China-pub.com
下载

第1章 XML

本章介绍可扩展标记语言（Extensible Markup Language，XML），下一章将介绍可扩展样式表语言转换（Extensible Stylesheet Language Transformation，XSLT）。本章适用于较熟悉Microsoft的SQL Server 2000的读者，但是对XML及其处理知之甚少的读者也适用。由于这两章的内容是介绍性的，熟悉XML和XSLT的读者可以跳过它们，而直接从第3章开始。

本章包括了以下内容：

- XML的由来和前景
- XML文档的组成
- 如何编写基本的DTD
- DTD存在的问题
- 将要取代DTD的W3C模式规范
- 模式文档的组成
- 如何编写基本的模式声明文档

本章不会过于详尽地讨论XML、文档类型定义（Document Type Definition，DTD）或XML模式，本章的主要目的在于介绍XML文档、DTD、DTD所存在的问题以及针对这些问题的提议，即XML模式解决方案。在进入这些主题之前，先让我们看看XML的发展历史。

1.1 XML的由来

与HTML类似，XML也是从80年代中期出现的标准通用标记语言（Standard Generalized Markup Language，SGML）派生出来的。SGML的出现带来了出版业的革新。如果你想了解SGML的历史及其在出版业中所起的作用，可以从以下网址获得著名的白皮书：

http://www.arbortext.com/data/getting_started_with_SGML/getting_started_with_sgml.html

SGML是用于定义用电子方式表示文本的设备无关、系统无关方法的国际标准，它也被称为ISO 8879：Information Processing —— Text and Office Systems —— Standard Generalized Markup Language，([Geneval]：ISO，1986)。更准确地说，SGML是一种元语言（metalanguage），是用于形式化地描述一种语言（在这种情况下是一种标记语言）的工具。在进一步讨论之前，我在这里先介绍一些术语。

标记（markup）或（同义词）编码，是使得对文本含义的解释变得明显的方法。从这个意义上讲，所有被打印的文本都是已编码的。标点符号、大写字母的使用、字母在页面上的排列、甚至单词之间的空格都可以被看作某种类型的标记。

所谓标记语言，是指用于对文本编码的一些约定。标记语言必须指定如下要素：

- 哪些标记是允许的。

需要全本请访问 www.ertongbook.com

China-pub.com 下载

- 哪些标记是必需的。
- 标记是如何与文本进行区分的。
- 标记的含义。

SGML有四个特征与其他标记语言不同，这四个特征是：

- 强调描述性的而不是过程性的标记，这使得它只是把文档的各个部分进行分类。
- 用文档类型概念描述文档结构。
- 在表示脚本时与任一系统无关。在该脚本中，文本用来允许数据从一个系统传入另一个系统而无信息丢失。
- 学习起来比较复杂且难度较大。

前三个特征是SGML的正式概念，而第四个特征是我给出的。由于SGML太复杂，使得SGML的子集——HTML和XML得到广泛应用。

我们对于HTML或多或少都有所了解。在对于XML的夸大其词的宣传中，你会听到这样的说法：现在XML已经替代Java语言、面向对象技术以及其他几项现有的技术，从而成为全球性的通信解决方案。它被夸大为各类不同的新的或传统的通信与数据技术之间的管道，以便以一个同源的商业解决方案来进行通信（见图1-1）。

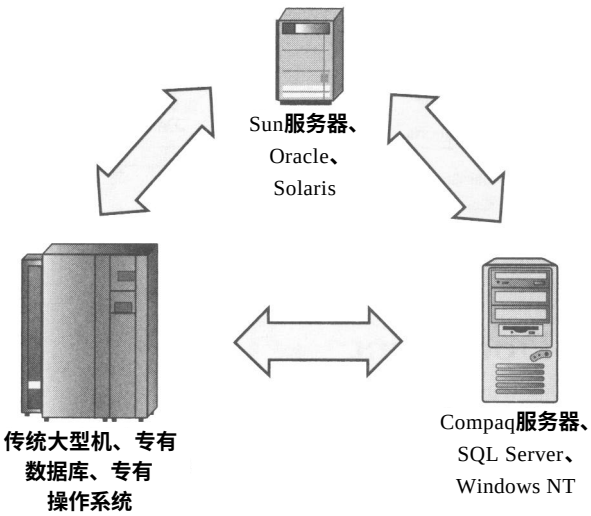

图1-1 XMP前景

也许你曾经对XML的宣传的真实性表示过怀疑。试想想三年前有关Java的宣传吧，那个时候到处都在宣扬Java的神话。而现在，那些浮夸的宣传都已经消失了，然而现在Java也确实可以提供企业解决方案。

Gartner小组绘制了被称作Hype图（Hype Chart）的图。这一图（见图1-2）给出了某些技术的出现及其成熟周期曲线。看看图1-2中XML的位置，可以看出它很快就要成为一种企业开发工具。

