第3章 XML有效性与模式

本章学习要点

本章主要介绍XML模式信息的基础知识，说明模式的概念及其作用，着重介绍了W3C推荐的两种XML模式语言——DTD(文档类型定义)和XML Schema，并分析了它们各自的优劣及发展趋势。通过本章内容的学习，读者应该掌握以下主要内容：

- XML模式信息的意义
- DTD的创建和使用方法
- XML Schema的创建和使用方法
- DTD和XML Schema的区别与各自的适用范围

3.1 模式信息的基本概念

第2章介绍了什么是“形式良好”的XML文档。“形式良好”是对XML文档的基本要求，使得XML文档结构清晰、完整，便于处理程序对其进行解析，从而简化处理程序的编写工作，并加快浏览速度，减少浏览所需占用的空间。然而，当撰写了一个符合XML 1.0“形式良好”的文档后，还会出现这样一个问题：将如何与其他应用程序交流所设计的文档及其结构？比如要写一个说明图书信息的XML文档，可以这样写：

```
<?xml version = "1.0" encoding = "GB2312" ?>
  <Books>
      <book>
      <title>数据库系统概论</title>
      <ISBN>7040195836</ISBN>
      <authors>
          <author>王珊</author>
          <author>萨师煊</author>
      </authors>
      <publisher>高等教育出版社</publisher>
      <price>33.80</price>
      </book>
      <book>
          <title>计算机网络</title>
      <ISBN>7505387863</ISBN>
      <authors>
          <author>谢希仁</author>
      </authors>
      <publisher>电子工业出版社</publisher>
```

```
        <price>35.00</price>
        </book>
    </Book>
```

也可以这么写：

```
<?xml version = "1.0" encoding = "GB2312" >
<图书系列>
      <图书>
          <书名>数据库系统概论</书名>
          <书号>7040195836</书号>
          <作者 人数 = "2">王珊,萨师煊</作者>
          <出版社>高等教育出版社</出版社>
          <价格>33.80</价格>
      </图书>
      <图书>
          <书名>计算机网络</书名>
          <书号>7505387863</书号>
          <作者 人数 = "1">谢希仁</作者>
          <出版社>电子工业出版社</出版社>
          <价格>35.00</价格>
      </图书>
</图书系列>
```

以上两个文档对于人来说是很容易看懂的，都是从书名、ISBN 号、作者、出版社和价格几方面描述图书，但对于计算机来说，就无法判断

```
<authors>
   <author>王珊</author>
   <author>萨师煊</author>
</authors>
```

和

```
<作者 人数 = '2'>王珊,萨师煊</作者>
```

这两个元素是不是一码事了。

我们知道，XML 实质上给文档开发者提供了一种基于信息描述的、能够体现信息之间逻辑关系的、可以确保文件的易读性和易搜索性的自定义标记，为了使其他用户以及程序能够正确地识别、比较和处理创建的自定义 XML 词汇表，需要某种通用的方式来说明或者约束 XML 文档的结构和文档的语法，正如没有规矩难以成方圆一样，这就是 XML 模式信息所要完成的功能。

XML 模式信息是一种描述信息结构的模型，用于定义 XML 文档的语法和词汇表，即规定文档的整体结构以及语法规则。这个“规则”可以非常简单，仅仅列出 XML 文档中需要包含的有效元素，例如元素、标记、属性、实体；也可以非常复杂，不但列出这些元素，还规定这些元素之间的内在联系，例如说明元素 A 中必须包含元素 B 或元素 C，但不能同时包含两个元素等。一个完全意义上的 XML 文档不仅应该是“形式良好的”，而且还应该是符合模式规则约束的，这样的文档我们称之为“有效的”XML 文档。简单地讲，XML 模式信息规定了一个语法分析器为了解释一个“有效的”XML 文档所需要知道的所有规则的细

节。XML 文档可以根据模式信息进行比较，这一过程称为“验证”。如果该文档与模式信息列出的约束规则相匹配，则认为该文档是有效的；如果该文档与约束规则不匹配，则认为该文档无效。

