第3章 文档类型定义

内容导读

XML 文档用于描述数据内容，并进行数据交流。如果使用 XML 文档在不同的应用程序间进行交互，则应保持文档一致性，也就是要求该文档是有效的。有效的 XML 文档要符合文档类型定义（Document Type Definition，DTD），根据 DTD 设计的文档，严格规范了 XML 的语法格式，能够保证数据正常地交流和共享。

本章主要围绕 DTD 的基本结构及语法格式进行详细介绍，并结合实例描述如何根据 DTD 进行 XML 文档设计。

本章要点

- ◇ 理解 DTD 的概念及文档基本结构。
- ◇ 掌握 DTD 中元素声明。
- ◇ 掌握 DTD 中属性声明。
- ◇ 掌握 DTD 在 XML 文档中的引用方式。
- ◇ 掌握 DTD 中实体的使用。

3.1 DTD 概述

3.1.1 DTD 简介

在信息的高速交流中，不同领域之间的信息交换越来越紧密，如何才能保证这些不同领域之间的信息可以更容易且更有效率地交换成为用户首要关注的问题。在实际的应用中，程序员经常会编写内容相似的 XML 文档，如描述个人简历信息、商品订单信息、产品说明等文档，这些文档可能不仅用于显示，还可能给应用程序提供数据信息或用于数据交流。然而 XML 作为一种元标记语言，不同开发团队开发的 XML 格式也不尽相同，如果没有一定的规范来约束 XML 文档，则会影响数据的共享。因此，在保证 XML 文档格式良好的同时，还需要使用 DTD 对文档进行有效性验证，以保证文档的结构和数据类型也符合相关应用程序的要求。

在如今的电子商务中，XML 作为一种经常使用的数据交流媒介，可以在不同企业的信息中进行交互，如果不同的厂商使用不同的标记描述具有相同意义的产品信息，必然导致数据无法进行共享与交互。就如同两个不同国家的人想要进行交流，如果各自使用自己本国

的语言沟通，必然会发生交流障碍。为了解决这个问题，就需要不同的领域来针对该领域的特性制定共同的信息内容模型，然后再通过这个共同的内容模型来标识信息。而 DTD 就是用来规范 XML 文档的一种内容模型。

DTD 是有效的 XML 文档的基础，主要用于规范和约定 XML 文档，目的就是让符合规范的 XML 文档成为数据交换的标准。由于 XML 文本通过树状结构来组织数据，因此在 DTD 中，需要根据 XML 文档结构，具体规定引用该 DTD 的 XML 文档可以使用哪些元素、元素之间如何进行嵌套、各个元素出现的先后顺序有何要求、元素中可包含的属性、属性值的数据类型、可使用的实体及符号规则各有什么特点等。这样不同的公司或团体只要根据具体的 DTD 建立相应的 XML 文档，就可以方便地使用 XML 文档进行数据交互。

DTD 可以看作一个或者多个 XML 文件的模板，每个 XML 文件中的元素、元素的属性、元素的排列方式、元素包含的内容、实体等相关内容，都可以通过模板进行创建。由于各行各业有自己的相关规范及要求，因此 DTD 可以根据不同行业的特点，建立相关的领域中整体性高、适应性广的规范文档，方便相关行业统一文档格式，以更好地进行数据共享和交互。

根据各个行业的要求，目前 DTD 在电子商务、医学、工商、建筑等领域已形成统一规范的文档。例如，在电子商务对应的 DTD 中，包含了商品名称、商品编号、商品信息、订单编号、订单信息等项目，这样任何使用以 XML 为基础文件的相关电子商务机构，都可以方便地读取该文件中的信息。DTD 的相关规范，也为 XML 文档在网络中合理、正确地传输起到基础作用。也就是说，只有严格按照 DTD 规范设计的文档，才能通过 XML 解析器的正确解析，以保证文档结构与数据的有效性。

因此，DTD 是 XML 中的一个重要技术之一。使用 DTD 规范的 XML 文档，在网络中才具有实用性。使用 DTD 的主要作用主要包括以下几个方面。

① 每个 XML 文档均可携带一个有关其自身格式的描述，以验证数据的有效性。

② 独立的团队可以共同使用某个已制定好标准的 DTD 来交换和共享数据。

③ 应用程序可以使用某个标准的 DTD 来验证从外部接收到的数据信息。

④ 用户根据 DTD 就能够获知对应 XML 文档的逻辑结构。

使用 DTD 设计的 XML 文档，可以为 XML 提供统一的设计模式，方便程序设计人员能够不依赖具体数据，即可获知 XML 文档的逻辑结构及相关元素、属性、实体等对应的内容。使用 DTD 设计的文档能够规范统一，在保证有效性的同时，更便于数据在网络中的传输与共享。

3.1.2　DTD 基本结构

DTD 的语法格式与 XML 语法不同，其原因是 DTD 最初是为 SGML 设计的，对于 DTD 文件而言，早在 XML 出现之前就已存在，每一份 SGML 文件均应有相应的 DTD。但是就目前而言，DTD 仍然是约束和规范 XML 文档的一种常用模式。

DTD 的基本结构包括 XML 文档元素的声明、元素间相互关系、属性列表声明以及实体的使用等。为了更好地理解 DTD 的结构，下面通过一个实例来说明 DTD 文档的基本结构。

文件 3-1-2-1.xml

```
<?xml version = "1.0" encoding = "gb2312"?>
<!-- 会员信息相关文档 -->
<!DOCTYPE 会员信息 [
```

```
<!ELEMENT 会员信息 (会员 * )>
<!ELEMENT 会员 (姓名,性别,生日,家庭住址,联系方式)>
<!ELEMENT 姓名 (#PCDATA)>
<!ELEMENT 性别 (#PCDATA)>
<!ELEMENT 生日 (#PCDATA)>
<!ELEMENT 家庭住址 (#PCDATA)>
<!ELEMENT 联系方式 (#PCDATA)>
<!ATTLIST 会员 卡号 ID #REQUIRED>
]>
<会员信息>
    <会员 卡号 = "SY102030">
        <姓名>张红</姓名>
        <性别>女</性别>
        <生日>1986 - 02 - 15 </生日>
        <家庭住址>金地滨河小区 2 号楼 203 室</家庭住址>
         <联系方式> 23232233 </联系方式>
</会员>
<会员 卡号 = "SY102031">
        <姓名>赵楠</姓名>
        <性别>女</性别>
        <生日>1988 - 11 - 20 </生日>
        <家庭住址>万科新里程 5 号楼 1103 室</家庭住址>
        <联系方式> 56561234 </联系方式>
    </会员>
</会员信息>
```

文档说明如下。

① 文档第 1 行是 XML 声明语句。

② 文档第 2 行为注释语句。

③ 文档“<! DOCTYPE 会员信息 [”至“]>”区域为 DTD 声明语句。

④ DTD 声明语句下方是对应有效的 XML 文档。

将该文档保存在 Altova XMLSpay 环境中,使用浏览器查看,程序运行后显示结果如图 3-1 所示。

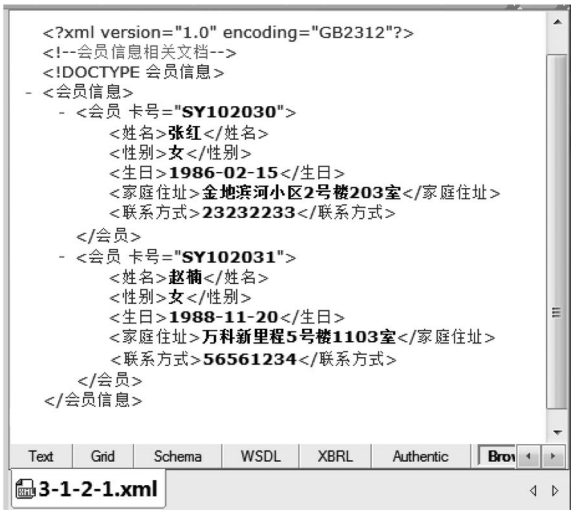

图 3-1　程序运行显示结果

3.2　DTD 中元素的声明

3.2.1　DTD 声明语句

DTD 有自己的语法规则。在 DTD 中，需要使用专用的关键字以告知解析器该区域之间的数据为 DTD 声明，其声明语法格式如下：

```
<!DOCTYPE 根元素 [
    <!-- DTD 定义内容 -->
]>
```