China-pub.com
下载

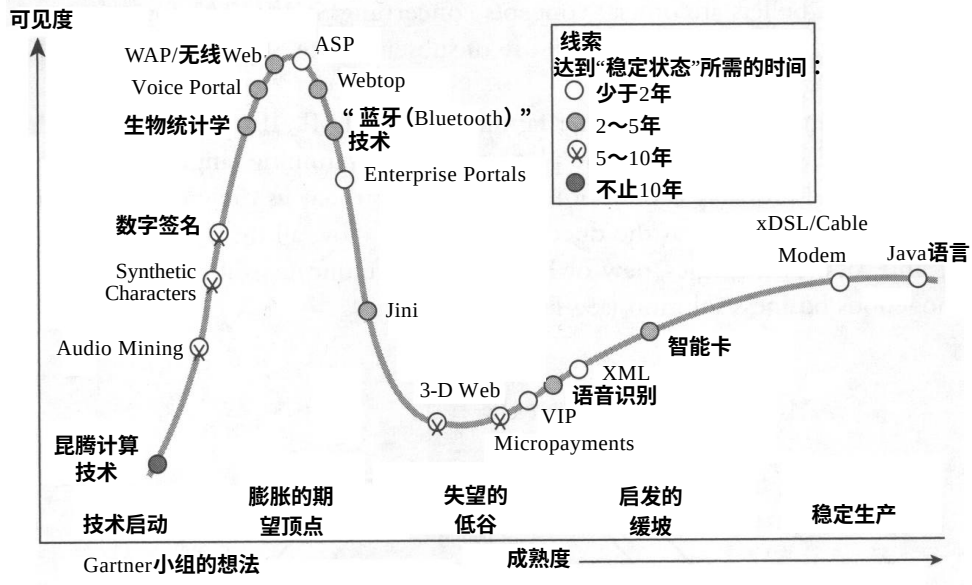

图1-2 Hype 图

1.2 XML文档

XML具有如下特征：

- 无须注册，与平台无关，可获得良好支持。
- 万维网联盟推荐标准（XML 1.0，W3C Recommendation，February 1998）。这使得XML成为全球性的标准。

文档在W3C中通过的步骤：

工作草案（Working Draft）。工作草案通常代表正在进行的工作以及W3C对于在某一方面开展工作的委托。它并不代表W3C中大多数人对于该文档的意见。

候选推荐（Candidate Recommendation）。候选推荐是一份稳定的工作草案，由主管向委员会提议，在实际应用中进行实现检验并反馈效果。

提议推荐（Proposed Recommendation）。提议推荐是一份经过实际应用检验的候选推荐，提交给顾问委员会进行复审。

推荐标准（Recommendation）。推荐标准反映了W3C中大多数人的意见，经主管同意才能提出。W3C认为，推荐标准中所反映的技术或者思想适宜于广泛的应用，因而W3C要推广其应用。

以上过程的图示见图1-3。这些步骤的确定性的解释见位于http://www.w3.org /Consortium/Process-20010208/tr.html的W3C的过程文档。

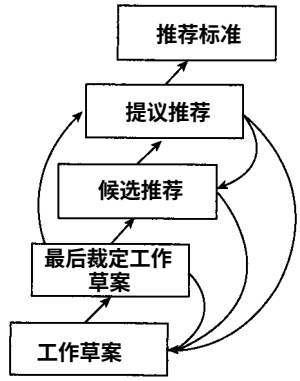

图1-3 XML形成过程

China-pub.com 下载

- XML是一种更加易于传输原有或者其他 B2B（企业对企业的电子商务）信息的解决方案。专有平台、数据格式、模式、业务规则等都可以通过一个简单的通用线程来链接，而且可以被接收方一致地处理。
- XML既可用于描述信息结构，也可以用于描述信息本身。可以将它看作一种数据描述语言。
- XML 包括一个技术家族或一组技术，如表 1-1所示。

表1-1 W3C XML技术

名称/URL地址	状态（2000年11月）	描 述
可扩展标记语言（XML）1.0 http://www.w3c.org/TR/REC-xml	推荐标准	XML的基本语法
XML信息集 http://www.w3c.org/TR/xml-infoset	工作草案	描述一个抽象的数据集合，其中包含了可以从XML文档中获取的有用信息
XML中的名字空间 http://www.w3c.org/TR/REC-xml-names	推荐标准	针对XML1.0的一个附录，该附录通过将XML文档和URL引用标识的名字空间联系起来，提供了一种在可扩展标记语言中限定元素和属性名称的简单方法
XML Base（XBase） http://www.w3c.org/TR/xmlbase	候选推荐	另一个附录，类似于HTML Base，用于为部分XML文档定义URI基地址
XML Inclusions（XInclude） http://www.w3c.org/TR/xinclude	工作草案	为将多个XML信息集合并为单个的复合信息集指定一个处理模型和语法
Document Object Model（文档对象模型，DOM）第二级 http://www.w3c.org/TR/DOM-level-2	推荐标准	允许操作、建立XML文档的可编程接口
XML模式第一部分：结构 http://www.w3c.org/TR/xmlschema-1	候选推荐	基于XML的语言，用于将XML元素和属性描述为层次结构类型
XML模式第二部分：数据类型 http://www.w3c.org/TR/xmlschema-2	候选推荐	一种基于XML的语言以及一组文本数据类型，用于描述用户生成的数据类型
XML路径语言1.0（XPath） http://www.w3c.org/TR/xpath	推荐标准	用于XML文档中寻址部分的语言，它被设计为由XSLT和XPointer使用
XML指针语言（XPointer） http://www.w3c.org/TR/xptr	候选推荐	利用XPath表达式来验证URI分段
XML链接语言（XLink） http://www.w3c.org/TR/xlink	候选推荐	允许向XML文档中插入元素，以创建并描述资源之间的链接
可扩展样式表语言transformations1.0（XSLT）） http://www.w3c.org/TR/xslt	推荐标准	一种语言，用来提供把一个XML文档变成另一个XML文档或者变成一种不同形式文档的方法

1.2.1 XML处理过程

让我们看看表1-1中列出的XML系统的不同成分是如何一起协同工作的，我把它称为 XML处理过程。图1-4给出了处理过程的所有步骤。下面，我们将讨论这一过程以及表 1-1中各元素之间的联系。

