内容的分界，特殊符号的形状，文本的表现形式，以及其他特殊的文档特征。例如，C 和 C++ 程序语言使用大括号 { ..... } 来界定数据或代码单元，例如函数、数据结构、对象定义，等等。主要用于手工编辑的排版语言则可能使用更易阅读的字符串，例如 .begin 和 .end。其他语言也可能使用其他字符或者字符串——它们也常常被称作标记（tag）。当然，在不同的标记集之间及其解释上经常会出现冲突。如果没有通用的分隔符表，没有通用的内部数据格式，要将数据从一种格式转变为另一种格式就会非常困难，更不用说在不同的应用和组织之间共享数据了。

1969年，人类第一次登上了月球。与此同时，IBM的研究人员 Ed Mosher，Ray Lorie 和 Charles F. Goldfarb 发明了第一种现代标记语言——通用标记语言（Generalized Markup Language，GML）。GML 是一种自参考的语言，它可以用于标记任何数据集合的结构，同时它也是一种元语言（meta-language）——能够描述其他语言及其语法和词汇表的语言。此后，GML 发展成了标准通用标记语言（Standard Generalized Markup Language，SGML）。1986年，SGML 被国际标准化组织（ISO）接受为国际性的数据存储和交换的标准，并收录在 ISO 8879 当中（参见 <http://www.iso.ch>）。考虑到 Web 对人类商业和通信行为产生的巨大影响，我们有理由相信，在技术发展史上，GML 的悄悄问世可能是比人类第一次到另一个天体上冒险更为重要的事件。

“标记”是一种传输元数据（即关于数据集本身的信息）的方法。标记语言使用文字串或“标记”来界定和描述这些数据。

下面是一个非常简单的 SGML 文档的样例：

程序清单 1-1

```
<!DOCTYPE email [  
<!ELEMENT email O O ((to & from & date & subject?), text) >  
<!ELEMENT text - O (para+) >  
<!ELEMENT para O O (#PCDATA) >  
<!ELEMENT (to, from, date, subject) - O (#PCDATA) >  
<date>10/12/99  
<to>you@yours.com  
<from>me@mine.com  
<text>I just mailed to say...
```

SGML 是一种非常强大（当然也相当复杂）的标记语言，它已经被美国政府及其合同商、大型制造公司、信息技术发布者广泛采用。出版商也经常使用 SGML 制作各类纸张文档，如书籍、报告、参考手册等。然后，这些 SGML 文档会被转换为合适的格式，接着交给排版者和印刷者处理。SGML 还被用来将技术规范应用于生产制造。但是，它的复杂性及其实现所需要的大量资金又意味着大多数商业用户和个人用户无法享受此项技术所带来的益处。

你可以在 <http://www.oasis-open.org/cover> 了解到有关 SGML 的更多信息。

不过，随着万维网实现技术的不断进步，必将驱动我们寻找到一种更为简便的途径。

## 1.2 XML 的起源和目的

1996年，万维网协会（或者叫 W3C，<http://www.w3c.org>）开始设计一种可扩展的标记语言，使其能够将 SGML 的灵活性和强大功能与已经被广泛采用的 HTML 结合起来。这种后来变成 XML 的语言继承了 SGML 的规范，而且实际上就是后者的一个子集。从 SGML 入手使得该设计小组能够将精力集中在简化已有的成果上。SGML 已经提供了一种可以无限扩展的语言，它允许任何人能够根据自己的需要加以扩充。XML 之所以要较 SGML 更为简化，很大程度上是出于易用性的考虑：人们对标记的读写过程应该使用现有的、简便通用的工具，同时，我们也应当简化

计算机对文档和数据交换的处理。由于有太多的可选功能，SGML变得过于复杂，以至于很难编写出针对这种语言的普通解释器，而XML的解释器则简单得多。此外，XML使得现有的Internet协议和软件更为协调，从而简化了数据处理和传输。作为一个不错的SGML子集，XML还保持了对现有的面向SGML的系统的向下兼容性，这样，用XML标记过的数据就仍然可以在这些系统中使用，为基于SGML的行业节省了大笔的改造费用，同时，与Web的结合也使得它们更便于被访问。

1998年2月，XML 1.0成为了W3C的推荐标准。包括Extended Backus-Naur Form (EBNF) 中语法标识在内的这个正式的规范可以很容易地从W3C的Web站点（<http://www.w3c.org/TR/REC-xml>）上得到；此外，XML规范的制定者之一Tim Bray还在<http://www.xml.com/axml/testaxml.htm>上提供了一个有着非常不错的注解的版本。

在<http://www.ucc.ie/xml/>上，由Peter Flynn等人代表W3C的XML研究组维护的一个XML 1.0 FAQ还提供了到其他与XML相关的主题的连接。

XML是一种界定文本数据的简便而标准的方法。它曾经被人称作“Web上的ASCII码”。就好像你可以使用自己喜爱的编程语言来创建任何一种数据结构，然后同其他人在其他计算平台上使用的其他语言来共享一样。XML的标记用来说明你所描述的概念，而属性则用来控制它们的结构。所以，你可以定义自己所设计出的语法并同其他人共享。

下面让我们来看一个简单的例子：

#### 程序清单 1-2

```
<?xml version="1.0"?>
<books>
  <book category="reference">
    <author>Nigel Rees</author>
    <title>Sayings of the Century</title>
    <price>8.95</price>
  </book>
  <book category="fiction">
    <author>Evelyn Waugh</author>
    <title>Sword of Honour</title>
    <price>12.99</price>
  </book>
</books>
```

我们不必过多地担心语法的特殊性，下面你会看到这种机制的强大所在——只要简单地添加标记就可以描述它们所封装的信息。

XML的数据描述机制意味着它将成为一种在Internet上共享信息的强大途径，因为：

- 它是开放的；XML能够在不同的用户和程序之间交换数据，而不论其平台如何。
- 它的自描述的特性使其对于B2B和企业内部网解决方案来说是一种有效的选择。
- 无需事先协调，我们就可以在程序之间共享数据。后面我们将看到，XML的机制使我们能够找出一类XML文档的结构。

为了使用XML文档，W3C为XML标准化了一套应用程序编程接口（Application Programming Interface, API），这样我们就可以轻松地编制读写XML的程序，同时，开发者团体

还设计了一套特殊的，免费赠送的，基于事件的替代 API。此外，XML在设计时已准备支持非欧洲语言和进行国际化。同 HTML 4.01 一样，XML 基于的是在 ISO/IEC 10646 字符集标准（等同于现在著名的 Unicode 标准，<http://www.unicode.org>）中定义的通用字符集（Universal Character Set，UCS）。可以说，所有促使 HTML 得以流行的特性都出现在了 XML 当中。

但是，XML 并非 HTML 的直接替代品。在仔细阅读了 XML 推荐标准（WWW 协会的对等标准）中的每一个字之后，我们并没有发现任何与可视化表现形式有关的内容。与注重数据及其表达方式的 HTML 不同，XML 只关心数据本身。