语法格式说明如下。

① ＜！ DOCTYPE：表示 DTD 声明部分开始。

② 根元素：由于 XML 采用树状结构，有且只有一个根，因此根元素表示与 XML 文档中对应的根元素的名称。

③ “DTD 定义内容”：包括 DTD 元素声明、DTD 属性声明和 DTD 实体声明。

④]＞：表示 DTD 声明部分结束。

3.2.2　元素声明的语法格式

元素是 XML 文档的基本组成部分，用来存放和组织数据。在 DTD 中规定了 XML 文档可使用哪些元素、元素间嵌套关系以及各个元素出现的先后顺序等。

XML 采用树状结构排列数据，根据它的结构类型，可以将元素分为叶子元素和枝干元素。所谓叶子元素表示元素中不包含任何子元素，该类元素为末端节点。枝干元素表示元素包含一个或多个子元素。例如，文件 3-1-2-1.xml，其对应元素的树状结构如图 3-2 所示。

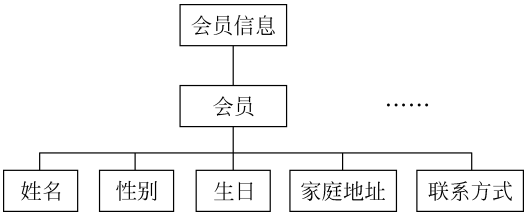

图 3-2　文件 3-1-2-1.xml 对应元素的树状结构

在树状结构中，<姓名><性别><生日><家庭住址><联系方式>元素为叶子元素，<会员信息><会员>为枝干元素。需要注意的是，<会员信息>为根元素，由于根元素中包含子元素，因此在元素分类中，根元素可以归为枝干元素。

1. 叶子元素声明语句

```
<!ELEMENT 元素名 元素内容类型>
```

语句说明如下。

① <! ELEMENT：元素声明语句的开始，关键字 ELEMENT 必须大写。

② 元素名：表示所声明元素的名称。

③ 元素内容类型：表示元素中数据内容的类型，常用类型有＃PCDATA、EMPTY 和 ANY；其中＃PCDATA 表示字符类型数据，EMPTY 表示空元素，ANY 表示元素为任意内容。

例如：

```
<!ELEMENT 姓名 (＃PCDATA)>
```

对应有效的 XML 文档为：

```
<姓名>张红</姓名>
```

或者：

```
<姓名/>
```

2. 枝干元素声明语句

```
<!ELEMENT 元素名 (子元素 1,子元素 2,……)>
```

语句说明如下。

① <! ELEMENT：元素声明语句的开始，关键字 ELEMENT 必须大写。

② 元素名：表示所声明的元素名称。

③ (子元素 1,子元素 2,……)：表示枝干元素的若干个子元素。

注意以下两点。

① 枝干元素内包含的子元素如果不止一个，需要用半角的逗号“,”将各子元素分隔开。

② 子元素列表(子元素 1,子元素 2,……)不仅定义该枝干元素拥有的子元素个数，而且表示该元素对应子元素出现时所必须遵守的顺序，并且每个子元素只能出现一次。

例如：

```
<!ELEMENT 会员信息 (姓名,性别,生日)>
<!ELEMENT 姓名 (＃PCDATA)>
<!ELEMENT 性别 (＃PCDATA)>
<!ELEMENT 生日 (＃PCDATA)>
```

对应有效的 XML 文档为：

```
<会员信息>
    <姓名>张红</姓名>
   <性别>女</性别>
   <生日>1986－02－15</生日>
</会员信息>
```

下面的 XML 文档中，子元素顺序错误，是一个非有效的 XML 文档：

```
<会员信息>
    <性别>女</性别>
   <姓名>张红</姓名>
```

```
    <生日>1986-02-15</生日>
</会员信息>
```

3. 选择性子元素的声明

在 XML 文档中，有时需要在两个或多个互斥的元素中选择某一元素作为其子元素，则语法格式为：

```
<!ELEMENT 元素名 (子元素 1|子元素 2|……)>
```

注意：可供选择的各个子元素之间需用“|”符号分隔，并只能在子元素列表中选择其一作为它的子元素。

例如：

```
<!ELEMENT 会员类型 (金卡会员|普通会员)>
<!ELEMENT 金卡会员 (#PCDATA)>
<!ELEMENT 普通会员 (#PCDATA)>
```

对应有效的 XML 文档为：

```
<会员类型>
    <金卡会员>三倍积分</金卡会员>
</会员类型>
```

或者：

```
<会员类型>
    <普通会员>双倍积分</普通会员>
</会员类型>
```

3.2.3　控制子元素出现次数的声明

在 DTD 中定义枝干元素时，如果该元素中的某个或某些子元素需要重复出现或不出现，此时可以使用特殊符号表示该元素子元素出现的次数，以达到对子元素次数的控制。控制子元素出现次数的符号如表 3-1 所示。

表 3-1　控制子元素出现次数的符号

声明符号	表示含义
无符号	子元素只能出现 1 次
?	子元素只能出现 0 次或 1 次
+	子元素至少出现 1 次或多次
*	子元素可出现 0 次或多次，即可出现任意次

控制子元素出现次数的符号可相互结合使用，达到对 XML 文档中子元素出现次数的灵活运用，通过以下几种常用实例进行说明。

1. 子元素只出现一次

例如：

```
<!ELEMENT 会员(姓名,联系方式)>
```

说明：表示<会员>元素中只能有<姓名>和<联系方式>两个子元素，并且两个子元素必须按顺序出现。对应有效文档为：

```
<会员>
    <姓名>张红</姓名>
    <联系方式> 23232233 </联系方式>
</会员>
```

2. 子元素至少出现一次

例 3-1

```
<!ELEMENT 会员 (姓名,联系方式) + >
```

说明：表示<会员>元素中有<姓名>和<联系方式>子元素，<姓名>和<联系方式>两个子元素必须按顺序出现，且至少出现一次或多次。对应有效文档的其中一种方式为：

```
<会员>
    <姓名>张红</姓名>
    <联系方式> 23232233 </联系方式>
    <姓名>赵楠</姓名>
    <联系方式> 56561234 </联系方式>
</会员>
```

例 3-2

```
<!ELEMENT 会员 (姓名,联系方式 + ) >
```

说明：表示<会员>元素中有<姓名>和<联系方式>子元素，其中<姓名>元素出现 1 次；<姓名>元素后为<联系方式>元素，且<联系方式>元素至少出现一次或多次。对应有效文档的其中一种方式为：

```
<会员>
    <姓名>张红</姓名>
    <联系方式> 23232233 </联系方式>
    <联系方式> 13233339999 </联系方式>
</会员>
```

3. 子元素出现零次或多次

例 3-3

```
<!ELEMENT 会员 (姓名, 联系方式) * >
```

说明：表示<会员>元素中的<姓名>和<联系方式>两个子元素必须按顺序同时出现，且可出现任意次。

例 3-4

```
<!ELEMENT 会员 (姓名, 联系方式 * ) >
```

说明：表示<会员>元素中的<姓名>和<联系方式>两个子元素必须按顺序同时出现，且<姓名>元素在前，必须出现一次，<联系方式>在后，且可出现任意次。

4. 子元素出现零次或一次

例 3-5

```
<!ELEMENT 会员 (姓名?)>
```

说明：表示<会员>元素中的<姓名>元素，可以出现，也可以不出现。

5. 子元素列表选择

如果选择性子元素中使用控制符号，那么对元素内容的控制将更加灵活。例如，<! ELEMENT 会员类型（金卡会员|普通会员）*>，选择性子元素（金卡会员|普通会员）中使用了控制符号“*”，其元素可以出现任意次，且顺序随意，以此达到对元素内容的灵活控制。

6. 混合型元素的声明

在 XML 中的枝干元素中，既可以存在子元素，也可以包含文本数据，这种元素称为混合型元素。混合型元素的声明的语法格式为：

```
<!ELEMENT 元素名 (#PCDATA|子元素 1|子元素 2|……)*>
```

应注意以下几点。

① 混合型元素在声明时，不可使用逗号，只能使用“|”符号分隔各元素。

② 枝干元素中的子元素，要求将每个子元素写在#PCDATA 之后。

③ 枝干元素中的各个子元素设置后，必须使用“*”符号。

文件 3-2-3-1. xml

```
<?xml version = "1.0" encoding = "gb2312"?>
<!DOCTYPE 会员信息 [
    <!ELEMENT 会员信息 (会员)>
    <!ELEMENT 会员 (#PCDATA | 姓名 | 联系方式)*>
    <!ELEMENT 姓名 (#PCDATA)>
    <!ELEMENT 联系方式 (#PCDATA)>
]>
<会员信息>
    <会员>这是关于张红的信息
        <姓名>张红</姓名>
        <联系方式> 23232233 </联系方式>
    </会员>
</会员信息>
```