China-pub.com
下载

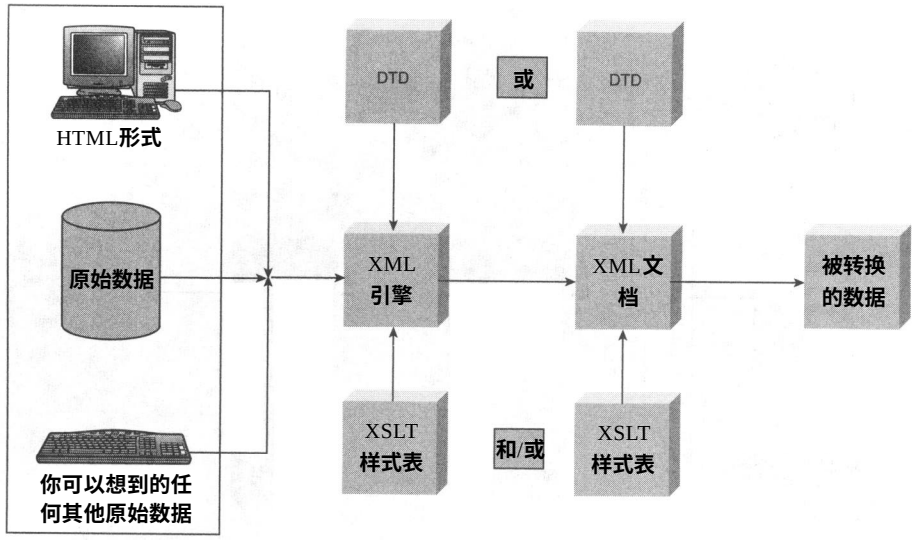

图1-4 XML处理过程

以下是处理过程中的数据流程：

1) 来自数据源的数据被传送到XML引擎。这一步可以通过记录集、游标或者无格式文本文件来完成。

2) 这时用DTD文档来验证创建的XML文档的正确结构。

3) 然后使用XSLT文档来修改文档的结构，以产生另一个不同的XML文档。通常，这一步骤产生的文档用于具有不同数据需求的B2B电子商务数据传输。另外，也可以生成HTML页面，供终端用户浏览。这样做的另一个好处是使格式与内容分离。

4) 如果还未曾使用过DTD，那么可以对此时生成的XML文档进行DTD检验。

5) 某个XSLT文档可以在此时被用来修改其结构。该过程可以发生而与在此之前是否曾经应用过一个XSLT文档无关。

6) 在以上的最后两步之后，转换后的文档就成了最终的形式。

后面，我们将多次回到这一流程图，以描述我们在流程图中所处的位置。

1.2.2 XML文档的组成

图1-5给出了XML处理过程。

代码清单1-1是一个示例XML文档，以下将使用这个文档来解释它所包含的元素。

代码清单1-1 XML文档示例

```
<?xml version="1.0" standalone="no"?>
<!-- ******* Resumes for Potential Hires ******* -->
<RESUMES xmlns='http://www.myorg.net/tags'>
  <PERSON PERSONID="p1">
    <NAME>
      <LAST>Shelton</LAST>
```

需要全本请访问 www.ertongbook.com

China-pub.com 下载

```
      <FIRST>Rick</FIRST>
    </NAME>
    <ADDRESS>
      <STREET>911 Intranet Ave.</STREET>
      <CITY>Canberra</CITY>
      <COUNTRY>Australia</COUNTRY>
      <PC>A34G-90</PC
    </ADDRESS>
    <TEL>(+612) 111-2345</TEL>
    <EMAIL>shelton@somewhere.com</EMAIL>
  </PERSON>

  <PERSON PERSONID="p2">
    <NAME>
      <LAST>Tenney</LAST
      <FIRST>Corey</FIRST>
    </NAME>
    <ADDRESS>
      <STREET>211 Yardwork Circle</STREET>
      <CITY>Roy, UT</CITY>
      <COUNTRY>USA</COUNTRY>
      <ZIP>64067</ZIP>
    </ADDRESS>
    <TEL/>
    <EMAIL>tenney@yardwork.com</EMAIL>
  </PERSON>
</RESUMES>
```

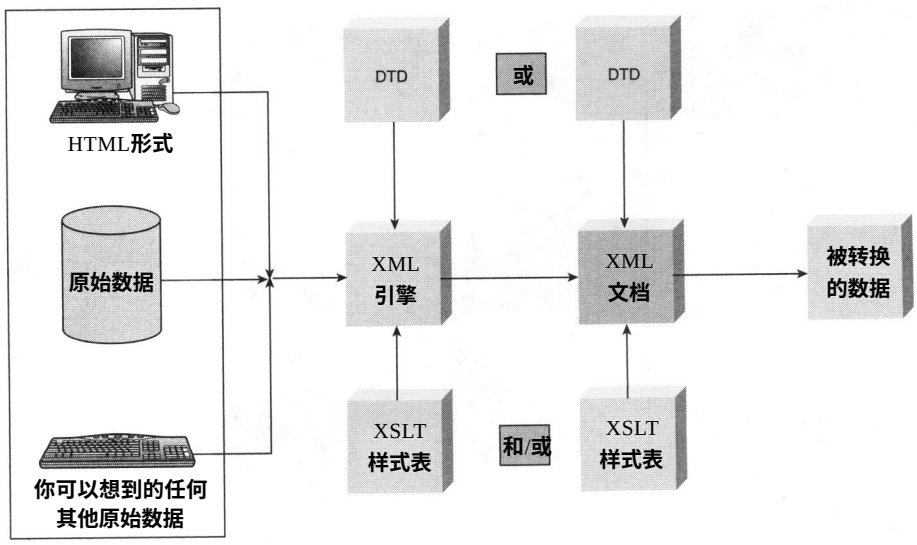

图1-5 XML处理过程中的文档

1. XML声明

XML文档可以用语句<?xml...?>作为开头（尽管这并不是必需的）。是XML声明指定了以下的数据是一个XML文档的XML声明。它还有附加的属性： version、encoding和standalone。