目前最常用的两种撰写 XML 模式信息的语言是由 W3C 推荐的 XML DTD (Document Type Definition，文档类型定义)和 XML Schema。在使用 XML 时，通常可以用已有的 DTD，如 MathML(在数学领域中使用的标记语言)、CML(在化学领域中使用的标记语言)，或是自行设计 DTD，以外接或内嵌方式将 DTD 与 XML 文档连接起来。由于 DTD 沿用 SGML 的模式机制，有自己特殊的 EBNF 语法，需要专用的解析器，且存在提供数据类型有限及不支持名称空间等缺陷，目前 W3C 正在研发 XML Schema 标准，在不久的将来可能会逐渐取代 DTD，并提供更多的功能。XML Schema 标准是一种描述信息结构的模型，作为 XML 的模式语言，用来定义 XML 文档的文本结构和数据类型等 XML 文档描述规则，且规范了文档中的标记和文本可能的组合形式。它不仅包括了 DTD 能实现的所有功能，而且它本身就是规范的 XML 文档。最重要的是，它能弥补 DTD 的不足，提供一系列新特色，本章将分别对 DTD 和 XML Schema 作详细的介绍。

3.2 DTD(Document Type Definition)

3.2.1 DTD 初步

DTD(Document Type Definition，文档类型定义)是 XML1.0 规范的一部分，它基于 EBNF(Extended Backus-Naur Form)语法，用于对 XML 文档进行“有效性”约束和验证，可以通过比较一个 XML 文档和相应 DTD 文件中的规则来判断该文档是否符合规范，元素标记的使用是否正确。DTD 文档是一个 ASCII 的文本文件，后缀名为.dtd。不同的应用程序只需定义标准的 DTD，各程序就能够依照 DTD 来建立、验证、交换并处理 XML 文档了。

通常，一个 DTD 文档包括：

- 元素的声明。
- 元素间关系的声明。
- 元素可使用的属性声明。
- 可使用的实体或符号声明。

在 3.1 节中，关于图书的两份 XML 文档是“形式良好”的，接下来就以它们为例，分别编写相应的 DTD，初步认识 DTD 的基本结构。

清单 3-1

```
<?xml version = "1.0" encoding = "GB2312" standalone = "yes"?>
<!-- DTD 文档开始 -->
<!DOCTYPE Books [
<!ELEMENT Books (book * ) >
<!-- 元素及其内容模型的声明 -->
<!ELEMENT book (title, ISBN, authors, publisher, price)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT authors (author * )>
```

```
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT author (#PCDATA )>
]>
<!-- DTD 文档结束 -->
<Books>
    <book>
        <title>数据库系统概论</title>
        <ISBN>7040195836</ISBN>
        <authors>
            <author>王珊</author>
            <author>萨师煊</author>
        </authors>
        <publisher>高等教育出版社</publisher>
        <price>33.80</price>
    </book>
    <book>
        <title>计算机网络</title>
        <ISBN>7505387863</ISBN>
        <authors>
            <author>谢希仁</author>
        </authors>
        <publisher>电子工业出版社</publisher>
        <price>35.00</price>
      </book>
</Books>
```

清单 3-2

```
<?xml version = "1.0" encoding = "GB2312" ?>
<!-- DTD 文档开始 -->
<!ELEMENT 图书系列 (图书 * )>
<!ELEMENT 图书 (书名, 书号, 作者, 出版社, 价格)>
<!ELEMENT 书名 (#PCDATA)>
<!ELEMENT 书号 (#PCDATA)>
<!ELEMENT 作者 (#PCDATA)>
<!ELEMENT 出版社 (#PCDATA)>
<!ELEMENT 价格 (#PCDATA)>
<!-- 元素的属性列表声明 -->
<!ATTLIST 作者
          人数 CDATA "1">
<!-- DTD 文档结束 -->
```

上例中,ELEMENT 关键字用来声明元素,后面是该元素的具体内容,而 ATTLIST 关键字用来声明元素所包含的属性。下面具体介绍 DTD 的定义方法。

3.2.2 关联 DTD 与 XML 文档

DTD 可以直接包含在使用它的 XML 文档中,也可以是一个完全独立的.dtd 文件,通过外部的 URL 与 XML 文档相关联,这些外部 DTD 可以被不同文档和机构所共享。