注意：混合型元素的声明方式不是非常严格，在实际应用中需根据实际情况谨慎使用。

3.2.4　XML 元素的数据类型

1. 空元素

空元素是指在标记间没有任何数据内容的元素。空元素的存在不影响 XML 数据的正

确性,空元素可以存放属性提供的额外信息。在 DTD 中使用关键字 EMPTY 定义空元素。其语法格式为:

```
<!ELEMENT 元素名 EMPTY>
```

例如:

```
<!ELEMENT 照片 EMPTY>
```

对应有效的 XML 文件为:

```
<照片></照片>
```

或者:

```
<照片/>
```

注意:关键字 EMPTY 应大写,并且不能加括号。

如果将元素写为<! ELEMENT 照片 (EMPTY)>,则系统会认为<照片>元素中包含<EMPTY>子元素,这是错误的写法。

2. 元素数据

如果元素内包含数据,则数据内容分为两种,分别使用关键字 #PCDATA 和 ANY 来定义。

元素可以使用关键字 #PCDATA 描述文本数据内容。#PCDATA 是 Parser Character DATA 的缩写,意为可解析的字符数据,其含义是指该元素的数据内容是符合语法规则的文本数据字符串,该字符串可以包含任意数量的字符数据,或者不包含任何内容。

如果不对元素的内容进行任何限制,可以使用 ANY 关键字,表示该元素中的内容可以是文本数据、子元素内容、空元素以及混合型元素等。

包含数据的 XML 文档对应的 DTD 语法为:

```
<!ELEMENT 元素名 (#PCDATA)>
```

或者:

```
<!ELEMENT 元素名 ANY>
```

说明:(#PCDATA)要求使用括号括起来,且 PCDATA 必须使用大写字符。ANY 不需使用括号,ANY 关键字也必须使用大写字符。

需要注意的是,采用关键字 ANY,可以将元素设置为任意顺序,同时不限定元素数量,因此使用 ANY 定义元素与 DTD 的设计目标(约束和规范 XML 文档)相违背,应尽量避免使用。ANY 一般不在最终的 DTD 文档中使用,而是在 DTD 文档的设计初期,其文档元素在还未确定的情况下,可暂时使用 ANY。在逐步完善 DTD 的过程中,再使用确定的其他元素取代。

3.3 DTD 中属性的声明

在 XML 中,属性是用来包含元素的额外信息。一个有效的 XML 文档,必须在相应的

DTD 中明确地声明与文档中元素一起使用的所有属性，这些在 DTD 中所声明的属性名称和具体的属性值包含在元素的起始标记中。

3.3.1　属性的声明语法

在 DTD 中，可以通过关键字 ATTLIST 声明属性，一个属性声明可以声明一个元素的一个或多个属性。其语法格式为：

```
<! ATTLIST 元素名 属性名 属性值类型 属性附加声明>
```

语法格式说明如下。

① ＜!ATTLIST：表示属性声明语言的开始。

② 元素名：属性所属的 XML 元素的名称。

③ 属性名：XML 元素对应属性的名称。

④ 属性值类型：指定属性值存在的类型。

⑤ 属性附加声明：描述属性额外的相关信息。

需注意以下几点。

① 一个元素可以定义多个属性，如果这些元素在一行定义，各属性之间需要使用空格分隔，也可以多行定义。如果需要在一条语句中为某个元素定义多个属性，其语法格式为：

```
<!ATTLIST 元素名 属性名 1 属性值 1 类型 属性 1 附加声明
                属性名 2 属性值 2 类型 属性 2 附加声明 …>
```

② 属性附加声明要紧跟在属性值类型之后。

在 DTD 中，属性中对应的属性值的类型，是用于描述属性值以何种方式存在的类型，也就是用于指定属性值中数据内容的形式。常见的属性值类型如表 3-2 所示。

表 3-2　常见的属性值类型

类　型	描　述
CDATA	值为字符数据
枚举类型	格式：(en1|en2|…)，表示此值是枚举列表中的一个值
ID	值为唯一的 id
IDREF	值为另外一个元素的 id
IDREFS	值为其他 id 的列表
NMTOKEN	值为合法的 XML 名称
NMTOKENS	值为合法的 XML 名称的列表
ENTITY	值是一个实体
ENTITIES	值是一个实体列表
NOTATION	值是符号的名称
xml：	值是一个预定义的 XML 值

在 DTD 中，属性附加声明是用于描述属性额外的相关信息，常用属性附加声明如表 3-3 所示。

表 3-3 常用属性附加声明

类 型	描 述
#REQUIRED	元素的属性是必须存在的,且必须给出一个属性值
#IMPLIED	元素中的属性可有可无
#FIXED "固定值"	元素中属性所对应的属性值是固定的,不能更改
"默认值"	元素中属性对应的默认值

下面通过一个实例了解 XML 属性的定义方式。

文件 3-3-1-1.xml

```
<?xml version = "1.0" encoding = "gb2312"?>
<!-- 会员信息相关文档 -->
<!DOCTYPE 会员信息 [
<!ELEMENT 会员信息 (会员 * )>
<!ELEMENT 会员 (姓名,联系方式)>
<!ELEMENT 姓名 (#PCDATA)>
<!ELEMENT 联系方式 (#PCDATA)>
<!ATTLIST 会员 卡号 ID #REQUIRED>
<!ATTLIST 姓名 性别 (男|女) #IMPLIED
                生日 CDATA #REQUIRED>
]>
<会员信息>
    <会员 卡号 = "SY102030">
        <姓名 性别 = "女" 生日 = "1986 - 02 - 15">张红</姓名>
            <联系方式> 23232233 </联系方式>
    </会员>
    <会员 卡号 = "SY102031">
            <姓名 生日 = "1988 - 11 - 20">赵楠</姓名>
        <联系方式> 56561234 </联系方式>
    </会员>
</会员信息>
```

其中,属性声明"<!ATTLIST 会员 卡号 ID #REQUIRED>"表明<会员>元素中有属性"卡号",属性值的类型是 ID 类型,即属性值只能唯一出现,"#REQUIRED"表明属性必须存在。

属性声明"<!ATTLIST 姓名 性别 (男|女) #IMPLIED 生日 CDATA #REQUIRED>"表明<姓名>元素中有属性"性别"和"生日"。其中"性别"属性值的类型为枚举类型,即属性值只能为"男"或"女"中的一个,"#IMPLIED"表明该属性可有可无;"生日"属性值的类型为 CDATA,即符合语法的任意字符串,"#REQUIRED"表明属性必须存在。

3.3.2 属性的附加声明

在 DTD 属性声明的语法中,根据 XML 文件是否必须为属性提供相应的值,通过属性的附加声明来描述。DTD 中一般使用 4 种方式描述属性的附加声明,分别为 #REQUIRED、#IMPLIED、#FIXED "固定值"和默认值。本书根据表 3-3 所示的属性附加声明类型进行详细介绍。

1. #REQUIRED

#REQUIRED 表示该元素的属性是必须存在的，且必须给出一个属性值。例如，DTD 为：

```
<!ELEMENT 会员 (#PCDATA)>
<!ATTLIST 会员 生日 CDATA #REQUIRED>
```

说明：DTD 声明<会员>元素的属性“生日”是必须存在的，且必须给出一个对应的属性值。对应有效的 XML 文本为：

```
<会员 生日 = "1986 - 02 - 15">张红</会员>
```

改为以下方式，则为无效的 XML 文档：

```
<会员>张红</会员>
```

2. #IMPLIED

#IMPLIED 表示在 XML 文档中该元素的属性是可有可无的。

```
<!ELEMENT 姓名 (#PCDATA)>
<!ATTLIST 姓名 生日 CDATA #IMPLIED>
```

说明：DTD 声明<姓名>元素的属性“生日”可有可无。对应有效的 XML 文本为：

```
<姓名 生日 = "1986 - 02 - 15">张红</会员>
```

或者：

```
<会员>张红</会员>
```

3. #FIXED "固定值"

表示在 XML 文档中该元素的这个属性值是给定的固定值，不能更改。

```
<!ELEMENT 会员 (#PCDATA)>
<!ATTLIST 会员 类型 CDATA #FIXED "金卡会员" >
```