China-pub.com
下载

version是必需的属性，而其他属性则是可选项。为什么版本很重要呢？如果有一天 XML的版本改变了（事实上到现在为止还未改变过），例如XSLT 1.0变成了XSLT 1.1，则必须有一种机制告诉开发者，他们所使用的文档是哪一版本。同时， encoding和standalone属性相对来说也很重要。在示例文档中，以下声明：

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

为我们给出了附加信息——该XML文档符合XML标准的1.0版本，而且有一个外部的DTD文档与之关联（该文档无法独立）。当然，如果该文档有一个在其内部定义的 DTD，那么standalone属性的值应该为“ yes ”。

在以上关于DTD的讨论之后，可能会认为 DTD是必需的；但实际上并非如此。在本章后面的“ 文档类型定义 ”（Document Type definition）和“ DOCTYPE 声明 ”两节中，我们将讨论DTD的更多细节。

> 顺便提一下，位于<?.....?>中的语句称作处理指令。处理指令用于让XML解析器执行任务而非仅仅解释文档中的数据。对于熟悉C或C++的人来说，这些处理指令类似于预处理程序指令。在外部解析器可以理解的前提下，甚至可以创建自己的处理指令。

2. 标记分隔符

XML文档由一个或多个元素组成，这些元素分别由两个（有时候是一个）标记组成。开始标记把元素的名称放在一个小于号（ <）和一个大于号（ >）之间；结束标记与开始标记类似，只是将一个正斜杠（/）放在元素名称的前面。如在示例文档中给出的如下语句：

```
<LAST>Shelton</LAST>
```

这是文档中的元素之一。文档中还有其他一些元素，其中有一个元素的开始标记是<STREET>，结束标记是</STREET>。

3. 元素名

以下是选择元素名的原则：

- 元素名由字母或下划线符号（ _）开头。
- 名字长度通常没有限制，但是要以方便为原则。
- 名字是大小写敏感的。例如：<ADDRESS>与<Address>是两个不同的名字。
- 名字绝不能以xml开头。
- 名字中可以包含任意数量的以下元素：
  - 字母
  - 数字
  - 下划线
  - 点
  - 连字符

4. 根元素

根元素说明文档数据的起始点和结束点。它是文档最外面的元素。在我们的例子中，<RESUMES> …</RESUMES>是根元素。

China-pub.com 下载

5. 空元素

有的元素没有与之相关联的数据，例如：当数据库的一个空域传入 XML文档中时，无论它是否包含数据，这个域仍然存在，因而需要一个空元素。空元素的起始标记由一个名字后面有一个正斜杠（/）的标记来表示。注意在标记分隔符的讨论中，我已经提到有时候元素只由一个标记组成。在我们的例子中，第二个人没有电话号码，因而我们放入一个空元素 <TEL/>。

6. 属性

XML元素属性的外观和功能都类似于 HTML标记中包含的属性。它们修改或者进一步定义与它们相关联的标记。它们包含在标记内部，由一个属性名和一个相关联的值组成。我们的resumes文档中的PERSON标记有一个属性，如下例所示：

```
<PERSON PERSONID="p1">
```

属性名称是PERSONID，在这个例子中，其值是p1。属性的值必须放在引号中。

7. 注释

在形式和功能上， XML注释与HTML注释相同。它们都以“ <!-- ”开始，以“ --> ”结束。我们的文档中的第二行

```
<!-- ******* Resumes for Potential Hires ******* -->
```

就是XML注释。

8. 实体引用

在XML文档中，有五个不允许出现在标记之间的字符。这些字符如表 1-2所示。

表1-2 实体替换

字 符	实 体
&（和号）	使用&
'（撇号）	使用'
<（小于号）	使用<
>（大于号）	使用>
“（引号）	使用"

更进一步的讨论将会指出为什么使用以上几个实体会干扰解析。 &符号用于分隔特殊字符，例如表 1-2中的“ 实体 ”一栏中所列出的字符。剩余的几个实体用于分隔标记本身以及属性数据。

9. CDATA

在XML文档中，如果有一段XML文档不希望让XML解析器触动和处理，那么可以将其放入CDATA节中。这些节被当作纯文本来看待。如果不希望文本中的字符，例如前面提到过的实体引用，被当作XML中的特殊字符来解释，那么把它们放入 CDATA节是一个很好的方法。 CDATA节由<![CDATA[和]]来分隔。

注意方括号的位置和顺序。这是惟一的一个在“ ！ ”和关键字之间有方括号的声明。

在下面的例子中，XML解析器将整个实体看作纯文本，而不进行解释：

China-pub.com
下载


```
<![CDATA[Here is a section on CDATA & its syntax with <element> tags]]>
```

10. 名字空间

名字空间是XML最近加进的一项新特性。它不是必需的，但使用它是明智的。它们的主要作用是保证元素的唯一性。考虑以下的情况：我可以在文档中定义一个标记 <NETSTORAGE>用于磁盘存储类型。在另外一个文档中，设计者可能有一个用同样的名字定义含义不同的标记。如果这些文档被合并（很有可能），则这些元素会发生冲突。