虽然 XML 本身是数据，但 XML 的研究者并没有忘记如何表达的问题。与依靠扩充代码部分的传统数据表示方法不同，XML 样式的表现技术是由数据驱动的。从最简单的到极端复杂的，无所不能。但是，不论采用什么技术，XML 的样式化都是通过另一个被称作样式单的文档来实现的。在其中，设计者会格式化样式和决定何时应用样式的规则。然后，这个样式单又可以用于其他多个文档中，以产生类似的效果。

样式和规则在显示时会应用到 XML 的数据上，它们甚至可以转换为 HTML——或者其他什么数据格式。本书的一条主线就是研究如何可视化地表示 XML 标记出的数据。在 HTML 中实现样式的特性可以说易如反掌，但老实说 XML 在这方面并不擅长。

最后，不管 Web 上把 XML 吹得多么神乎其神，但它并不能把之前的难题一扫而光。它能够使编程者轻松自如地完成许多有趣的事情，但 XML 并不是编程语言，也不是基于对象的平台，更不是操作系统。它只是一种能够思考、交换和表示数据的，独立于平台的，强大而精巧的技术。

现在，我们已经对 XML 略知一二，下面让我们来看一看它是如何应用到 Web 的体系结构当中的。

### 1.3 Web 体系结构：过去与未来

第一代 Web 应用程序继承了传统的客户机-服务器模式的软件体系结构。其中某些将关系型数据库作为第三层使用，但所有各层由专用的、固定的方法进行控制。颇具讽刺意义的是，这与 Web 的精髓是背道而驰的——由简单的协议控制的松散的、开放的资源集合。与此相反，XML 则出色地实现了这一目标。它为编程者提供的工具能够构建真正的、由开放标准和自描述数据控制的多层分布式系统。让我们比较一下传统 Web 体系结构与 XML 驱动下的 Web 体系结构。

#### 1.3.1 传统 Web 体系结构

首先，让我们了解一下 Web 应用程序的体系结构（参见图 1-1）。客户端程序是一个浏览器，它充当着浏览者的代理的角色。浏览器将对页面的请求发送给 HTTP 服务器。这个请求会跟随着一系列的参数名称和值。这些可能是被添加在页面 URL 的后面（HTTP GET），也可能是单独发送（HTTP POST）（要了解关于 HTTP 的详细内容，请访问 <http://www.w3.org/Protocols/>）。参数及其名称是由应用程序决定的，而且必须为客户端所知晓，要做到这一点，就需要把它们放置到发出请求的页面中。随后，服务器应用程序的开发者也必须创建客户端页面。任何希望利用服务器的人员都必须使用客户端页面，或者在客户端页面颠倒设计请求的结构。但是，只要服

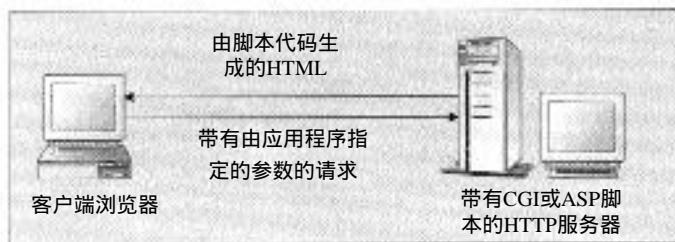


图 1-1

务器改变请求的结构，这类颠倒设计就不再起作用。

在这种前提下，服务器会通过 CGI脚本或ASP代码来动态生成 HTML以满足这类请求。这类脚本可能使用一套数据库，或者进行自身的一些处理。很少（如果有的话）有多个 HTTP服务器协作完成一个请求。

当然，这种结构足以正常的工作，但它确实有一些重要的限制：

- 我们被限制在了客户端浏览器上。
- 没有人建立服务器页面与可编程的代理或其他应用程序通信。请求的结构是固定的。你或者去协调整个服务器应用程序开发组的工作，或者是颠倒设计结构。结果是，如果不是服务器开发组的成员，就很难编制客户端应用程序。
- 所有的内容都以 HTML的形式传递。这就限制了客户端进行任何后期处理的能力，而且在传递时，用户所能看到的也仅限于服务器应用程序决定的内容。
- 如果你希望看到不同的显示方式，比如用图形替代表格，或者另一种排序方式，就必须再到服务器上打一个来回。
- 应用程序非常脆弱；客户端和服务端必须紧密同步。请求只要有一点点变形就会中断服务器应用程序。

那么，XML如何帮助我们解决这些问题呢？

### 1.3.2 XML下的Web体系结构

现在，我们将目光放得长远一些，看一看 XML在未来会给我们带来什么。客户端——浏览器或者程序——将一个XML文档作为请求发送给服务器。同传统的 Web应用程序一样，它包含了指定参数。但是，与传统的客户端不同，请求的结构会根据在运行时从服务器上得到的标准化机制来正式指定。这一机制会保证服务器所期望的结构，并允许客户端在传输前验证请求的正确性。服务器也能够在接收时执行有效性验证。

一旦请求到达，服务器可能继续传统 Web应用程序的处理流程，也可能做出某些改动。XML并不会专注于服务器与服务器之间的通信，但 XML研究者中的改革派使用XML来格式化这类通信。既然在这类交换中服务器收到的数据也是 XML，那么第一个服务器将几个文档合并起来，或者将一个文档转化成另一种格式以满足请求就非常简单了。客户端用来得到结构内容的方法也可以在服务器上使用。由于 XML天生就是层次结构的，所以它很容易就可以对非关系型的数据源进行编码。但是，大多数服务器上维护的数据本来就是关系型的，因此 XML的研究者



花费了大量的时间思考和实践如何使关系型数据与 XML 结构相匹配。所有这些都使 XML 成为在服务器与服务器应用程序之间交换数据的最佳媒介。一旦 XML 被选中成为某个组织交换数据的机制，编程人员可以很快地得到或编制组件和实用程序库以便操作数据。这些对于处理来自客户机应用程序的请求也同样有用（参见图 1-2）。