说明：DTD 声明<会员>元素的属性为“类型”，其对应固定值为“金卡会员”，且内容不能更改，即使属性不出现，其值也默认为“金卡会员”。对应有效的 XML 文本为：

```
<会员 类型 = "金卡会员">刘洋</会员>
```

或者：

```
<会员>刘洋</会员>
```

4. 默认值

若 XML 文档没有规定必须设定元素的属性，但为了便于应用程序的处理需求，可以指定属性的默认值。如果该元素没有设定属性，则使用默认值作为其属性；如果该元素的属

性需要指定其他值,则其他值可覆盖默认值。

```
<!ELEMENT 会员 (#PCDATA)>
<!ATTLIST 会员 类型 CDATA "普通会员" >
```

说明:DTD 声明<会员>元素的属性“类型”对应一个默认值为“普通会员”,对应有效的 XML 文本为:

```
<会员 类型 = "普通会员">刘洋</会员>
```

或者:

```
<会员>刘洋</会员>
```

<会员>元素的属性即使设定为默认值,但“类型”属性仍然可以指定其他值,例如:

```
<会员 类型 = "金卡会员">王刚</会员>
```

3.3.3 属性值的类型

属性值类型是为属性指定其属性值的种类,具体类型如表 3-2 所示。本书介绍常用的几种属性值类型。

1. CDATA 类型

CDATA(Character DATA)表示属性值是简单的文本数据,即任意符合语法的字符串,但当字符串中出现“<”或“&”等字符时,需要使用特殊字符替代。CDATA 是使用的最普遍、最简单的数据类型。

2. ID 类型

ID 表示属性值具有唯一性,一般用来表示个人唯一身份的内容,如编号、学号、身份证号等。ID 表示的属性值在整个 XML 文档中不可重复,而且第一个字符只能使用中英文字符或下画线。

下面通过一个实例了解属性中 ID 类型的使用方式。

文件 3-3-3-1.xml

```
<?xml version = "1.0" encoding = "gb2312"?>
<!DOCTYPE 简历信息 [
<!ELEMENT 简历信息 (简历 + )>
<!ELEMENT 简历 (联系人,联系方式 + )>
<!ELEMENT 联系人 (#PCDATA)>
<!ELEMENT 联系方式 (#PCDATA)>
<!ATTLIST 联系人 编号 ID #REQUIRED >
]>
<!-- 每个联系人都必须有一个唯一的编号 -->
<简历信息>
    <简历>
        <联系人 编号 = "A001">李智</联系人>
        <联系方式> 23228888 </联系方式>
```

```
        <联系方式> lizhi@aaa.com </联系方式>
    </简历>
    <简历>
        <联系人 编号 = "A002">王鹏</联系人>
        <联系方式> wangpeng@bbb.com </联系方式>
    </简历>
    <简历>
        <联系人 编号 = "A003">张海清</联系人>
        <联系方式> zhanghaiqing@aaa.com </联系方式>
    </简历>
</简历信息>
```

说明："<! ATTLIST 联系人 编号 ID #REQUIRED >"定义了<联系人>元素的属性"编号"必须出现，且"编号"对应的属性值只能唯一出现。

需要注意的是，声明属性值为 ID 类型时，一般不能指定默认值，也不能使用 #FIXED 设定固定值，ID 类型的属性经常使用 #REQUIRED 进行附加声明。

3. IDREF 类型

IDREF 和 IDREFS 类型的属性必须引用对应的 ID 属性值。对于 IDREF 类型的属性，其属性值必须引用文档中出现的某个 ID 值。下面通过一个实例了解属性中 IDREF 类型的使用方法。

文件 3-3-3-2. xml

```
<?xml version = "1.0" encoding = "gb2312"?>
<! DOCTYPE 简历信息 [
<! ELEMENT 简历信息 (简历 + ,应聘职位 + )>
<! ELEMENT 简历 (联系人,联系方式 + )>
<! ELEMENT 联系人 ( #PCDATA)>
<! ELEMENT 联系方式 ( #PCDATA)>
<! ELEMENT 应聘职位 ( #PCDATA)>
<! ATTLIST 联系人 编号 ID #REQUIRED >
<! ATTLIST 应聘职位 入选编号 IDREF #REQUIRED >
]>
<!-- 每个联系人都必须有一个唯一的编号 -->
<!-- 入选编号对应的属性值必须是编号中曾经出现的一个值 -->
<简历信息>
    <简历>
        <联系人 编号 = "A001">李智</联系人>
        <联系方式> 23228888 </联系方式>
        <联系方式> lizhi@aaa.com </联系方式>
    </简历>
    <简历>
        <联系人 编号 = "A002">王鹏</联系人>
        <联系方式> wangpeng@bbb.com </联系方式>
    </简历>
    <简历>
        <联系人 编号 = "A003">张海清</联系人>
        <联系方式> zhanghaiqing@aaa.com </联系方式>
```

```
    </简历>
    <应聘职位 入选编号 = "A001">经理</应聘职位>
    <应聘职位 入选编号 = "A003">副经理</应聘职位>
</简历信息>
```

说明："<! ATTLIST 应聘职位 入选编号 IDREF #REQUIRED>"定义了<应聘职位>元素的属性"入选编号"必须出现，且"入选编号"只能从 ID 值中选取一个作为它的属性值出现。

4. IDREFS 类型

对于 IDREFS 类型的属性，其对应的属性值必须引用文档中出现的某一个或多个 ID 值。如果对应多个 ID 属性值，这多个属性值之间需要用空格分开，并且放在同一对半角的引号中。下面通过一个实例了解属性中 IDREFS 类型的使用方法。

文件 3-3-3-3.xml

```
<?xml version = "1.0" encoding = "gb2312"?>
<!DOCTYPE 简历信息 [
<!ELEMENT 简历信息 (简历 + ,应聘职位 + )>
<!ELEMENT 简历 (联系人,联系方式 + )>
<!ELEMENT 联系人 (#PCDATA)>
<!ELEMENT 联系方式 (#PCDATA)>
<!ELEMENT 应聘职位 (#PCDATA)>
<!ATTLIST 联系人 编号 ID #REQUIRED>
<!ATTLIST 应聘职位 入选编号 IDREFS #REQUIRED>
]>
<!-- 每个联系人都必须有一个唯一的编号 -->
<!-- 入选编号对应的属性值必须是编号中曾出现的一个或多个值 -->
<简历信息>
    <简历>
        <联系人 编号 = "A001">李智</联系人>
        <联系方式> 23228888 </联系方式>
        <联系方式> lizhi@aaa.com </联系方式>
    </简历>
    <简历>
        <联系人 编号 = "A002">王鹏</联系人>
        <联系方式> wangpeng@bbb.com </联系方式>
    </简历>
    <简历>
        <联系人 编号 = "A003">张海清</联系人>
        <联系方式> zhanghaiqing@aaa.com </联系方式>
    </简历>
    <应聘职位 入选编号 = "A001">经理</应聘职位>
    <应聘职位 入选编号 = "A002 A003">副经理</应聘职位>
</简历信息>
```

说明："<! ATTLIST 应聘职位 入选编号 IDREFS #REQUIRED>"定义了<应聘职位>元素的属性"入选编号"必须出现，且"入选编号"是从 ID 值中选取一个或多个作为它的属性值出现。

5. 枚举类型

枚举类型的属性表示需要限制属性值只能是其中的一个。下面通过一个实例了解属性中枚举类型的使用方法。

文件 3-3-3-4.xml

```
<?xml version = "1.0" encoding = "gb2312"?>
<!DOCTYPE 简历信息 [
<!ELEMENT 简历信息 (简历 + )>
<!ELEMENT 简历 (联系人,联系方式 + )>
<!ELEMENT 联系人 ( #PCDATA)>
<!ELEMENT 联系方式 ( #PCDATA)>
<!ATTLIST 联系人 性别 (男|女) #REQUIRED >
]>
<简历信息>
    <简历>
        <联系人 性别 = "男">李智</联系人>
        <联系方式> 23228888 </联系方式>
        <联系方式> lizhi@aaa.com </联系方式>
    </简历>
    <简历>
        <联系人 性别 = "男">王鹏</联系人>
        <联系方式> wangpeng@bbb.com </联系方式>
    </简历>
    <简历>
        <联系人 性别 = "女">张海清</联系人>
        <联系方式> zhanghaiqing@aaa.com </联系方式>
    </简历>
</简历信息>
```