下面的一个例子中有一个 XHTML文档，该文档中有标准的 XHTML标记。现在需要将一些自己设计的元素加入到此文档中，但是，一部分自己设计的标记名称与 XHTML标记的名称相同，这肯定会干扰HTML解释器的工作，并且一定会产生预料之外的结果。名字空间分配则可以通过给标记分配一个作用域来避免这种情况的出现。如果这看上去有些难以理解，请记住可以在两个不同的文档中定义两个名字相同而含义不同的标记，而你（或者你所属的公司）合并这两个文档的机率可能相当高。

名字空间定义通常都被指定为标记的一项属性。可以通过将一个分号和需要的名称添加到xmlns声明中来给这些定义在文档中分配一个名称（作用域）。

在我们的例子中，把名字res分配给了名字空间'http://www.myorg.net.restags':

```
<RESUMES xmlns:res='http://www.myorg.net/restags'>
```

在下面的例子中，当使用此声明时，我将对此名称加上所有标记：

```
<RES:PERSON  PERSONID="p1">
    <RES:NAME>
        <RES:LAST>Shelton</LAST>
        <RES:FIRST>Rick</FIRST>
```

再考虑一下XML文档和根元素声明。这个例子说明了该文档的缺省名字空间：

```
<RESUMES smlns='http://www.myorg.net/tags'>
```

文档中所有没有名字空间前缀的标记都被假定属于该文档的名字空间。因此，在下面的例子中，前缀不是必需的：

```
<PERSON PERSONID="p1">
     <:NAME>
         <LAST>Shelton</LAST>
         <FIRST>Rick</FIRST>
```

现在，这看起来可能太多了一些，但是在第 3章讨论样式表转换时，你将会发现这一构造会经常用到。

也有可能在同一个XML文档中有一个以上的名字空间声明：

```
<RESUMES xmlns='http://www.myorg.net'>
      xmlns:phonenumbers='http://www.phones.net'
```

在此声明下，请看代码清单 1-2中的文档片段。

China-pub.com 下载

代码清单1-2 Resumes XML文档片段

```
<ADDRESS>
  <STREET>911 Intranet Ave.</STREET>
  <CITY>Canberra</CITY>
  <COUNTRY>Australia</COUNTRY>
</ADDRESS>
<phonenumbers:TEL>(+612) 111-2345</TEL>
<EMAIL>shelton@somewhere.com</EMAIL>
```

由于第一个名字空间声明没有加有名字，所以它是缺省名字空间，因而文档中除了 <TEL>标记之外的其他标记都不需要名字空间前缀。第二个名字空间是为应用于标记 <TEL>的文档定义的。这并不是什么花招，事实上，<TEL>标记和一个与文档中其他所有标记都不同的、文档的书写者事先知道的名字空间相关联。

1.2.3 格式良好的XML文档

要使XML解析器能够正确处理 XML文档，则文档至少应该是格式良好的（ well-formed）。为了满足这一需求，该文档必须至少遵循以下标准：

- 文档必须有一个与之相关联的 DTD或者有一个XML声明，其standalone属性设置为no。试检查在我们的实例文档中的 XML声明，如代码清单1-1所示。
- 属性必须括在引号中。至于是双引号还是单引号并不重要。请看实例文档中的第一个 <PERSON>元素的属性。
- 除了空元素之外，所有的元素都必须有一个开始标记和一个结束标记。
- 空标记必须有一个结束标记“ / ”。例如，在我们的文档中用于第二个人的空标记 <TEL/>。
- 标记必须被正确地嵌套。以下的标记顺序是不允许使用的： ..... 。
- 标记（markup）不能用作元素文本的一部分。

1.3 文档类型定义

现在我们已经讨论了 XML文档。下面将讨论描述 XML文档结构的实体，此文档就是文档类型定义（Document Type Definition，DTD）。

现在，让我们复习一下 XML处理过程（见图1-6）。

我曾经听说过对 DTD的各种各样的评价，例如从“ 一条走到尽头的路 ”，“ 学习它就是浪费时间 ”等。这些说法基于这样的事实： W3C将要用新的描述 XML文档结构的方法 —— XML模式 —— 来取代DTD（在本章后面的部分，我们将讨论模式）。DTD之所以被取代确实是有其原因的（见本章的1.4节）。

我认为有些人忽视了一点： DTD的存在时间很长，其应用范围很广。如果要取代这些 DTD，首先需要理解它们，然后将其转换为模式文档。而将一种你所不理解的东西转换为另外一种东西通常是困难的。