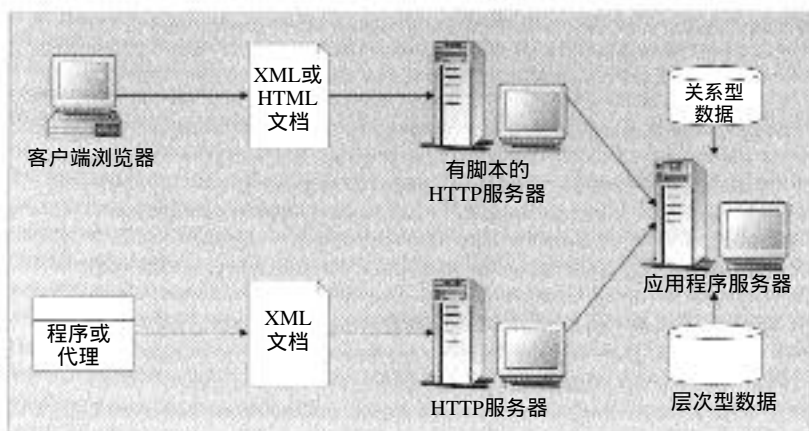


图 1-2

那么这些对我们来说有什么益处呢？首先，我们不再局限于基于浏览器的客户端。XML 本身就是数据，而且可以由程序任意地控制。同样的数据，即可以设定其样式化以便在浏览器中显示，也可以交给一个代理进行后台处理。在这个机制中，XML 文档无需假设数据的最终用途。如果得知客户端需要 HTML，由数据驱动转换过程就会使用 XML 文档生成 HTML 页面。而生成 XML 的底层应用程序不需要任何修改。

在这种机制下，服务器端的应用程序与客户端的耦合程度要松散得多，因为程序具备了找出 XML 文档的结构的能力。这样一来，富有创新意义的应用程序就可以根据程序的要求编写结构新颖的各类文档，应用程序也无需为每一种新的文档类型编制定制的软件。更为典型的应用是创建工业标准标记集，应用程序也可以利用结构自动感知机制来避免因版本更迭带来的矛盾。

在不久的将来，网络中的服务器，客户机和应用程序所进行的处理都将使用这种机制交换数据。幸运的是，这种机制扩展起来并不困难，而且能够在运行时自动找出数据的结构。事实上，任何一种平台都支持这种机制，它使用简单，能够处理来自不同数据源的标记数据。应用程序的开发者可以使用来自非传统数据源或其他服务器的数据来满足客户端的请求。自此，Web 开发已经从客户机-服务器计算体系迈向真正的多层模式。

毫不夸张地说，这一目标离实现已为期不远。

## 1.4 XML 基本特征

现在，让我们了解一下一旦采用 XML，我们能够完成哪些工作，仔细观察 XML 如何使我们实现自己的目标。在前面的章节中，我们曾经大概了解到，XML 是一种使用标记标记内容以传输信息的简单方法。标记用于界定内容，而 XML 的语法允许我们自行定义任意复杂度的结构。这一切都是使用普通的文本，而不是二进制的格式来实现的，这对于跨平台交换数据来说

确实是一个非常不错的方案。实际上，每一种普通的操作系统（只要不是嵌入使用的）能够以某种方式处理文本。这也是 HTML 之所以能够在很短的时间内流行起来的原因。XML 把这一优势提高到一个更新的层次，因为我们可以根据自己的需要任意地扩展 XML。由于这种扩展机制是标准化的，所以我们可以自动地将这类扩展信息传递给任何读取我们数据的人——编程者或机器。下面，我们需要了解 XML 的几个重要方面的内容，只有这样，我们才能够充分理解，并有效地将它们应用到我们的程序中。

### 自描述数据

界定 XML 内容的标记会给所界定的数据中的每一个元素命名。在标记中，我们还会发现特定的属性，它们会为所描述的元素提供某些附加信息。例如：

程序清单 1-3

```
<car>
  <tyre_pressures>
    <front_pressure units="psi">28</front_pressure>
    <back_pressure units="psi">32</back_pressure>
  </tyre_pressures>
</car>
```

之所以说数据是自描述的，在于其中的每个项目都有自己的名字，而这个名字又与文档所描述的现实世界中的问题所涉及的某项具体内容密切相关。到目前为止，我们所了解的内容都还与 HTML 非常相似。虽然 HTML 标记的含义也是由 W3C 制订的。但是如果你希望描述某种不为 HTML 推荐标准所涉及的事务，很快就会发现黔驴技穷。考虑一下 XML，看一看同样的内容，HTML 和 XML 的表达方法有何不同：

程序清单 1-4

```
<Person>
  <Name>
    <First>Thomas</First>
    <Last>Atkins</Last>
  </Name>
  <Age>30</Age>
</Person>
```

程序清单 1-5

```
<Person>
  <Name>
    <First>Thomas</First>
    <Last>Atkins</Last>
  </Name>
  <Age>30</Age>
</Person>
```

两个表单所描述的都是一个人的名字和年龄。在第一个用 XML 实现的表单中，我们可以将

其中的每一点内容与现实世界中的人的概念关联起来。我们界定出姓名的各个部分，我们知道哪个是姓，哪个是名。只要了解到关于人的某些信息，数据的含义对于我们来说就非常清楚了。相反，HTML表单将数据格式化为一个表格，但是没有一个明确的方法能够说明我们处理的内容是关于一个人的信息。其中的一点组织信息——年龄——也被作为内容而不是结构被隐藏起来了。当我们阅读到这段内容时会正确地理解，但计算机是不会的，不管你告诉它多少有关描述的规则也是如此。

XML中的“扩展”一词指的是定义新的标记及其用途的标准机制。由于这一切均是标准化的，所以我们拥有固定不变的途径来描述这些新标记并同其他XML用户交流。对于我们在标记中使用的属性来说，情况也是如此。最近大多数定义用于文档的XML标记集的计划建议都是针对元数据的——关于数据的数据——它们用来在特定的XML文档族中交流标记结构的信息。不仅我们的数据是自解释的，那些关于数据的数据同样也是自解释的。

现在让我们研究一下XML标记在特定领域中的使用。

## 1.5 词汇表

我们曾经在前面暗示过，XML最具特点的特性是它天生的扩展性。与其相比，HTML开始时只是一种简单的标记语言（带有固定的标记集），用于在Internet上交流科技论文，但随着浏览器的开发者不断添加新的标记和功能，这项技术迅速发展起来。在HTML中增加的许多内容都是多媒体传输功能和浮华的商业化Web页面。遗憾的是，这些标记都是各自公司半专用化的，如果用在其他浏览器中经常会出现问题。在这当中，某些部分成为了HTML的正式内容，但大多数都是专用的。可惜的是，这些并没有在数据建模、语义标记或者结构化信息交换协议上为HTML提供多少帮助。

与此相比，XML则一向致力于简便而快速地，根据企业、科学规范或者其他方面的需要来构造定制的标记集。同时，每一个企业（甚至每一个人）都可以选择定义自己的XML标记集，XML的一个好处就在于能够共享这类“词汇表”，它们都使用同样的基本语法、分析程序以及其他工具。可共享的XML词汇表不仅提供了更易于查询的文档和数据库，而且为在不同的组织和计算机应用程序之间交换信息提供了一条途径。

XML“词汇表”是对XML数据的描述，作为信息交换的媒介，它经常是与人类在某种领域（例如商业、化学、法律、音乐）的活动息息相关的。

你可以在<http://www.oasis-open.org/cover/>找到Robin Cover的文章“The SGML/XML Web Page”，这是一篇非常出色的、关于XML词汇表开发的参考文献。