据此,DTD 分为内部 DTD 和外部 DTD 两种。

3.2.2.1 内部 DTD

关联 DTD 与 XML 文档的一个最简单方法是，在 XML 文档内部的序言部分加入一个 DTD 描述，加入的位置是紧接在 XML 处理指令之后。一个包含 DTD 的 XML 文件的结构为：

```
<?xml version = "1.0" encoding = "GB2312" standalone = "yes"?>
<!DOCTYPE 根元素名[…
]>
XML 文档主体部分…
```

其中：

- <!DOCTYPE：表示该 XML 文档所设定的 DTD 从这里开始，注意 DOCTYPE 是大写，这条指令是文档类型声明指令。
- 根元素名：指定此 DTD 的根元素的名称，一个 XML 文件只能有一个根元素。注意，如果 XML 文件使用了 DTD，那么文件中的根元素就在这里指定。
- […]>：在“[ ]”标记里面定义 XML 文档所能使用的元素、属性和实体等规则，然后用“>”结束 DTD 的定义。

这样，我们就定义了一个 XML 文档，它以 DOCTYPE 中规定的根元素名作为其根元素。清单 3-1 就是一个使用内部 DTD 的 XML 文档实例，其浏览器中的显示结果如图 3-1 所示。图中，DTD 的具体内容并不显示出来，除了有一行文档类型声明指令外，它和普通 XML 文档的显示并无差别。

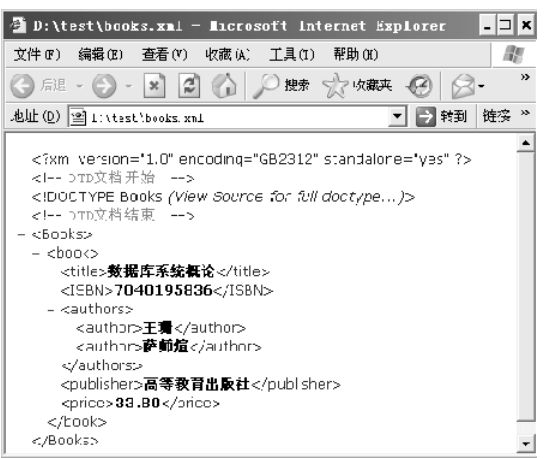

图 3-1 引用内部 DTD 的 XML 文档

3.2.2.2 外部 DTD

如果为每一个 XML 文件加入一段 DTD 定义，是相当繁琐的。更多的情况下，我们会为一批 XML 文件定义一个相同的 DTD。比如，有几十家相互联系的、具有合作伙伴关系的公司、厂商，他们相互之间的交换电子文档都是用 XML 文档，有相同的格式，如果为每一篇单独定义既麻烦，又不利于统一格式，我们可以考虑将这些 XML 文档的 DTD 放在某个地方，让所有交换的 XML 文档都使用此 DTD，这就是外部 DTD。外部 DTD 还保证当需要

对 DTD 做出改动时，不用逐一修改每个引用了它的 XML 文档，只需更改一个公用的 DTD 文件即可。

外部 DTD 的创建方式及语法和内部 DTD 一样，把清单 3-2 存为后缀名为.dtd 的文件就产生了一个外部 DTD。值得注意的是，外部 DTD 文件中不包含<!DOCTYPE…语句声明。