说明："<! ATTLIST 联系人 性别 (男|女) #REQUIRED >"定义了<联系人>元素的属性"性别"必须出现,且"性别"只能从"男"或"女"中选取一个作为它的属性值出现。

6. NMTOKEN

NMTOKEN(NameTOKEN)表示 XML 名称标记,它对应的属性值遵守 XML 元素名称的命名规则,即所对应的属性值使用的字符必须是中英文字符、数字、点字符、短横线、下画线。此外,还可使用冒号,且第一个字符可以是任意字符。NMTOKEN 类型是 CDATA 类型的一个子集。下面通过一个实例了解属性中 NMTOKEN 类型的使用方法。

文件 3-3-3-5.xml

```
<?xml version = "1.0" encoding = "gb2312"?>
<!DOCTYPE 个人简历 [
    <!ELEMENT 个人简历 (简历 + )>
    <!ELEMENT 简历 (联系人)>
    <!ELEMENT 联系人 ( #PCDATA)>
    <!ATTLIST 联系人 联系方式 NMTOKEN #REQUIRED >
]>
<个人简历>
```

```
    <简历><联系人 联系方式 = "010 - 23228888">李智</联系人></简历>
    <简历><联系人 联系方式 = "021 - 43435555">王鹏</联系人></简历>
    <简历><联系人 联系方式 = "010 - 44553322">张海清</联系人></简历>
</个人简历>
```

说明："<! ATTLIST 联系人 联系方式 NMTOKEN ＃REQUIRED>"定义了<联系人>元素的属性"联系方式"必须出现,且"联系方式"属性值可以是数字和短横线组合的字符串。

7. NMTOKENS

NMTOKENS类型包含一个或多个XML名称标记,多个属性值之间用空格分开,并放在一对半角的引号中。下面通过一个实例了解属性中NMTOKENS类型的使用方法。

文件 3-3-3-6. xml

```
<?xml version = "1.0" encoding = "gb2312"?>
<! DOCTYPE 个人简历 [
    <! ELEMENT 个人简历 (简历 + )>
    <! ELEMENT 简历 (联系人)>
    <! ELEMENT 联系人 ( ＃PCDATA)>
    <! ATTLIST 联系人 联系方式 NMTOKENS ＃REQUIRED >
]>
<个人简历>
    <简历>
        <联系人 联系方式 = "010 - 23228888 010 - 010 - 23226666">李智</联系人>
    </简历>
    <简历>
        <联系人 联系方式 = "021 - 43435555 021 - 43436666">王鹏</联系人>
    </简历>
    <简历>
        <联系人 联系方式 = "010 - 44553322 13844332233">张海清</联系人>
    </简历>
</个人简历>
```

说明："<! ATTLIST 联系人 联系方式 NMTOKEN ＃REQUIRED>"定义了<联系人>元素的属性"联系方式"必须出现,且"联系方式"属性值可以是数字和短横线组合的多个字符串,不同的字符串之间用空格隔开,并放在一对半角单引号或双引号中。

8. ENTITY

ENTITY类型指定的属性用于定义DTD中的实体。实体是用于定义和引用文本或特殊字符快捷方式的变量。ENTITY类型中对应的属性能够将图像、声音等二进制数据的内容进行引用,该类实体属于不可解析实体。下面通过一个实例了解属性中ENTITY类型的使用方法。

文件 3-3-3-7. xml

```
<?xml version = "1.0" encoding = "GB2312"?>
<! DOCTYPE images [
<! ELEMENT images ( image * )>
<! ELEMENT image EMPTY >
```

```
<!ATTLIST image source ENTITY #REQUIRED>
<!ENTITY src SYSTEM "image.gif">
]>
<images>
    <image source="src"/>
</images>
```

9. ENTITYS

ENTITYS类型指定的属性用于定义DTD中实体的集合。下面通过一个实例了解属性中ENTITYS类型的使用方法。

```
<!ELEMENT images EMPTY>
<!ATTLIST images sources ENTITYS #REQUIRED>
<!ENTITY image1 SYSTEM "1.gif">
<!ENTITY image2 SYSTEM "2.gif">
```

3.4　DTD的基本结构

DTD是有效的XML文本的基础，使用DTD可以规范XML语法。DTD和XML进行相互关联时，必须遵循一定的语法规则。在XML中使用DTD进行程序设计时，有3种引用方式，分别为内部DTD的引用、外部DTD的引用和混合DTD的引用。

3.4.1　内部DTD的引用

内部DTD引用方式是指在XML文档中直接包含DTD的相应文本，XML和DTD在同一个XML文本中存在。

内部DTD存在于XML文本中，如文件3-1-2-1.xml是一个内部DTD。DTD文件头必须使用专用的关键字以告知解析器这个区段数据是DTD的声明内容，其声明语法为：

```
<?xml version="1.0" encoding="UTF-8" standalong= "yes"?>
<!DOCTYPE 根元素名 [
    <!-- DTD定义内容 -->
    …… 
]>
```

注意：关键字DOCTYPE必须大写，内部DTD可以在XML声明中将standalong设置为yes。

3.4.2　外部DTD的引用

外部DTD引用方式是指将DTD作为一个独立的文件单独保存，该DTD是一个文本文件，扩展名为.dtd。引入外部DTD文档，需要在XML文本的DOCTYPE指令中使用关键字SYSTEM或者PUBLIC进行引用，以达到对XML文本数据规范的目的。

外部DTD作为一个外部文件可以被多个XML文档引用，其优点是不同的组织和个人

如果使用相同的 DTD 规范 XML 文本,就可以将 DTD 作为一个单独的文件保存,然后同时引用该 DTD 进行数据交流,这样一个 DTD 就能够被多个 XML 文档共享。

在 Altova XMLSpy 环境中,建立 DTD 文件的步骤是:单击【File】→【New】菜单命令,选择【Create new document】对话框中的【dtd Document Type Definition】列表项,单击【OK】按钮,即可创建 DTD 文档,如图 3-3 所示。

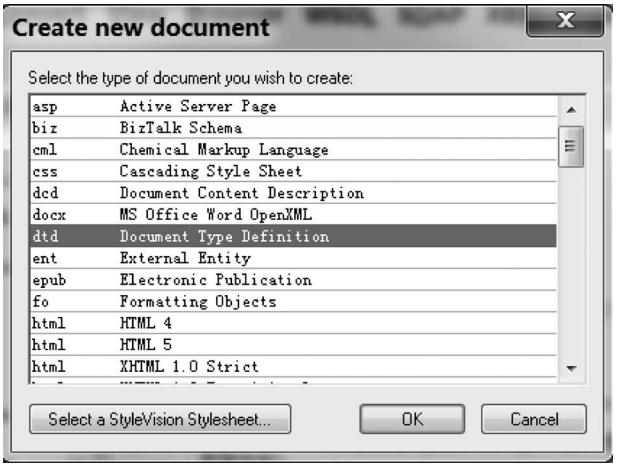

图 3-3 新建 DTD 文档对话框

在外部 DTD 的语法格式中,由于 DTD 是 XML 从 SGML 中继承而来的一种验证机制,因此 DTD 文件的第一行是 XML 声明语句。外部 DTD 的语法格式为:

```
<?xml version = "1.0" encoding = "UTF - 8" standalong = "no"?>
<!-- DTD 元素声明 -->
<!-- DTD 属性声明 -->
<!-- DTD 实体声明 -->
…… 
```

对于外部 DTD,根据其性质分为两种:一种是公有文件,是指国际上标准组织制定或者行业内部得到广泛认可的 DTD 文件,使用关键字 PUBLIC;另一种是私有文件,指未公开的、属于个人或某些组织的 DTD 文件,使用关键字 SYSTEM。

DTD 需要在 XML 文档中引用才可使用,根据两种不同的外部 DTD,XML 规定了相应的两种引用方式。

1. 引用公有的 DTD 文件

在 XML 中引用公有的 DTD 文件的语法格式为:

```
<!DOCTYPE 根元素 PUBLIC "文件路径及文件名">
```

或者

```
<!DOCTYPE 根元素 PUBLIC "文件名" "文件路径">
```

语法格式说明如下。

① <!DOCTYPE：表示 DTD 声明的开始。
② 根元素：指定 XML 中根元素的名称。
③ PUBLIC：指定外部 DTD 文件是公有的。
④ 文件路径及文件名：通过 URL 将外部 DTD 引用到 XML 文档中。
⑤ 文件名：公有 DTD 的逻辑名，使用时需要调用这个指定的逻辑名。
⑥ 文件路径：文件的 URL 地址。