下面，我们就很快地浏览一下目前比较重要的XML词汇表，但并不深入到其语法细节。

### 1.5.1 科学词汇表

第一个应用XML的是Peter Murray-Rust的“JUMBO”浏览器，主要用于化学标记语言（Chemical Markup Language，CML）（参见<http://www.xml-cml.org>）。CML曾经被称作“分子HTML”，但CML还可以进行不同种类的文件格式的转化（不造成任何语义上的改变），并能够



创建适合于专业出版的结构化文档（参见图 1-3）。

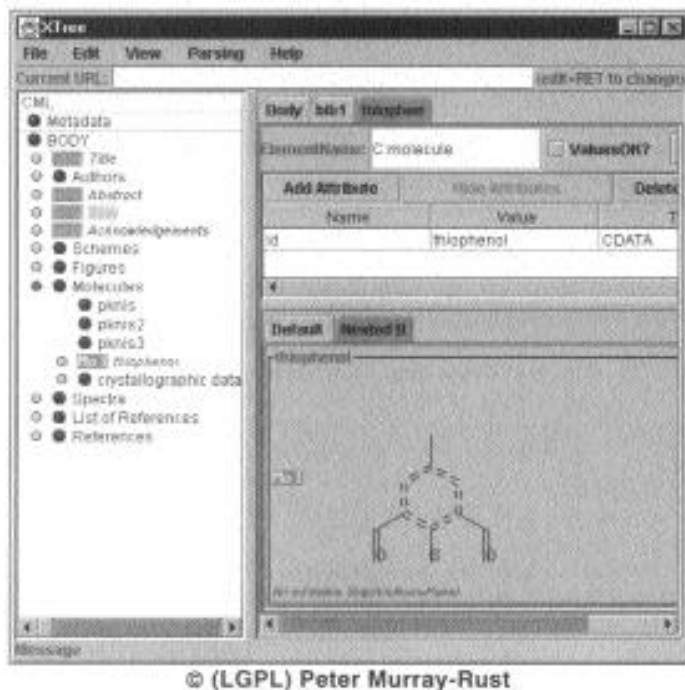


图 1-3

科学的基础语言是数学，XML词汇表中也有有一种MathML，它为数学表达式的转化提供了一条途径。MathML会用纯粹的图片以及（或者）粗糙的 ASCII码来代替各类方程式，以便在适当的浏览器中准确地显示出来，并为代数、几何、统计及其他数学软件工具的符号提供交换格式。（要了解MathML的有关信息，可以访问<http://www.w3.org/Math/>。）

其他科学类的词汇表还包括生命信息序列标记语言（Bioinformatic Sequence Markup Language，BSML），用于基因序列和映射所产生的大量信息（参见<http://www.visualgenomics.com/bsml/index.html>）；NASA用于控制实验室仪器的仪器标记语言（Instrument Markup Language，IML）和它的一种实例天文仪器标记语言（Astronomical Instrument Markup Language，AIML）——参见<http://pioneer.gsfc.nasa.gov/public/aiml/>。这些ML都是XML词汇表在结构化技术文档、传播科学和技术信息方面的经典应用。此外，XML的这类用途也为开发强大的教育类工具提供了坚实的基础。

### 1.5.2 商业词汇表

商业是计算机应用得最为广泛的一个领域。世界金融活动的大量信息使用各种各样的格式在计算机网络中进行传递。大多数的这类数据格式都是专用的；要真正全面地了解国际货币转账的协议几乎是不可想象的！但是，财政和商务信息确实需要在企业之间进行交换，而这些信息将会因通用信息标准而受益匪浅。

商务活动既交换产品也交换货币，我们通常把这类活动叫做交易。这类交易经常会牵涉到交换纸面上的正式法律文书。通常，这些文书可以使用电子数据交换（Electronic Data Interchange, EDI）标准进行电子交换。EDI定义了一种是大多数商务活动基础的格式，它适用于许多商家对商家的交易。北美地区 EDI的起源可以追溯到 70年代早期的运输数据统筹委员会（Transportation Data Coordinating Committee, TDCC）。90年代前期，ANSI发布了X12标准（即为大家所熟知的“ASC X12”）。美国地区所进行的这类标准的开发受非盈利组织数据交换标准协会（Data Interchange Standards Association, DISA）的监督。

当然，和大多数北美标准一样，X12也不适用于世界上的其他地区。大多数国家使用美国商业及运输电子数据交换监督标准（United Nations/Electronic Data Interchange for Administration, Commerce and Transport, UN/EDIFACT）（参见<http://www.unece.org>）。UN/EDIFACT的维护、开发、发展都是UN/EDIFACT工作组（UN/EDIFACT Working Group, EWG）的职责，它是商业贸易精简监督中心（Centre for Facilitation of Administration, Commerce and Trade, CEFACT）下辖的一个授权工作组。欧洲委员会的开放信息交换（Open Information Interchange, OII）服务则是了解关于UN/EDIFACT的信息的另一个来源。

Internet开放交易协会（Open Buying on the Internet Consortium, OBI）是一个非盈利组织，专门从事Internet上的商家对商家的交易的开放标准的开发（参见<http://www.openbuy.org>）。虽然OBI v2.0现在还是基于ASC X12标准的，但它正在被修改为一种XML词汇表。同时，Ariba和Microsoft公司也出于类似的目的开发了CommerceXML（cXML）（参见<http://www.cxml.org/home>），CommerceOne参见<http://www.commerceone.com>则提供了通用商务库（Common Business Library, CBL）。我们将在第12章了解电子商务的有关信息。

Microsoft公司还启动了BizTalk Framework的初期工作（<http://www.biztalk.org>），它得到了来自SAP、CommerceOne、Boeing和BP/Amoco之类公司的支持。这个关于XML计划和消息描述的支持库将能够“启动电子商务和应用程序集成”。

另一个得到广泛支持的是RosettaNet组织，它所从事的就是描述如何使用XML、UML和其他通用协议进行商家对商家的信息交换。它的基础由两个主要的部分组成：首先是适用于所有领域的产品的一套技术规范；其次是对公司和商业交易的描述。

几年前，Microsoft、Intuit和CheckFree联合起来开发了一套用于在线财务数据传递的开放规范，即基于SGML的开放财务交换（Open Financial Exchange, OFE或OFX）规范。但是，由于OFX允许包含未定义的元素，所以它从不是完全准确的SGML。1998年，XML的引入也曾被认为是OFX的目的之一。到了1999年春天，OFX和相关标准被移植为一种新的标记语言——交互式财务交换（Interactive Financial Exchange, IFX）——要了解有关信息，请访问<http://www.ifxforum.org/>。

### 1.5.3 法律词汇表

印刷表单的数字表示在商业、法律和医药方面仍然存在一个问题。一种可能解决方案就是UWI.Com的可扩展的表单描述语言（Extensible Forms Description Language, XFDL）。这种XML词汇表支持精确的布局、计算、输入验证、数字签名、以及法律事务记录和审计跟踪。（要