为了引用一个外部 DTD，必须将 XML 文档的声明部分作如下修改：

首先，在 XML 声明中必须说明这个文件不是自成一体的，即 standalone 属性的属性值不再是“yes”，而是“no”：

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

另外，应添加文档类型声明表示引用了一个外部 DTD：

```
<!DOCTYPE 根元素名 SYSTEM "外部 DTD 文件的 URL">
```

或

```
<!DOCTYPE 根元素名 PUBLIC "DTD 名称" "外部 DTD 文件的 URL">
```

其中：

- <!DOCTYPE：表示开始声明引用外部 DTD。
- 根元素名：是指该 DTD 的根元素的名称。
- SYSTEM 关键字：是指该外部 DTD 文件是私有的，即个人创建的，没有公开发行，只是个人或在组织内部或者几个合作机构之间使用。
- PUBLIC 关键字：是指该外部 DTD 是一个由权威机构制定的，提供给特定行业或公众使用的 DTD，用 PUBLIC 的 DTD 都有一个逻辑名称——DTD 名称，我们必须在调用时指明这个逻辑名称。
- DTD 名称：DTD 名称通常由 4 个部分组成：第一部分表明 DTD 性质，即 ISO 标准的 DTD 以“ISO”三个字母开头；被改进的非 ISO 标准的 DTD 以“+”开头；未被改进的非 ISO 标准的 DTD 以“-”开头。第二部分表明 DTD 所有者的名称。第三部分指示 DTD 所描述的 XML 文档的类型及主要内容。第四部分表明所使用语言的种类(英文 EN，中文 ZH，法文 FR 等，具体可参见 ISO 639)。这 4 个部分之间必须用双斜杠“//”分隔，而且前后顺序固定。例如，<!DOCTYPE 图书系列 PUBLIC "-//WHPU//Book Data//ZH" "http://www.wupu.edu.cn/book.dtd">，对于 DTD 的命名通常不是其引用者的任务，XML 文档的编写者只要在自己的文件中把事先定义好的 DTD 名称放在相应的位置中就可以了。
- 外部 DTD 文件的 URL：是用 URL 的方式指明外部 DTD 文件的位置。

我们将清单 3-2 的这份 DTD 文件存放在 URL 为 http://www.whpu.edu.cn/的地方，文件名为 book.dtd。那么引用这一 DTD 文件的 XML 文档如下：

清单 3-3

```
<?xml version = "1.0" encoding = "GB2312" standalone = "no" >?
<!DOCTYPE 图书系列 SYSTEM http://www.xml.com/book.dtd>
<图书系列>
    <图书>
```

```
        <书名>数据库系统概论</书名>
        <书号>7040195836</书号>
        <作者 人数 = "2">王珊,萨师煊</作者>
        <出版社>高等教育出版社</出版社>
        <价格>33.80</价格>
    </图书>
    <图书>
        <书名>计算机网络</书名>
        <书号>7505387863</书号>
        <作者 人数 = "1">谢希仁</作者>
        <出版社>电子工业出版社</出版社>
        <价格>35.00</价格>
    </图书>
</图书系列>
```

上面的外部 DTD 文件的 URL 是一个绝对路径,除此以外,它还可以是一个相对路径,如:

```
<!DOCTYPE 图书系列 SYSTEM "book.dtd">
```

表示这个 DTD 文件和引用它的 XML 文件在同一个目录下。显然,使用外部 DTD,还可以方便地把 DTD 文件从 XML 文档中分离出来,使用到另外的文件。

此外,在一个 XML 文档中可以引用不同的外部 DTD 文件。同时,在引用外部 DTD 时,还可以在 XML 文档中自定义内部的 DTD。但需要注意的是,不能对某一元素进行重复定义。

3.2.3 声明元素

DTD 需要对 XML 文档中所能出现的全部元素进行声明,规定元素中的每一个细节。本节将介绍如何声明元素,包括元素的名称、元素的内容模型。

3.2.3.1 声明元素的语法

声明元素的语法为:

```
<!ELEMENT 元素名 元素内容描述>
```

其中:

- <!ELEMENT:表示元素声明指令开始,ELEMENT 是 XML 的关键字,不能更改,也不可以改成小写。
- 元素名:表示要声明的元素的名称。
- 元素内容描述:指明此元素能包含什么样的内容,元素的类型如何,该类型的内容模型如何等。XML 标准将元素按内容划分为 5 种类型。

3.2.3.2 元素的类型

1. #PCDATA 类型

#PCDATA 代表字符数据,是最为基本的元素类型,也就是不包含其他子元素的元素。声明一个基本元素类型的语法为:

```
<!ELEMENT 元素名 (#PCDATA)>
```

例如，清单 3-1 中的<!ELEMENT title (#PCDATA)>表示声明了一个元素 title，它的内容为代表书名的字符数据，不能包含其他元素。对于这个声明，以下元素都是有效的：

```
<title>数据库系统概论</title>
<title>计算机网络</title>
```

以下元素则无效：

```
<title>
      <subtitle>数据库系统概论</subtitle>
   </title>
```

2. EMPTY 元素类型

如果一个元素不包含任何内容，则声明它为空类型。声明语法为：

```
<!ELEMENT 元素名 EMPTY>
```

XML 文档中则以<元素名/>来表示空类型。例如，声明了<!ELEMENT test EMPTY>，则 XML 文档中必须在相应的地方出现一个空元素：<test/>。