注意：关键字 DOCTYPE 和 PUBLIC 必须大写。例如：

```
<!DOCTYPE struts-config PUBLIC
          "-//Apache Software Foundation//DTD Struts Configuration 1.2//EN"
          "http://struts.apache.org/dtds/struts-config_1_2.dtd">
```

2. 引用私有的 DTD 文件

在 XML 中引用私有的 DTD 文件的语法格式为：

```
<!DOCTYPE 根元素 SYSTEM "文件路径及文件名">
```

说明如下。
① <!DOCTYPE：表示 DTD 声明的开始。
② 根元素：指定 XML 中根元素的名称。
③ SYSTEM：指定外部 DTD 文件是私有的。
④ 文件路径及文件名：通过 URL 将外部 DTD 引用到 XML 文档中。

注意：关键字 DOCTYPE 和 SYSTEM 必须大写。

使用 SYSTEM 引用外部 DTD 文档，一般适合教学或个人团队使用。本章的教学实践中都使用 SYSTEM 关键字进行引用。

例如，一个名为 book.dtd 的外部 DTD 文件存放在 URL 为 http://www.aaa.com 的网址中，那么，在 XML 中引用该 DTD 文件的指令为：

```
<!DOCTYPE book SYSTEM "http://www.aaa.com/book.dtd">
```

定义一个名为 book.dtd 的外部 DTD 文件存放在硬盘 C:\Administratord 对应的地址中，那么在 XML 中引用该 DTD 文件绝对路径对应的指令为：

```
<!DOCTYPE book SYSTEM "C:\Administrator\book.dtd">
```

如果 book.dtd 和对应 XML 文档在同一文件夹，可以在 XML 中使用 book.dtd 的相对路径，对应的指令为：

```
<!DOCTYPE book SYSTEM " book.dtd">
```

下面将文件 3-1-2-1.xml 对应的文档改为外部 DTD 文件，对应的 DTD 文档内容如下：

文件 3-4-2-1.dtd

```
<?xml version="1.0" encoding="gb2312"?>
<!-- 会员信息对应的 DTD 文件，文件名为"3-4-2-1.dtd" -->
<!ELEMENT 会员信息 (会员*)>
```

```
<!ELEMENT 会员 (姓名,性别,生日,家庭住址,联系方式)>
<!ELEMENT 姓名 (#PCDATA)>
<!ELEMENT 性别 (#PCDATA)>
<!ELEMENT 生日 (#PCDATA)>
<!ELEMENT 家庭住址 (#PCDATA)>
<!ELEMENT 联系方式 (#PCDATA)>
<!ATTLIST 会员 卡号 ID #REQUIRED>
```

文件 3-4-2-1.xml

```
<?xml version="1.0" encoding="gb2312"?>
<!-- 会员信息对应的 XML 文档 -->
<!DOCTYPE 会员信息 SYSTEM "3-4-2-1.dtd">
<会员信息>
    <会员 卡号="SY102030">
        <姓名>张红</姓名>
        <性别>女</性别>
        <生日>1986-02-15</生日>
        <家庭住址>金地滨河小区 2 号楼 203 室</家庭住址>
        <联系方式>23232233</联系方式>
    </会员>
    <会员 卡号="SY102031">
        <姓名>赵楠</姓名>
        <性别>女</性别>
        <生日>1988-11-20</生日>
        <家庭住址>万科新里程 5 号楼 1103 室</家庭住址>
        <联系方式>56561234</联系方式>
    </会员>
</会员信息>
```

使用外部 DTD 的优点是可以将常用的外部 DTD 放在一个共享的地址中,就能够被不同的 XML 引用,一般建议使用外部 DTD。

3.4.3 混合 DTD 引用方式

在 XML 文本中既使用了外部 DTD 也使用了内部 DTD,这种方式称为混合 DTD。一般是某些 XML 文本需要使用已公开的外部 DTD,但是还需要在此 DTD 中加入新的内容,这部分新的内容可以作为内部 DTD 定义。混合 DTD 的使用,不仅方便 XML 在内部 DTD 中增加新的内容,而且不影响其他外部 DTD 的 XML 文档,使用比较灵活。

需要注意的是,使用混合 DTD 时,不允许在两个 DTD 中同时定义同一个元素或属性;否则 XML 解析器将提示错误。

例如,已经创建好一个相关的外部 DTD,对应文档如下:

文件 3-4-3-1.dtd

```
<?xml version="1.0" encoding="gb2312"?>
<!ELEMENT 会员 (姓名,性别,联系方式)>
```

创建一个 XML 文件,需要调用文件 3-4-3-1.dtd 对应内容,并且在该文件中增加新的信息,对应文件如下:

文件 3-4-3-1.xml

```
<?xml version = "1.0" encoding = "gb2312"?>
<!DOCTYPE 会员 SYSTEM "3 - 4 - 3 - 1.dtd " [
    <!ELEMENT 姓名 (#PCDATA)>
    <!ELEMENT 性别 (#PCDATA)>
    <!ELEMENT 联系方式 (手机, 座机, E - Mail)>
    <!ELEMENT 手机 (#PCDATA)>
    <!ELEMENT 座机 (#PCDATA)>
    <!ELEMENT E - Mail (#PCDATA)>
]>
<会员>
    <姓名>赵楠</姓名>
    <性别>女</性别>
    <联系方式>
        <手机> 13212341122 </手机>
        <座机> 56561234 </座机>
        < E - Mail > zhaonan@163.com </E - Mail >
    </联系方式>
</会员>
```

3.5　实体的声明与引用

从数据处理的角度看，现实世界中的客观事物称为实体，它是现实世界中任何可区分、可识别的事物。实体可以指人，如教师、学生等，也可以指物，如书、桌子等。实体不仅可以描述能够触及的客观对象；也可以描述抽象的事件，如演出、比赛等；还可以描述事物与事物之间的联系，如学生选课、客户订货等。

在 XML 文本中，实体是用来存放符合语法规则的 XML 文档片段的单元。文档片段是 XML 中的一段代码，也可以仅仅是某个元素或者多个元素的组合，还可以是如图像、声音等二进制数据。因此，实体是 XML 文档中一种数据单位，用于定义引用普通文本或特殊字符的变量。使用 XML 的实体机制，能够将多种不同形态的数据合并加入 XML 文件中。因此，实体的使用是在 XML 中节省大量时间的一种工具。

3.5.1　实体的分类

在 XML 中对应的实体是一个定义好的数据或数据集合，通过相应的引用方式，将这些数据或数据集合引入 XML 或者 DTD 所需的地方。实体可以简单地理解为引用数据的一种方法，数据可以是普通的文本也可以是二进制数据。定义实体时，每个实体都有一个自己的名字，以及对应需要定义的实体内容，通过实体名字引入实体内容，解析器就可以将具体的实体内容来代替文档中的实体名显示在浏览器中，方便用户获取文档信息，如在 2.8 节中使用的预定义实体就是一种特殊的实体。

根据实体种类的不同，实体可分为以下 3 大类。

1. 通用实体和参数实体

通用实体是最简单的实体形式,通常用来替代文档具体内容,一般在 DTD 中定义,可以在 XML 和 DTD 中引用。参数实体是一种只能在 DTD 中进行定义和使用的实体类型。

2. 内部实体和外部实体

在实体定义中,如果实体内容定义并没有关联外部独立的文件,则称为内部实体;如果实体内容保存在一个独立的外部文件中,则称为外部实体。

3. 可解析实体和不可解析实体

根据实体内容是否能够被解析器解析,分为可解析实体和不可解析实体。可解析实体是规范的 XML 文本;不可解析实体是不应该被解析器解析的二进制数据。

一般情况下,可以将不同实体进行组合,生成更多种类实体。常用的实体组合为内部通用实体、外部通用实体、内部参数实体、外部参数实体、内部可解析实体、外部可解析实体、内部不可解析实体及外部不可解析实体等。

3.5.2 内部通用实体

内部通用实体是在 DTD 中定义的一段具体数据内容,可以在 XML 元素或者 DTD 中引用。实体使用<! ENTITY >进行声明。在 DTD 中声明内部通用实体的格式为:

```
<!ENTITY 实体名 实体内容>
```