了解这方面的详细信息，请访问 <http://www.uwi.com/xfdl/>。）

#### 1.5.4 医学词汇表

医学信息覆盖了XML应用的方方面面。医学参考文献和相关科学论文使用XML标记有助于转换成易于演示的各种格式；而且它们在使用结构化的搜索时会更容易得到结果，与简单的布尔逻辑的自由文本搜索相比，这种搜索会更强大、更集中。在各类组织之间——从医院到药店到保险公司和（或）政府中介——必须在大量的、独立的计算机系统之间交换临床、财务和行政信息。1987年，一次ANSI X12会议提出了健康7级（Health Level 7，HL7）标准用于目前美国主要的大型医院，而且还应用到了澳大利亚、西欧、以色列和日本。虽然这个标准目前没有使用XML实现，但向XML版本移植的进程已经启动。

#### 1.5.5 计算机词汇表

Internet和WWW也需要设法描述不同来源和各种格式之间的信息交换。下面我们将列出其中的一部分。

最早的XML词汇表之一是频道定义格式（Channel Definition Format，CDF），这个由微软公司提出的格式允许Web站点提供定期的信息更新（被称作“频道”），自动传送给相关订阅者。遗憾的是，CDF提交给W3C已经超过两年半，而且不再与XML兼容（因为它基于的是XML规范的早期草案，其中包含目前属于不正规的语法）。

还有一种被称作结构化图形格式（Structured Graph Format，SGF）的XML格式，根据正式的结构图算法来描述Web站点的结构（<http://www.isl.hiroshima-u.ac.jp/projects/SGF/index.html>）。

Netscape的Mozilla项目使用的是基于XML的用户接口语言（XML-based User interface Language，XUL），为描述用户接口提供了一种跨平台的途径。XUL包括用于UI控制的元素类型，用于内容数据的HTML4标记，用于用户事件处理的JavaScript（参见<http://www.mozilla.org/xpfe/xp toolkit/xulintro.html>）。

IBM的小型标记语言（Bean Markup Language，BML）是一种基于XML的组件配置标记语言，专门为JavaBean组件模式而定制。BML可以用来描述如何创建新的JavaBean，访问和（或）配置现有的JavaBean，将一个JavaBean的事件与另一个绑定起来，以及调用其他JavaBean的任意方法（参见<http://www.alphaworks.ibm.com/formula/bml>）。

网络上的软件销售可以使用开放软件描述格式（Open Software Description，OSD）来控制，该计划由Marimba和Microsoft公司提出（参见<http://www.marimba.com/products/wihitepapers/osdwp.html>）。

现代数据库设计使用基于数据建模的严格设计流程，它们通常使用统一建模语言（Unified Modeling Language，UML）——虽然它也是以“ML”结尾，但它并不是基于XML的标记语言。元对象工具（Meta Object Facility，MOF）是一种用于发布数据知识库和元数据管理的对象管理组（Object Management Group，OMG）标准。OMG是大型系统公司（如IBM和Unisys）和数据库软件公司（如Oracle、Rational和Sybase）以XML元数据交换（XML Metadata Interchange，XMI）规范的形式将XML、UML和MOF合并而成。同时，它也不是严格意义上的XML词汇表

(因为它是XML的超集),但XMI是XML强大功能的一个很好的例证(参见 <http://www4.ibm.com/software/ad/features/xmi.html> )。

公共电话交换网络应该是最保守的技术领域之一。多年来,这个网络使用的是非常复杂的协议,信令系统7 (Signaling System 7, SS7)。最近,出现了几种基于XML的替代产物,其中包括呼叫策略标记语言 (Call Policy Markup Language, CPML——参见 <http://www.dticorp.com/ESP%20white%20paper.htm> )。这是传统专用电信行业向开放标准发展时的一个副产品。IP语音 (Voice-over-IP) 则是语音与包交换网络结合的另一种趋势,它得到了朗讯科技 (以前的贝尔实验室)、北方电讯和思科公司的全力支持。

前面你可能已经注意到,许多这类词汇表都是由一组公司协作制定的,其目的就在于能够简化数据交换。下面我们将详细了解它们的意义所在。

## 1.6 XML技术的主要特性

现在,我们已经了解了XML的起源,它们为什么和在哪些方面适合于Web的分布式应用体系结构,以及一些已经使用XML开发出的特定词汇表,下面,让我们按照本书章节的顺序来看一看与XML核心功能有关的特性和规范。

### 1.6.1 格式正规的XML

XML的语法规则是我们利用它进行任何工作的基础。下一章,我们将帮助你理解“格式正规的”XML的特性。你会了解元素是什么,如何使用它们,以及如何使用属性修改它们。我们将再次提出XML词汇表的概念,因为我们要开始讨论XML的使用。我们还要开始涉及在应用程序中操作XML文档的标准解析器。最基本的,格式正规的XML会遵守W3C的XML 1.0推荐标准的语法要求。解析器就是检验一个文档是否符合XML语法规则的处理工具(稍候我们会详细讨论)。下面是一个简单的格式正规的文档(先不用考虑语法的精确细节):

程序清单 1-6

```
<?xml version="1.0" ?>
<pet_store store_ID="11218976">
  <purchases customer_ID="334343">
    <creature>
      <creature_type>llama</creature_type>
      <species>Vicuna</species>
    </creature>
    <feed>
      <daily_feed>Ruminant grain feed</daily_feed>
      <daily_feed_quantity>2</daily_feed_quantity>
    </feed>
  </purchases>
</pet_store>
```

通过下一章的实践举例,我们将开始了解将内容与显示相分开的好处。与XML相关的HTML确实有某些限制。这也是普通的Web浏览器就是普通的HTML应用程序的原因。对于所有

的用途，HTML面临着许多天生的问题，而 XML在保持了强大的功能（HTML正是依靠它们改变了计算技术的面貌）和简单性的同时提供了相应的解决方案。XML的基础就是你向下一代Web体系结构迈进的出发点。

### 1.6.2 文档类型定义

一旦你坐下来书写XML词汇表，马上会产生一种迫切的需要：如何指定书写XML文档的规则？如果任何人都能够使用可扩展标记语言创建自己的标记词汇表，那么我们如何确信能够在应用程序中使用XML文档呢？答案在于一套被称作文档类型定义（Document Type Definition，DTD）的信息集合。这些定义保存了由设计者添加的、用于扩展XML核心规则的部分，并创建用来描述某些问题或状态的词汇表。这是你在了解XML词汇表的结构的过程中遇到的第一个机制。我们所强调的未来Web应用程序体系结构的诸多好处都有赖于此机制。通过学习DTD，你将会充分掌握如何验证应用程序之间交换的文档，并开始有机会及时发现新的词汇表。稍后，我们将看一看其他用来表述XML文档结构的机制，但DTD目前仍是唯一得到官方认可的途径。