3. ANY 元素类型

ANY 代表自由格式。该元素可以包含任何内容，一般只把文档的根元素规定为 ANY 类型，声明语法为：

```
<!ELEMENT 元素名 ANY>
```

值得一提的是，将元素规定为 ANY 类型之后，元素的内容可以是任意数量和顺序的子元素，也可以是可解析的字符数据。这是对元素内容最宽松的限定，实际对元素内容几乎没有任何要求，除非文档明确要求这样的元素，否则应尽量避免使用该元素类型的声明。

4. 子元素类型

这类元素中只能包含指定的子元素，在 DTD 中通过正规表达式规定子元素出现的顺序和次数。

- 如果要求子元素按照规定的次序依次出现，则可声明为：

```
<!ELEMENT 元素名 (子元素名, 子元素名, …) >
```

其中，由圆括号括起来的部分即为元素所包含的子元素列表，子元素之间用逗号隔开。使用这种元素类型声明后，元素包含哪些子元素，各子元素出现的位置和次数都有明确的规定，在具体的 XML 文档中，必须严格执行，否则不能通过有效性验证。

例如：声明<!ELEMENT 图书 (书名,作者,作者,价格) >，则以下元素有效：

```
<图书>
    <书名>数据库系统概论</书名>
    <作者>王珊</作者>
    <作者>萨师煊</作者>
    <价格>33.80</价格>
</图书>
```

而以下将价格和书名两个子元素颠倒则无效：

```
<图书>
    <价格>33.80</价格>
    <作者>王珊</作者>
    <作者>萨师煊</作者>
    <书名>数据库系统概论</书名>
</图书>
```

- 有些情况下，需要在多个子元素中选择一个使用，这种选择性子元素类型可以这样声明：

```
<!ELEMENT 元素名（子元素名 | 子元素名 | …）>
```

其中，由圆括号括起来的部分即为元素所包含的子元素列表，子元素之间用竖线隔开，表示元素必须在子元素列表中选择一个使用，而且只能选择一个。例如，标识一本图书既可以使用书名也可以使用 ISBN 书号，两者选其一即可。见清单 3-4。

清单 3-4

```
<?xml version = "1.0" encoding = "GB2312" standalone = :"yes" ?>
<!DOCTYPE 图书 [
<!ELEMENT 图书 (图书代号, 价格)>
<!ELEMENT 图书代号 (书名 | 书号)>
<!ELEMENT 书名 (#PCDATA)>
<!ELEMENT 书号 (#PCDATA)>
<!ELEMENT 价格 (#PCDATA)>
]>    <书号>7040195836</书号>
<图书>
    <图书代号>
        <书名>计算机网络</书名>
    </图书代号>
    <价格>35.00</价格>
</图书>
<图书>
    <图书代号>
        <书号>7040195836</书号>
    </图书代号>
    <价格>33.80</价格>
</图书>
```

控制元素出现的次数，有以下方法可供选择：

- 规定一个元素可能出现 0 次或 1 次，则在元素后加一个“?”符号。
- 规定一个元素可能出现 0 次或任意多次，即不限定次数，则在元素后加一个“*”符号。
- 规定一个元素可能出现 1 次或任意多次，则在元素后加一个“+”符号。
- 明确规定一个元素出现的次数，如 2 次、6 次，就在子元素列表的相应位置将该元素重复几次。

清单 3-5 是对如何控制元素出现次数的详细举例。

清单 3-5

```
<?xml version = "1.0" encoding = "GB2312" standalone = "yes" ?>
<!DOCTYPE 图书系列 [
<!ELEMENT 图书系列 (图书 * )>
<!ELEMENT 图书 (书名, 书号?, 作者 + , 出版社, 价格)>
<!ELEMENT 书名 (#PCDATA)>
<!ELEMENT 书号 (#PCDATA)>
<!ELEMENT 作者 (#PCDATA)>
<!ELEMENT 出版社 (#PCDATA)>
<!ELEMENT 价格 (#PCDATA)>
]>
<图书系列>
    <图书>
        <书名>数据库系统概论</书名>
        <书号>7040195836</书号>
        <作者>王珊</作者>
        <作者>萨师煊</作者>
        <出版社>高等教育出版社</出版社>
        <价格>33.80</价格>
    </图书>
    <图书>
        <书名>计算机网络</书名>
        <作者>谢希仁</作者>
        <出版社>电子工业出版社</出版社>
        <价格>35.00</价格>
    </图书>
</图书系列>
```