China-pub.com
下载

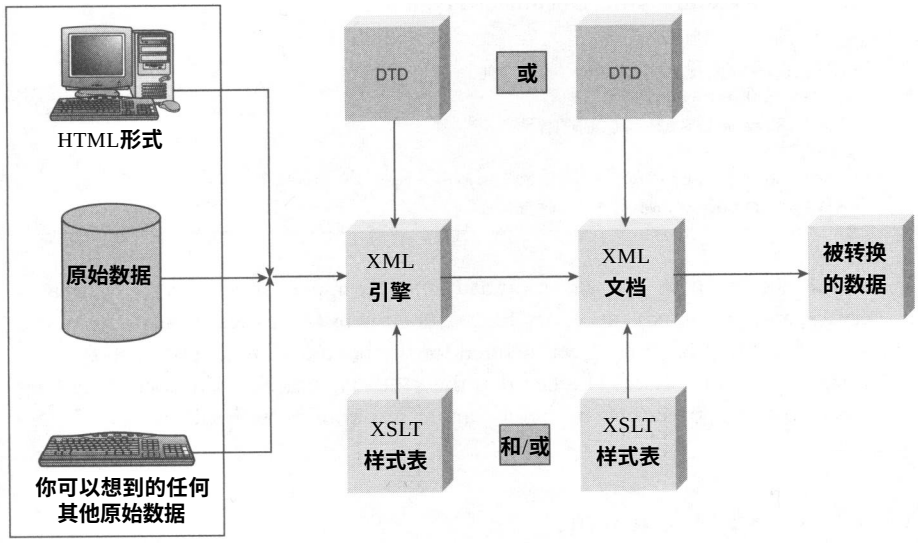

图1-6 XML处理过程中的DTD

什么时候需要使用DTD？对于大的文档管理项目来说DTD是必需的。在大量类型类似的文档的管理中，坚持一个标准是一个严重的问题，因为文档通常都是由不同的作者在不同的时间完成的。不幸的是，在书写自己那部分的时候，作者们都倾向于采用各不相同的风格，结果文档只有一点点或者无任何相似之处，即使它们被认为是相同的。对这一问题的解决方案是建立某种风格指南。这样，DTD就诞生了，它强制作者们遵从一个标准的文档格式。

什么时候不需要使用DTD？对于小文档来说，使用DTD纯属浪费时间。编写DTD是一项困难的工作，正如编写信息系统的过程一样，并不简单。

例如：对于信件、备忘录、传真以及诸如此类的文档来说，使用 DTD是大材小用了。通常没有必要对它们进行严格的限制。但是，有的情况下也有必要对信件使用 DTD（例如，当法律规定要记录信件内容的时候）。

DTD可以定义在XML文档的内部，也可以在内部引用，而在外部访问。此外， DTD并不是XML文档中必需的元素，而是一个可选的实体。需要有效性检查的 XML解析器需要DTD来保证文档具有特定的结构，而不需要有效性检查的XML解析器仍然可以解释该文档的数据。

下面，我们将开始讨论DTD的组成部分以及它们定义XML文档结构的方式。

1.3.1 DOCTYPE声明

DOCTYPE声明并不是DTD声明。它是一条具有自己的语法的 XML指令，它定义了XML文档的根元素。请看以下的例子：

```
<!DOCTYPE RESUMES [the rest of the declarations for this XML document]>
```

这里，叹号（!）位于关键字DOCTYPE的前面，接着是文档的根元素，最后是对 XML文档的所有声明，这些声明位于方括号中。

需要全本请访问 www.ertongbook.com

China-pub.com 下载

1.3.2 Public与Private

DOCTYPE声明有两个附加的属性，这两个属性指定 DTD是公共（public）的还是私有（private）的。

公共的DTD由关键字PUBLIC声明。它是被公共化了的，可广泛使用并且被统一分配一个名称，以保证在每一个实例中都使用同一个 DTD。它也给出了DTD的位置。例如，定义HTML的结构的DTD。按照命名约定，它被称之为正式公共标识符（ formal public identifier），而这超出了本书的讨论范围。本书只讨论XML规范。

私有的DTD使用关键字SYSTEM。它指定一个应用到根元素里面的所有元素的 DTD。这种类型的DTD通常用于有限的范围之内，例如一个经常被你的公司使用，但是不经常被大众或者其他公司使用的XML文档。下面的例子详细说明了前面的例子，它同时给出了声明的两种形式：

```
<!DOCTYPE RESUMES SYSTEM "location of DTD">
<!DOCTYPE RESUMES PUBLIC "name" "location of DTD">
```

需要强调的是：PUBLIC和SYSTEM仅用于外部DTD。如果这些关键字被省去而又需要DTD，那么DTD必需在文档的内部定义。

> 前面提到过XML声明语句（<?xml...?>）及其属性standalone。如果standalone属性值为no，则文档需要DTD。此时，DTD是内部的还是外部的都无关紧要。如果standalone属性的值为yes，则说明不存在外部DTD。

1.3.3 注释

DTD注释在形式和功能上都与 HTML、XML注释相同。它们以 <!--开始，以-->结束。以下是一个DTD注释的例子：

```
<!-- ******* Here are the element descriptions *******-->
```

1.3.4 声明元素

XML文档中的每一个元素都必须在适当的 DTD中加以声明。这些声明必须包含在DOCTYPE声明的方括号内，如下所示：

```
<!DOCTYPE RESUMES [all element declarations here]>
```

在声明元素的时候，使用以下的语法：

```
<!ELEMENT 元素名称 规则>
```

元素名称是在标记分隔符之间的名字，作用于这些名字的规则紧随其后。

1. ANY

这是最简单的元素规则。它规定：在开始标记和结束标记之间，可以出现所有其他标记和字符数据（PCDATA）。它的写法如下：

```
<!ELEMENT RESUMES ANY>
```

China-pub.com
下载

这条规则通常用于根元素的声明中。

考虑一下这条规则，在使用这条规则时，你实际上是在说其他所有的标记和数据。当使用这条规则时，XML解析器将接受在文档根元素与其结束标记之间的所有内容。事实上，这等于是关闭了文档的有效性检查。这样，你一定会觉得自己是世界上最好的 DTD编写者，因为解析器永远都不会报错。但是，这样做的结果往往就是背离了标准却又找不出原因。因而在使用时一定要注意。

2. #PCDATA

有时候，只需要显示字符数据。此时，我们可以使用 #PCDATA规则，其表述形式如下：

```
<!ELEMENT ADDRESS (#PCDATA)>
```

PCDATA代表已解析的字符数据（parsed character data）。在上面的例子中，注意声明被括在圆括弧中。另外还需要注意 PCDATA意味着在被声明的元素的开始标记和结束标记之间不允许出现其他元素，而只能出现字符数据。表 1-3列出了PCDATA示例。