接下来是前面提及的宠物商店的例子的继续，不过这次有了一个DTD（再次提醒不必太在意语法细节）：

程序清单 1-7

```
<?xml version="1.0" ?>
<!DOCTYPE pet_store [
  <!ELEMENT pet_store (purchases?) >
  <!ATTLIST pet_store storeID CDATA #IMPLIED >
  <!ELEMENT purchases (creature*, feed*) >
  <!ATTLIST purchases customer_ID CDATA #IMPLIED >
  <!ELEMENT creature (creature_type, species)+ >
  <!ELEMENT creature_type (#PCDATA) >
  <!ELEMENT species (#PCDATA) >
  <!ELEMENT feed (daily_feed, daily_feed_quantity)+ >
  <!ELEMENT daily_feed (#PCDATA) >
  <!ELEMENT daily_feed_quantity (#PCDATA) >
]>
<pet_store store_ID="11218976">
  <purchases customer_ID="334343">
    <creature>
      <creature_type>llama</creature_type>
      <species>Vicuna</species>
    </creature>
    <feed>
      <daily_feed>Ruminant Grain Feed</daily_feed>
      <daily_feed_quantity>2</daily_feed_quantity>
    </feed>
  </purchases>
</pet_store>
```

正如大家能够看到的，DTD有自己的语法规则，但它们使你能够非常清楚地指出对于特定类别XML文档，哪些是允许的，哪些是不允许的。这直接导致了验证和非验证的解析器的区别。非验证的解析器仅仅根据XML语法的核心规则判断文档是否是格式正规。验证的解析器则还要



根据DTD进行检验，以根据 DTD规则决定文档是否合法。但是，为什么我们还需要非验证的解析器呢？你如何将验证的解析器与 DTD结合起来呢？你如何根据 DTD来描述问题或应用呢？这些问题都将在第3章进行讨论。

### 1.6.3 数据建模

XML应用程序成功的关键因素之一就是 XML词汇表的高效性。词汇表是元素及其属性、以及你所指定的文档结构的规范。虽然只要有一个不怎么样的数据库模式就能够编制数据库应用程序，但没有一个有效的词汇表，你就不再能够创建一套好的支持 XML的应用程序。

### 1.6.4 文档对象模型

一旦拥有了为需求而设计的词汇表，就可以考虑应用程序的其他方面了。在一些新的 Web应用程序的结构中，应用程序必须处理 XML文档，及文档的各个部分。客户端创建请求，而服务器解析这些请求，构造新的请求，并进行响应。DOM是完成这项任务的一个 API，其中指定用于处理HTML和XML文档的一系列对象和接口。W3C维护DOM推荐标准，它是两个用于XML文档且受到广泛支持的API中的一个（另一个是针对XML的简单API，下一节将进行介绍）。

DOM提供了文档的三个结构化视图。与 DOM兼容的解析器读取整个文档，并通过在内存中构造一个对象树而提供文档的一个视图。文档的主要组件结构是对象树中的节点，访问对象树并处理它是通过使用 DOM接口浏览解析树实现的第 5章将介绍 DOM对象模型，并讲解如何使用 DOM接口进行处理XML文档的编码工作。

### 1.6.5 XML 简单API

在使用XML文档时，另一个重要的 API是XML简单API（Simple API for XML，SAX）。与 DOM不同，SAX并非是一个标准组织的产品。它是早期 XML开发过程中，许多XML开发者为了寻求一种有效的API而创造出来的产品。SAX至今依然流行的原因是因为它通过另一种途径来访问XML文档。SAX并不把应用程序的注意力集中在整个文档的树状图上，而是在解析时提供事件驱动。事件的内容往往是这样的形式，“这里是一个起始标记；这里有一些元素内容；这里是一个结束标记”，等等。兼容SAX的解析器并不保留文档；而是在使用它的程序处理文档的每个部分时发出通知。对一个事件所做出的反映取决于使用解析器的程序。一个程序对文档状态的维护要承担全部责任。这样就可以根据需要来维持适量的信息以满足激活应用程序的需要。

我们可以想象，这样一来，解析器会相当短小精悍，对系统资源要求甚少。对于处理大的XML文档来说会非常理想。如果你面对的是一个由各部分目录组成的 16M大小的XML文档，肯定不希望将它们都读入到内存中。虽然目前 DOM是最常见的XML的API，但找到SAX解析器也不困难。第6章我们会了解SAX是什么，如何使用，以及何时使用。你看到的将是对如何使用兼容SAX的解析器处理XML文档的精彩介绍。

### 1.6.6 命名空间和模式

我们真诚地希望，随着大家越来越多地了解到 XML和它给应用程序带来的好处之后，能够

编制出更加巧妙的文档和词汇表。你也可能想在结构的自动检测方面了解更多的信息。

如果你尝试用过 DTD 来完成这项工作，会遇到一些障碍。解决的办法就是 XML 的模式 (Schema) 和命名空间 (Namespace)，针对它们的开发工作早已开始。

随着 XML 的发展，正在开发的 XML 词汇表已经越来越多。其结果之一，就是开发者开始遇到来自别的开发者的、对于解决自身问题大有好处的词汇表。这类词汇表不能满足所有需要，但它们确实能够极大地简化开发新词汇表的工作。当你正在开发一项解决常见问题的新应用时，如果有人已经考虑过这个问题并编制了一个 XML 词汇表可是个不错的消息。如果你不需要它所有的部分，可以借用其中的内容重新编制一个。即使孤立无援，你也会希望分阶段、有步骤地完成；将一个大的问题分解为若干小的问题会有利于它的解决。

XML 的研究者意识到了这一问题并设计出命名空间作为解决办法。命名空间是文档设计者们希望利用的名称的资源。通过它们的属性，你可以利用其他资源并毫不犹豫地采纳其中的部分内容。如果你应付的是一个大难题，可以编写一系列词汇表，其中的每一个只涉及整个问题的一小部分，然后使用命名空间将各类 XML 词汇表混合起来。

这样做的问题之一是 DTD——XML 1.0 中指定词汇表的办法——不允许你使用命名空间。我们在第 7 章会讲到，DTD 还有其他问题。现有的解决方案是模式，使用 XML 语法的替代品。这一章会解释几种采用模式的途径，并使用本书的目录举例来说明它的好处。第 3 章开发的 DTD 将被转化为模式的形式。你会看到一些与 DTD 相比，模式的过人之处。最后，当我们创建一个使用基于浏览器的脚本代码的索引程序时，你会全面了解如何编制模式的代码。