说明如下。

① <! ENTITY:表示开始声明一个实体,关键字 ENTITY 必须大写。

② 实体名:表示实体的具体名称。该名称必须以下画线或中英文字符开头,且可以由任意的中英文字符、数字、句点符(.)、短横线(-)、下画线(_)等组成。

③ 实体内容:表示通过实体名所引用的具体实体内容。内容是一串包含在半角引号内的连续字符,并且不能包含“&”和“%”字符。

实体定义完成后,就需要使用相应的方式在 XML 或 DTD 中引用实体。在文档中引用内部通用实体时,需要在实体名前添加“&”符号,在实体名后添加“;”符号,因此实体引用的语法格式为:

```
&实体名;
```

下面通过几个具体实例来了解内部通用实体的用法。

文件 3-5-2-1.xml

```
<?xml version="1.0" encoding="gb2312"?>
<!DOCTYPE 会员[
    <!ELEMENT 会员 (姓名,电话,联系方式)>
    <!ELEMENT 姓名 (#PCDATA)>
    <!ELEMENT 电话 (#PCDATA)>
    <!ELEMENT 联系方式 (#PCDATA)>
```

```
    <!ENTITY 电话 "手机号码为'13233339999'">
    <!ENTITY 联系方式 "&电话; &amp; E-mail 为'zhanghong@sina.com'">
]>
<会员>
    <姓名>张红</姓名>
    <电话>&电话;</电话>
    <联系方式>&联系方式;</联系方式>
</会员>
```

使用实体后，文件 3-5-2-1.xml 在 Altova XMLSpy 中显示的效果如图 3-4 所示。

图 3-4　内部通用实体显示效果

在文件 3-5-2-1.xml 中，“<! ENTITY 电话 "手机号码为 '13233339999'">”定义实体名为“电话”，实体内容为“手机号码为 '13233339999'”，在对应的 XML 中，使用实体引用“&电话；”将实体内容显示在浏览器中，这是比较典型的在 DTD 中定义实体、在 XML 中引用实体的方式。而程序段“<! ENTITY 联系方式 "&电话; & EMAIL 为 'zhanghong@sina.com'">”定义实体名为“联系方式”，实体内容为“&电话; & E-Mail 为'zhanghong@sina.com'”，可以看出，在实体内容中，出现了实体引用“&电话;”，这是在 DTD 中定义实体、在 DTD 中引用实体的一种方式。下面再通过一个实例加深对实体的理解。

文件 3-5-2-2.xml

```
<?xml version = "1.0" encoding = "GB2312" standalone = "yes"?>
<!DOCTYPE 学生列表 [
    <!ELEMENT 学生列表 (学校, 分院, 学生 * )>
    <!ELEMENT 学校 (#PCDATA)>
    <!ELEMENT 分院 (#PCDATA)>
    <!ELEMENT 学生 (#PCDATA)>
    <!ENTITY college "理工大学">
    <!ENTITY department "&college;信息工程分院">
    <!ATTLIST 学生 学号 ID #REQUIRED
                性别 (男|女) #REQUIRED >
]>
<学生列表>
    <学校>&college;</学校>
    <分院>&department;学生名单</分院>
    <学生 学号 = "A10301101" 性别 = "男">张宏</学生>
    <学生 学号 = "A10301102" 性别 = "女">李娜</学生>
```

```
    <!-- 其他学生信息 -->
</学生列表>
```

使用实体后,文件 3-5-2-2.xml 在 Altova XMLSpy 中显示的效果如图 3-5 所示。

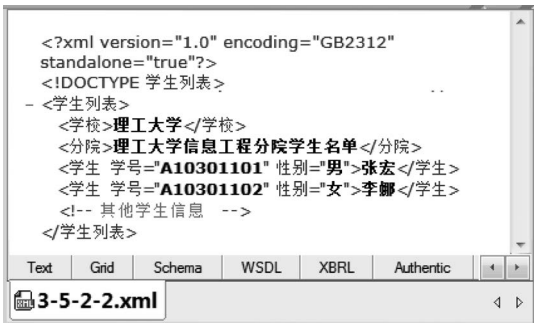

图 3-5 内部通用实体显示效果

由图 3-4 和 3-5 可以看出,DTD 中定义的实体在 XML 中引用后,所对应的文档已经将实体内容取代。因此,使用内部通用实体的好处有以下几点。

① 实体的引用,大大提高了文档的书写效率,使得文档的外观更加简洁明了。

② 如果文档中需要多次对内容进行修改,那么只需要修改实体定义中的语句,就可修改文档中所有引用该实体的部分,因而提高了文档修改的效率。

③ 实体的引用,方便一些常用的数据进行多次引用,使得文档的准确率大大提高。

3.5.3 外部通用实体

外部通用实体是在文档实体以外定义的实体对象,它所对应的内容通常为一个独立存在的文件。也就是说,XML 通过其他 XML 文档或文档片段嵌入该 XML 文档中,并通过实体引用的方式,使解析器在对应文件的 URL 资源上找到所需要的文档或文档片段,这些 XML 文档或文档片段可以合并成一个较大的新的 XML 文档。通过外部文档的引用,外部实体具有更好的灵活性与共享性。

外部通用实体声明的语法格式为:

```
<!ENTITY 实体名 SYSTEM 实体的 URI>
```

格式说明如下。

① <!ENTITY:表示开始声明一个实体,关键字 ENTITY 必须大写。

② 实体名:表示实体的名称。该名称必须以下画线或字母开头,后面可以是字母、数字、句点符(.)、短横线(-)、下画线(_)等。

③ SYSTEM:定义外部实体关键字。

④ 实体的 URL:表示实体参考文件的地址路径,该地址可以是完整的 URL 地址,也可以是相对地址,此地址需要半角引号括起来。

在 XML 文档或者 DTD 中引用外部通用实体时,需要在实体名前添加“&”符号,在实体名后添加“;”符号,其语法格式为:

&实体名;

下面通过一个具体实例来了解外部通用实体的用法。

文件 3-5-3-1.txt

```
11228000
```

文件 3-5-3-1.xml

```
<?xml version = "1.0" encoding = "gb2312" standalone = "no"?>
<!DOCTYPE 学院信息[
<!ELEMENT 学院信息 (分院 * ,联系方式 * )>
<!ELEMENT 分院 ( # PCDATA)>
<!ELEMENT 联系方式 ( # PCDATA)>
<!ENTITY department "信息与控制分院">
<!ENTITY tel SYSTEM "3 - 5 - 3 - 1.txt">
]>
<学院信息>
    <分院> &department;</分院>
    <联系方式> &tel;</联系方式>
</学院信息>
```

文件 3-5-3-1.xml 在浏览器中显示的效果如图 3-6 所示。

图 3-6　外部通用实体显示效果

3.5.4　内部参数实体

参数实体是指所定义的实体只能出现在 DTD 文件中，其主要用途是简化 DTD 语法。内部参数实体是指在独立的外部 DTD 文档的内部定义的参数实体，内部参数实体只能用于外部 DTD，无法用于内部 DTD。这里需注意，内部通用实体的“内部”是指 XML 文档内部，内部参数实体的“内部”是指外部 DTD 文档的内部。

声明内部参数实体的语法格式为：

```
<!ENTITY % 实体名 实体内容>
```

说明如下。

① <!ENTITY：表示开始声明一个实体，关键字 ENTITY 必须大写。

② %：表示声明的是一个参数实体。

③ 实体名：表示实体的名称。注意：实体名与前面的%之间有空格。

④ 实体内容：表示实体的数据内容。其内容是一串包含在半角引号内的连续字符，并且不能包含“&”和“%”字符。

在 DTD 中引用内部参数实体时，需要在实体名前添加“%”符号，在实体名后添加“;”符号，其语法格式为：

```
%实体名;
```

注意：内部参数实体必须先定义后引用，在定义时“%”与实体名之间必须有空格隔开，引用时不需要空格。

下面通过一个实例来了解内部参数实体的用法。

文件 3-5-4-1.dtd

```
<?xml version = "1.0" encoding = "gb2312"?>
<!ELEMENT 学校信息 (教师信息,学生信息)>
<!ENTITY % personal "(姓名,性别,出生年月,联系电话)">
<!ELEMENT 教师信息 %personal;>
<!ELEMENT 学生信息 %personal;>
<!ELEMENT 姓名 (#PCDATA)>
<!ELEMENT 性别 (#PCDATA)>
<!ELEMENT 出生年月 (#PCDATA)>
<!ELEMENT 联系电话 (#PCDATA)>
```