清单 3-5 中,DTD 规定 XML 文档的根元素为“图书系列”,其下有 0 个或多个“图书”元素,“图书”元素又包含 5 个元素,它们是“书名”、“书号”、“作者”、“出版社”和“价格”,其中,“书号”元素后的“?”表示该元素可以出现也可以不出现,“作者”元素后的“+”表示作者至少一个,也可以多个。紧接着的 XML 文档中,有两个“图书”元素,分别描述了数据库系统概论和计算机网络两本图书,后者没有“书号”元素,前者作者有两位,而后者仅一位。

5. 混合元素类型

如果一个元素中可以包含字符数据或指定子元素,或者是同时包含两者,则这类元素为混合类型,混合元素类型的声明语法为:

```
<!ELEMENT 元素名 (#PCDATA | 子元素名 | 子元素名 |…) * >
```

两点说明:

第一,混合元素类型只适用在不限次数的标签中,即声明中圆括号后必须有“ * ”号。

第二,混合元素类型只能用在选择性子元素类型中,即圆括号内由“|”号隔开各项内容。

清单 3-6 是对混合元素类型声明的举例。

清单 3-6

```
<?xml version = "1.0" encoding = "GB2312" standalone = "yes" ?>
<!DOCTYPE 图书系列 [
```

```
<!ELEMENT 图书系列 (图书 * )>
<!ELEMENT 图书 ( #PCDATA | 书名 | 书号 | 作者 | 出版社 | 价格) * >
<!ELEMENT 书名 ( #PCDATA)>
<!ELEMENT 书号 ( #PCDATA)>
<!ELEMENT 作者 ( #PCDATA)>
<!ELEMENT 出版社 ( #PCDATA)>
<!ELEMENT 价格 ( #PCDATA)>
]>
<图书系列>
      <图书>
            <书名>数据库系统概论</书名>
            <书号>7040195836</书号>
            <作者>王珊</作者>
            <作者>萨师煊</作者>
            <出版社>高等教育出版社</出版社>
            该教材为面向 21 世纪课程教材
      </图书>
      <图书>
            <书名>计算机网络</书名>
            <作者>谢希仁</作者>
            <出版社>电子工业出版社</出版社>
            该教材为高等学校电子信息类规划教材
            <价格>35.00</价格>
      </图书>
</图书系列>
```

清单 3-6 中,<!ELEMENT 图书 (#PCDATA | 书名 | 书号 | 作者 | 出版社 | 价格) * > 声明了“图书”元素为混合类型,则图书元素既可以包含字符数据,也可以任意包含其他 5 个子元素。由于混合类型容易打乱应有的层次关系,一般不主张使用混合元素。

以上是 DTD 中可以声明的 5 种元素类型,我们还可以使用分组符号“()”将部分子元素组合为一个元素组,对待元素组和对待普通元素一样,在元素组内部,元素按照规定的方式出现,而且可以配合使用控制出现次数的“?”、“ * ”和“+”来实现更复杂的元素内容设定。

例如,声明<!ELEMENT 图书 (书名,(作者, 作者职称, 作者 E-mail)+,出版社, 价格)>,则以下元素为有效:

```
<图书>
        <书名>数据库系统概论</书名>
        <作者>王珊</作者>
        <作者职称>教授</作者职称>
        <作者 E - mail>example1@yahoo.com</作者 E - mail>
        <作者>萨师煊</作者>
        <作者职称>教授</作者职称>
        <作者 E - mail>example2@yahoo.com</作者 E - mail>
        <出版社>电子工业出版社</出版社>
        <价格>33.80</价格>
  </图书>
```

3.2.4 声明元素的属性

XML 元素的属性是对元素的补充和修饰,通过属性,可以将一些简单的特性与元素相