### 1.6.7 链接和查询

HTML 定义的特征之一就是链接。在这种标记语言当中，它可能是最受欢迎的一个部分。在使用时，关系型数据库会形成一个链接自身的表单，使用外键获取另一个表的数据。任何技术如果要应用到稳定的数据库中，就必须具备某种与数据主体链接的能力。XML 也不例外。许多开发者希望把一个 XML 文档与另一个链接起来，或者将非 XML 的内容与 XML 文档链接起来。如果在 XML 中有链接的话，图像和二进制数据就能够和 XML 文档配合起来使用。所以，链接是 XML 研究者们投入极大精力的一个重要领域。W3C 正在制定相关的几项计划，其中比较吸引人的是 XLink 和 XPoint。第 8 章将向你介绍这些计划。你会学习如何在自己的应用程序中使用 XML 链接。由于对 XML 解析器中的链接来说没有标准支持，所以看到的是可能实现的链接的常见形式。

XML 研究者们感兴趣的另一个领域是查询。我们必须由某种途径能够把一些规则传递给 XML 解析器，然后得到符合这些规则的文档片断集合。有了这种能力，大的 XML 文档就成为了某种类型的数据库。除了有类似于数据库查询的特征以外，XML 的查询也是转换操作的中心，操作和改变 XML 的关键模式。第 8 章将从为 XML 提出的主要计划中提炼出 XML 查询的语法知识。

### 1.6.8 转换 XML

转换 (transformation) 是一项非常强大的 XML 技术。通过转换，编程者可以根据应用在一个文档中的一套规则将这个文档转换成另一种形式的文档。XML 转换用于在类似的 XML 词汇

表之间进行转换，以及把 XML 文档转换为其他基于文本的文件格式，比如以逗号分隔值的文本文件。对于 Web 开发者来说，这是一个非常重要的工具。如果要合并现有的资源，你需要执行一些有效的转换以得到统一的格式。如果你在同若干伙伴进行合作，比如 B2B 运作，就肯定需要将文档从一种格式转换到另一种格式。第 9 章将告诉你如何做到这一点。转换的有趣在于映射规则是在一个独立的文档而不是代码中指定。如果你需要动态地在一系列相关格式之间进行转化，可以开发一系列的规则文档。在运行的时候，你在决定使用哪一个转换，然后将正确的规则应用到手头的文档上。对于 B2B 模式和供应链应用来说，这项功能特别有用。

一种被称作可扩展样式语言（Extensible Style Language，XSL）的用于 XML 的样式语言中的部分内容是最新形式的转换。实际上，转换语言被称作 XSL 转换部分（XSL Transformation，XSLT）。XSLT 主要用来识别 XML 文档以便使用 XSL 样式。即便它并没有被设计成一种通用的转换语言，XSLT 的弹性还是相当大的，允许你进行大多数 XML 中的转换、排序和组织工作，而不需要编写自己的程序代码。实际上，你根据元素显示的先后关系编写 XML 的转换规则。在第 9 章，你会了解编程者可以用来指定转换规则的 XSLT 语法。你将学会如何根据自己提供的规则定位文档中特定元素的技术。然后了解如何指定在源 XML 文档上执行的转换。完成这些之后，你就会掌握使用 XSLT 执行 XML 文档的数据驱动操作的方法。

### 1.6.9 XML 和数据库

用 Web 前端连接关系型数据库是相当常见的。但是，XML 的数据模式天生就是层次结构，这会使得在将它们和大多数普通数据库使用的关系型模式相匹配时遇到一些困难。虽然使用 XML 作为关系型数据库的接口并不直接利用 XML 的独特功能，但它能将现有的数据引入到新的系统中。既然 XML 是一种流行的、与平台无关的连接方式，那么编程者肯定需要一条途径作为 XML 和数据库之间的接口。许多数据库厂商在意识到这一事实后，已经开始在自己的引擎中增加对 XML 的本机支持。

XML 从本质上讲基本是层次性的，但大多数现有的、常见的数据库系统都是关系型的。在两者进行映射时会导致某些问题。第 10 章会探讨将 XML 词汇表映射到关系型表以及相反操作时的有效策略。在了解了 XML 能够从哪些层次去改变应用程序与数据结构的接口方式后，这一章将开发一个普通的脚本，它用来定义一个关系型数据库中的、能够映射到自己的 XML 模式的表。有了这些工具，你就可以完成自己的应用中负责连接服务器程序和后台数据的部分。

### 1.6.10 服务器到服务器

支持 XML 的 Web 应用程序能够连接起来形成系统。到目前为止，大多数编程者都把 Web 应用程序严格地视作客户机-服务器结构。Web 客户机从 Web 服务器上获取信息。当一个服务器访问数据库时，它并不会向另一台 Web 服务器寻求帮助。随着时间的流逝，越来越多的应用程序资源开始由 Web 服务器控制，这样一来，实现多个服务器一起来解决问题的能力就变得重要起来。由于一台服务器能够呼叫另一台服务器以寻求数据和处理能力，我们可以在现有应用的基础上编制出精致的分布式程序。既然这些系统经常使用不同的服务器软件和分布式计算技术，所以需要 XML 来提供一个抽象层来集成不同的系统。从另一个服务器获取 XML，进行操作，然后把

结果传递给客户机是可以满足客户机请求的需要的。许多使用 XML来有效地完成这类工作的技术正处在开发当中。其中包括 XML-RPC，简单对象访问协议（Simple Object Access Protocol，SOAP），以及分布式Web数据交换（Web Distributed Data Exchange，WDDX）。

XML-RPC是一种远程执行驻留在服务器上的进程的协议。这与传统的 RPC非常相似，后者允许我们命名一个过程以便执行并可以提供一个参数列表。XML-RPC将XML当作是完成这类工作的途径，这样能够减少与特定平台有关的问题。因为是 XML，所以对于编程人员来说，使本地资源可用于远程执行就简单化了。XML-RPC已经在许多常见平台上得以实现。

SOAP与XML-RPC类似，也使用XML来访问HTTP之上的对象的方法和属性。XML用来描述被调用的方法和被传递的数据，这样能够避免对任何特定类型的分布式对象技术的依赖。

WDDX是一种使用XML串行化数据结构的技术。例如，它可以用作通过 Internet返回数据库结果的低级机制。

第11章将为你提供这方面的知识以及其他使用 XML进行服务器-服务器通信的方法。对于网络通信问题来说，XML-RPC、SOAP和WDDX是非常明智的XML应用方案。掌握了这些技术，你就能够更好地在现有的和未来的基于 Web的资源的基础上构建多层分布式系统。

当你读完第11章之后，你会对何时何地什么样的技术会更有效有更其清晰的概念。你会看到一个服务器-服务器通信的实践举例，了解到当第一个服务器上没有任何需要的书籍时，如何从另一个服务器上获取图书书目信息。

#### 1.6.11 电子商务与XML

XML被广泛地认为是解决应用程序之间的数据交换问题的方案。电子商务，特别是商业组织之间的交易，走到了XML应用的最前列。许多年来，电子数据交换（Electronic Data Interchange，EDI）是商业结构之间数据交换的标准（你可以访问<http://www.geocities.com/WallStreet/Floor/5815/>了解更多信息）。但是，EDI有许多缺陷，这极大地限制了它在大型商业结构和高额交易中的使用。它使用特定的网络和数据格式来交换数据。结果是EDI系统的实现成本高，时间长。典型的小型企业根本负担不起这些。XML利用开放的Internet所带来的益处则改变了这些。定义适当的XML词汇表能够符合原先的EDI结构。这使得XML的EDI实现能够充分利用现有的第三方的XML工具和解析器。