文件 3-5-4-1.xml

```
<?xml version = "1.0" encoding = "gb2312" standalone = "no"?>
<!DOCTYPE 学校信息 SYSTEM "3-5-4-1.dtd">
<学校信息>
    <教师信息>
        <姓名>赵薇</姓名>
        <性别>女</性别>
        <出生年月>1980-01-01</出生年月>
        <联系电话>12345678</联系电话>
    </教师信息>
    <学生信息>
        <姓名>张丽</姓名>
        <性别>女</性别>
        <出生年月>1987-01-01</出生年月>
        <联系电话>87654321</联系电话>
    </学生信息>
</学校信息>
```

3.5.5 外部参数实体

外部参数实体是在外部 DTD 文档中声明的参数实体，不同的 DTD 定义语句可以根据不同的需要、不同的逻辑功能被定义为不同的外部参数实体，然后通过外部参数实体的声明将多个独立的 DTD 文档综合成一个大的 DTD 文档。外部参数实体是在文档外部定义，并且只能在 DTD 中使用的实体。

声明外部参数实体的语法格式为：

```
<!ENTITY % 实体名 SYSTEM 实体的 URL>
```

说明如下。

① <!ENTITY：表示开始声明一个实体，关键字 ENTITY 必须大写。

② %：表示声明的是一个参数实体。

③ 实体名：表示实体的名称。SYSTEM：定义外部实体关键字。注意：实体名与前面的%之间有空格。

④ 实体的 URL：表示外部参数实体参考文件的地址路径，该地址可以是完整的 URL 地址，也可以是相对地址，此地址需要半角引号括起来。

在 DTD 中引用外部参数实体时，需要在实体名前添加“%”符号，在实体名后添加“;”符号，其语法格式为：

```
%实体名;
```

注意：外部参数实体必须先定义后引用，在定义时“%”与实体名之间必须有空格分开，引用时不需要空格。

下面通过一个实例来了解外部参数实体的用法。

文件 3-5-5-1.dtd

```
<?xml version="1.0" encoding="gb2312"?>
<!ELEMENT 学院信息 (校名,分院)>
<!ELEMENT 校名 (#PCDATA)>
<!ELEMENT 分院 (#PCDATA)>
```

文件 3-5-5-2.dtd

```
<?xml version="1.0" encoding=" gb2312"?>
<!ELEMENT 教师信息 (姓名,性别,专业)>
<!ELEMENT 姓名 (#PCDATA)>
<!ELEMENT 性别 (#PCDATA)>
<!ELEMENT 专业(#PCDATA)>
```

文件 3-5-5-3.dtd

```
<?xml version="1.0" encoding="gb2312"?>
<!ELEMENT 学校简介 (学院信息,教师信息)>
<!ENTITY % school SYSTEM "3-5-5-1.dtd">
<!ENTITY % teacher SYSTEM "3-5-5-2.dtd">
%school;
%teacher;
```

文件 3-5-5-1.xml

```
<?xml version="1.0" encoding="gb2312" standalone="no"?>
<!DOCTYPE 学校简介 SYSTEM "3-5-5-3.dtd">
<学校简介>
    <学院信息>
        <校名>理工大学</校名>
        <分院>信息与控制分院</分院>
    </学院信息>
```

```
    <教师信息>
        <姓名>王鹏</姓名>
        <性别>男</性别>
        <专业>软件工程</专业>
    </教师信息>
</学校简介>
```

3.6 DTD 特性

DTD 是用来验证 XML 有效性的一种方式,根据 DTD 创建的文档可以方便地在网络中进行数据交互与共享。DTD 的出现能够有效地推动 XML 的发展,然而它也受到一些因素的限制。一般情况下,使用 DTD 文件存在以下问题。

① DTD 具有独立的语法。

② DTD 文件不符合 XML 文档的语法规则。

③ DTD 用于描述数据类型的方式过于简单。

④ DTD 不支持命名空间。

⑤ DTD 扩展机制较弱。

由于 DTD 存在的不足之处,W3C 推出了另一种用于规范和约束 XML 文档的标准——XML Schema。与 DTD 不同的是,XML Schema 采用和 XML 相同的语法规则,并且在语法定义上,比 DTD 更为强大、扩展性更好。具体参见第 4 章。

3.7 小结

DTD(文档类型定义)是有效的 XML 文档的基础,主要用于规范和约定 XML 文档,目的就是让符合规范的 XML 文档成为数据交换的标准。

在 DTD 中,根据 XML 文档结构,具体规定引用该 DTD 的 XML 文档可以使用的元素名称、元素间嵌套关系、各个元素出现的先后顺序、属性和属性值数据类型以及可使用的实体及符号规则的特性。根据 DTD 建立相应的 XML 文档,就可以方便地使用 XML 文档进行数据交互。

在 DTD 中有 3 种引用方式,分别是内部 DTD 引用方式、外部 DTD 引用方式和混合 DTD 引用方式。

在 DTD 中使用关键字 ELEMENT 定义元素,使用关键字 ATTLIST 定义属性列表,使用关键字 ENTITY 定义实体。

DTD 在使用中存在一定的局限性,如语法与 XML 完全不同、数据类型简单且扩展性较弱。

3.8 习题

1. 选择题

(1) XML 文档如下:

```
<?xml version = "1.0"?>
    <! DOCTYPE greeting [
        < ELEMENT greeting ( # PCDATA)>
]>
< greeting >
    Hello, World!
</greeting >
```

上面的 XML 文档属于(　　)文档。

A. 无效的　　B. 有效的　　C. 格式良好的　　D. 格式错误的

(2) 在 DTD 中,设定一个元素可以出现任意次,则使用的符号为(　　)。

A. ?　　B. *　　C. !　　D. +

(3) 下列(　　)是引用通用实体的正确方法。

A. &RefEntity;　　B. %RefEntity;　　C. @RefEntity;　　D. !RefEntity;

(4) 下列(　　)是引用参数实体的正确方法。

A. &RefEntity;　　B. %RefEntity;　　C. @RefEntity;　　D. !RefEntity;

(5) 在 XML 文档的 DTD 机制中,(　　)最适合模仿关系型数据的主键和外键的关系。

A. key/keyref　　B. ID/IDREF

C. CDATA/PCADTA　　D. ENTITY

2. 填空题

(1) 有效的 XML 文档为<图书 类别="计算机"> XML 基础教程</图书>,其中属性“类别”的属性值可有可无,使用 DTD 定义属性的语法为:

_____。

(2) 已知通用实体<! ENTITY　content "客户信息">,写出在文档中引用该实体的语法:_____。

(3) 引用外部 DTD 文件时,需要在 XML 文档的中声明所要使用的 DTD 文件。假如 XML 文档的根元素为 root,外部 DTD 的文件名为 Filename.dtd,那么引用该外部 DTD 的语法为:

_____。

(4) 在 XML 中对应的_____是一个定义好的数据或数据集合,通过相应的引用方式,将这些数据或数据集合引入 XML 或者 DTD 所需的地方。

(5) 空元素是指在标记间没有任何数据内容的元素。空元素的存在不影响 XML 数据的正确性,空元素可以存放属性提供的额外信息。在 DTD 中使用关键字_____定义空元素。

3. 简答题

(1) 什么是 DTD? 它有什么作用?

(2) 在 DTD 中如何声明元素?

(3) 在 DTD 中如何声明属性?

(4) 在 DTD 中,使用哪些符号控制子元素出现的次数?

(5) 什么是实体? 如何分类? 各有什么特点?

4. 上机操作

根据以下的 XML 文档,写出相应的外部 DTD,并验证文档的有效性。

```
<?xml version = "1.0" encoding = "GB2312"?>
<图书列表>
    <图书信息 类别 = "计算机类">
        < ISBN > 7 - 302 - 12066 - 8 </ISBN >
        <书名> Java 面向对象程序设计</书名>
        <作者列表>
            <作者>王岩</作者>
            <作者>杨柯</作者>
        </作者列表>
        <出版社>大连理工大学出版社</出版社>
        <价格> 32.00 </价格>
    </图书信息>
    <图书信息 类别 = "儿童教育类">
        < ISBN > 7 - 303 - 1978 </ISBN >
        <书名>宝宝睡前故事</书名>
        <作者列表>
            <作者>张丽</作者>
        </作者列表>
        <出版社>海豚教育出版社</出版社>
        <价格> 12.00 </价格>
    </图书信息>
    <图书信息>
        < ISBN > 7 - 5037 - 3444 </ISBN >
        <书名>经济管理学</书名>
        <出版社>商务出版社</出版社>
        <价格> 19.00 </价格>
    </图书信息>
</图书列表>
```