表1-3 PCDATA示例

示 例	有 效 性
<STREET></STREET>	有效
<STREET>911 Intranet Ave.</STREET>	有效
<STREET>911 Intranet Ave.</STREET>	无效。因为只有PCDATA可以出现在<STREET>标记之间

与表1-3列出的情况相对照，有时候需要强制一个元素处于另一个元素之内。例如：

```
<!ELEMENT ADDRESS (STREET)>
<!ELEMENT STREET (#PCDATA)>
```

这些声明规定了元素 ADDRESS必须包含另一元素 STREET。元素STREET只能包含一个字符数据，而且只能出现一次。表 1-4给出了必需的元素示例。

表1-4 必需的元素示例

示 例	有 效 性
<ADDRESS> <STREET>211 Yardwork Circle</STREET>	有效
<ADDRESS> <CITY>Canberra</CITY>	无效。因为必须有一个STREET元素

3. 元素序列

如何在一个元素的内部以要求的顺序定义一个元素序列呢？在圆括号中按顺序定义必要的元素，中间用逗号隔开，如下所示：

```
<!ELEMENT ADDRESS (STREET,CITY,COUNTRY)>
<!ELEMENT STREET (#PCDATA)>
<!ELEMENT CITY (#PCDATA)>
<!ELEMENT COUNTRY (#PCDATA)>
```

China-pub.com 下载

这一声明指定 ADDRESS元素必须按照顺序包括三个另外的元素： STREET、CITY和COUNTRY。这三个另外的元素只能由PCDATA组成，如下所示：

```
<ADDRESS>
  <STREET>911 Intranet Ave.</STREET>
  <CITY>Canberra</CITY>
  <COUNTRY>Australia</COUNTRY>
</ADDRESS>
```

4. 选择元素

如何声明是一个特定的元素还是另一个不同的元素显示在一个给定元素之内（只能二选一）？可以把管道符号（|）放在两个元素之间来进行选择。代码清单 1-3给出了这种声明的一个例子：

代码清单1-3 在元素之间指定一个选择

```
<!ELEMENT ADDRESS (STREET, CITY, COUNTRY, (ZIP|PC))>
<!ELEMENT STREET (#PCDATA)>
<!ELEMENT CITY (#PCDATA)>
<!ELEMENT COUNTRY (#PCDATA)>
<!ELEMENT ZIP (#PCDATA)>
<!ELEMENT PC (#PCDATA)>
```

这里，除了我们已经谈到过的STREET、CITY和COUNTRY之外，还声明一强制性的第五个元素，该元素既可以是美国邮政编码（ ZIP），也可能是邮递区号（ PC）。回顾在1.2.2节中的实例XML文档，你就可以知道第一个工作候选人的地址中有一个邮递区号，而第二个工作候选人的地址中有一个美国邮政编码。

5. 子元素

通常，子元素是包含在其他元素中的元素。第 6章在讨论XPath查询的时候，将会给出更加正式的定义。但是这里的定义对于讨论 DTD来说已经足够了。

现在，对于我感到困惑的三种可能性（下面一段将介绍这三种可能性）来说，每一次我都必须使用其一。如果你熟悉正则表达式的话，那么在这里你就不会有什么问题。

元素或者在括号中的元素组有可能虽然已经声明，但是在某个元素的特定实例中并不出现。另外，一个元素在另一个元素中可能多次出现。还有其他一些可能性，下面将要讨论这些可能性。

请记住：我们已经讨论了单个元素在 PCDATA节中出现的情况。

以下是其他可能性：

- 一次或者多次。当一个元素或者元素组必须出现一次，但是可以出现多次的时候，使用加号（+），如下所示：

```
<!ELEMENT PERSON (TEL)+>
```

这个例子表示PERSON元素必须包括一个电话号码，但是也可以包括多个电话号码。

- 零次或者多次。当一个元素或者元素组可能出现任意多次，但是也可能不出现的时候，使

China-pub.com
下载

用星号（*）来表示，如下：

```
<!ELEMENT PERSON (TEL)*>
```

这里我们声明PERSON元素可以包括一个或多个PHONE元素，也可以一个都不包括。

- 零次或者一次。当一个元素或者元素组出现一次或一次都不出现时，用问号（ ?）来表示：

```
<!ELEMENT PERSON (TEL)?>
```

这里我们声明PERSON元素必须可以包含一个电话号码元素，或者一个电话号码元素都不包含。

- 圆括号。在需要的地方可以自由地使用圆括号来表示元素间的关系。但是要记住，括号位置的微小变化可能对DTD定义造成巨大影响。以下是我们的基础声明：

```
<!ELEMENT ADDRESS (STREET,CITY,COUNTRY,(ZIP|PC))>
```

在上面的声明中，ADDRESS包括了元素STREET、CITY、COUNTRY和ZIP或PC。

而在以下的代码中， ADDRESS包括元素 STREET、CITY和PC或者 STREET、CITY、COUNTRY和PC。

```
<!ELEMENT ADDRESS (STREET,CITY,（COUNTRY,ZIP）|PC)>
```

注意，圆括号和+、*、？的组合增加了声明元素时的灵活性。例如：

```
<!ELEMENT PERSON (NAME,ADDRESS, (TEL|FAX)+,(ZIP|PC)>
```

元素PERSON必须包含元素NAME、ADDRESS，任意数目、任意组合的 TEL或者FAX，以及ZIP或者PC元素。

6. 空元素

这是另外一种简单声明。任何是空元素的元素用关键字 EMPTY来声明：

```
<!ELEMENT7 TEL EMPTY>
```

在此声明中有效的TEL元素可以是<TEL>或者<TEL></TEL>。

7. 实体

在DTD中，实体声明的功能与应用程序中宏的功能非常类似。本质上，它们都是一组字符被另一组字符所替换。实体声明如下所示：

```
<!ENTITY name "substitute characters">
```