在第12章中，你会看到 XML是如何应用到EDI中，创造出一种通用的、低花费的电子商务结构。这与我们在前一节讨论的服务器-服务器通信来说是一个巨大的飞跃。当信息从一个合作者传递到另一个时，它所跨越的不仅仅是不同位置的服务器，而且是不同的格式。这样一来，词汇表创建工具变得越来越重要。同样，数据转换的工具也是不可或缺。第12章将会把这两个问题紧密结合在一起。

XML的流行和强大也使得它在数据交换领域占据了一席之地。看起来，简单性是它的一个很大的优势。无论你是否对 XML在EDI上的应用感兴趣，第12章中介绍的能够简化两个应用程序之间的数据交换的其他标准都将有助于你解决自己在计算技术方面的问题。看一看我们的图书目录的例子。当你在向销售商展示自己的目录时，应该怎么做呢？我们不可能假设他们只使用我们的词汇表，如果他们要购买任何书籍，就必须拥有一种交易机制。解决的办法就是用于



交换这类数据的标准交易词汇表。

### 1.6.12 使用样式

虽然我们曾经强调过将数据与表达方式相分开的重要性，以及直接把一个应用程序同另一个程序连接的意义所在，但最后我们还是要在用户面前显示 XML 格式的数据。即使是在新的 Web 体系结构中，基于浏览器的传输也总是很重要的。在不久的将来，基于浏览器的客户端仍然会占据统治地位。甚至某些应用程序所关心的主要就是这类数据的显示。如果这正好是你的需求，那么你就必须了解样式的使用。在这里我要强调一点，XML 研究人员曾经设计出几种数据驱动技术来将原始的 XML 转变为格式丰富的可视化内容。数据驱动的方式与传统的在脚本代码中的硬编码样式有所不同。XML 样式对于 Web 开发人员来说是一种重要的工具。它是快速实现 XML 数据的用户接口的途径，并将决定有多少方式可以应用到传递给客户端的单个数据主体。它使得 Web 服务器无需更改生成数据的编码就可以把 HTML 传递给一个特定的客户机。它使得 Web 客户机能够让用户在不同视图之间切换，而无须再次到服务器打一个来回以获取同样的信息。

用于 XML 样式的技术在其复杂性和实践意义方面千差万别，我们将在第 13 章中介绍最重要的部分。可能最简单的技术就是层叠样式表单（Cascading Style Sheet，CSS）。这些并不完全是 XML 研究人员的成果；它们是 XML 开发者从针对 HTML 的 Web 开发过程中继承，并加以扩充而成。它们是将特定的样式信息传递给命名的 XML 元素的基本手段。这与你所熟悉的字处理程序更改文档中的样式信息（字体、颜色、倾斜度、颜色、等等）的方式一样。

另一个样式应用是可扩展样式表单语言（Extensible Stylesheet Language，XSL）。这个 XML 的副产品将 CSS 所关心的样式内容与 XML 天生特性结合在了一起。编程者使用 XSL 指定如何将 XML 数据映射为可视化内容。与 CSS 不同，XSL 允许你指定基于 XML 数据前后关系的样式，甚至可以通过在自己的 XSL 样式表单中嵌入脚本代码来执行处理任务。

### 1.6.13 无线应用协议和 WML

使用 XML 标记的数据所占据的空间会多于以本机二进制格式编码的数据。对于 Web 应用来说这通常不是问题，因为即使是拨号连接所提供的带宽也足够传输大量的 XML 数据。但是，新的无线设备所支持的速率往往更低。这些设备是从由浏览器组成的同构 Web 向由传统的和非传统的客户端组成的异构 Web 发展的第一个步骤。这些客户端有着各种各样的能力，所以需要有各种各样的技术来传递数据和内容。那么，我们能够在蜂窝电话或者个人数字助理这样的非传统客户机上使用 XML 吗？

对这一问题目前有了一种新颖的解决方案——无线应用协议（Wireless Application Protocol，WAP）。它巧妙地利用了 XML 令牌化的二进制表示，利用标准化的、自描述形式的 XML 来满足二进制数据的精炼形式。与 XML 不同的是，WAP 在许多层次起作用。它为网络协议的各个层次和应用程序层指定组件。一个名为 WAP 论坛的专业组织是这一计划背后的主要推动力量。它正在和 W3C、IETF 以及其他标准组织一起推广这一协议。

第 14 章将会告诉编程者 WAP 的组成，无线标记语言（Wireless Markup Language，WML）



的内容，以及它们是如何协同工作在低带宽的无线网络上传送自描述的数据。

## 1.7 XML的应用程序举例

接下来的两章我们将讨论开发一套标记语言版本的出版商书籍目录（非常凑巧的是它也是属于Wrox的）。对目录的标记化和它的模式将会贯穿全书，以说明 XML技术的使用的方方面面，在后面的每一章中我们会逐个涉及这些技术。每个章节中提到的 XML用法与该章的标题有着密切的关系，但各章之间则是相当独立，而每个例子都会利用到前面章节中学到的内容。但是，为了综合了解 XML在应用程序中的使用，我们建议你学习本书后面部分关于整体实例的内容。数据目录应用程序简单易懂，在其中我们演示了 XML如何用于传统文档标记和普通的数据建模。在其中我们会开发一个 XML词汇表，其中包含了典型的出版元数据。对于那些与任何出版商和客户/消费者之间的交易有关的结构化数据来说，这个词汇表能够成为信息发送的基础，这些交易包括：搜索书籍；罗列书籍名称及价格；请求了解一本书籍的更多内容；监控书店中书籍的库存情况；将数据或内容发布到 Web上；等等。由此得到的 XML文档使得信息能够在整个商业过程中的任何阶段使用，因为它们需要信息的交换是独立于平台的。

## 1.8 小结

本章我们讨论了通常标记语言开发的目的，并捎带谈一些 XML的动机。接下来我们研究了 XML对Web应用的体系结构的影响。在这些影响中，有些是潜在的，有的被人们意识到，但有一点非常清楚——技术的整体发展方向是使用数据交换的开放标准来构建分布式应用程序。

然后，我们更细致地深入到使这种发展得以实现的 XML特性。至关重要的是，XML是一种元语言的语法标准，它允许编制针对特定任务的词汇表但却可以通过普通的 API使用。我们列出了一些最重要的、基于标准的词汇表，这些针对各种类型网络化通信系统的词汇表或者已投入使用，或者正处在开发研制阶段。

最后，我们简要讨论了与 W3C XML 1.0推荐标准有关的主要 XML技术。

下面，让我们来看一看 XML的核心语法。