引号是必需的。在进行该声明之后，替换字符串将会在实体名出现的位置替换它。让我们看一个例子：

```
<!ENTITY trademark "&#x174">
```

在这一声明之后，可以在如下语句中使用它：

```
<drive>
    Our GadgetDrive&trademark is revolutionary
</drive>
```

其中的&trademark将被替换为™符号。

注意，我们所讨论的是通用实体引用，只在一个 XML文档内起作用。也存在其他的几种类

China-pub.com 下载

型，但这些超出了本书的讨论范围。

1.3.5 声明属性

属性由名字-值对组成。它们出现于开始标记中，为该标记本身或者标记对中的数据提供其他的数据和信息。这里有一个例子，其中的HAIRCOLOR是标记<STEPCHILD>的一个属性。

```
<STEPCHILD HAIRCOLOR="red">John</STEPCHILD>
```

现在来了解一下在DTD中各种不同的属性是如何被处理的。

1. 单一属性

在XML文档中出现的属性必须在相关的DTD中进行声明。可以使用下面的声明来指定这些属性：

```
<！ATTLIST 元素 属性 类型 缺省值>
```

ATTLIST后面的第一个词是包含属性的元素名称，然后是属性名称、属性的数据类型和缺省值。

2. 多个属性

大多数的HTML标记可能有一个以上的属性，例如，<table>标记可以有WIDTH、CELLSPACING、BORDER以及更多的属性。XML元素也不例外，它们也可以有多个属性。在DTD中，有两种定义多个属性的方法。第一种是把每一个属性列为一行，如下所示：

```
<!ATTLIST TRIANGLE BASE CDATA "1">
<!ATTLIST TRIANGLE HEIGHT CDATA "1">
```

第二种方法通过安排适当的空白，使声明对用户来说显得清楚。如下所示：

```
<!ATTLIST TRIANGLE BASE CDATA "1"
                   HEIGHT CDATA "1">
```

在这两种方法中，你可以选择自己喜欢的一种方法。第二种方法确实与顺序相关，而且肯定会碰到这样书写的DTD。但是，你必须决定用何种方式来进行书写。

3. 缺省值

指定一个缺省值通常不是必需的。DTD规范中包含了三个关键字，这三个关键字为缺省值的使用提供了回旋余地，如下所示：

- #REQUIRED。在我们的XML文档中，假设一个元素有一个我们需要强制作者来对其填充数据的属性（例如一个作者的email地址，以便同作者联系），如下所示：

```
<!ELEMENT AUTHOR EMPTY>
<!ATTLIST AUTHOR EMAIL CDATA #REQUIRED>
```

这将会要求作者提供其email地址。

- #IMPLIED。IMPLIED关键字指定，如果该属性不在元素中出现，就没有什么问题；忽略而已它就可以了。然而，它是必要的，因为XML解析器将解析此文档，并通知调用它的应用程序此处无可用的值。然后，由应用程序决定如何处理。考虑一个被通知该属性无可用值的SQL应用程序。如果该属性映射到SQL数据库中的一列，则应用程序会给该列填写一

China-pub.com
下载

个空值，如下所示：

```
<!ELEMENT AUTHOR EMPTY>
<!ATTLIST AUTHOR TEL CDATA #IMPLIED>
```

如果AUTHOR元素的TEL属性不存在，则忽略它。

- #FIXED。FIXED声明由作者提供的缺省值不可改变。下面的例子说明作者的 LASTNAME 属性的缺省值被设置为“ Griffin ”，而且不可改变：

```
<!ELEMENT AUTHOR EMPTY>
<!ATTLIST AUTHOR LASTNAME CDATA #FIXED "Griffin">
```

实际上，这是一个只读的值。

4. 数据类型

有10种数据类型用于属性值，如表 1-5所示。

表1-5 属性数据类型

数据类型	描 述
CDATA	字符数据
枚举	一系列值，只能从中选择一个
ENTITY	在DTD中声明的一个实体
多实体ENTITIES	DTD中声明的、由空白隔开的多个实体
ID	唯一的元素标识符
IDREF	一个唯一的ID类型属性值
IDREFS	属于元素的由空白隔开的多个IDRREF
NMTOKEN	声明一个合法的XML名字
NMTOKENS	多个由空白隔开的XML名字标记
NOTATION	DTD中声明的符号

90%的DTD属性数据类型都是由 CDATA类型或者枚举类型组成的。这些类型在后面做了详细介绍。对于其他8种数据类型的讨论超出了本书的范围。

下面讨论CDATA和枚举两种类型：

- CDATA。这是属性中最一般的类型。这种类型的属性可以是任意的字符串值，只要它们中不包括小于号（<）或双引号（"）。如果属性中必须包括这两种字符，则必须使用标准实体引用（&lt和&quot）。如果属性值只包括一个引号，那么可以将它括在单引号中，以避免使用实体。下面的例子把值7英寸（"）赋给了BASE属性。

```
<!ATTLIST TRIANGLE BASE CDATA ‘ 7" ’ >
```

如果属性值同时包括单引号和双引号，则没有分隔该值的引号必须改成实体值。

- 枚举类型。前面讨论过元素声明，并了解到两个元素中选一个的声明可以出现在文档中。回忆下面的例子：

```
<!ELEMENT ADDRESS (STREET,CITY,COUNTRY,(ZIP|PC))>
```

这表示第四个元素既可以是美国邮政编码也可以是邮递区号。在属性值声明中，也可以做类似的工作。例如：

需要全本请访问 www.ertongbook.com