第3章 chapter 3

XML Schema

本章学习目标

- 了解 XML Schema 的概念和定义内容
- 掌握 XML Schema 的文档结构
- 掌握 XML Schema 的数据类型
- 掌握 XML Schema 的元素定义
- 掌握 XML Schema 的属性定义
- 熟悉 XML Schema 模式重用的方式

本章首先介绍 XML Schema 的概念及定义内容，然后详细介绍 XML Schema 的文档结构及使用方法，最后介绍实现 XML Schema 模式重用的两种方式。

3.1 XML Schema 概述

XML Schema 是 2001 年 5 月正式发布的 W3C 推荐标准，现在已成为全球公认的 XML 环境下首选的数据建模工具。XML Schema 是 Microsoft 公司开发的一种定义 XML 文档的模式，称为 XML 模式定义语言——XML Schema Definition，简称 XSD。使用 XSD，我们不仅可以描述 XML 文档的结构以便颁布业务标准还可以使用支持 XSD 的通用化 XML 解析器对 XML 文档进行解析，并自动地检查其是否满足给定的业务标准。

XML Schema 的优势如下。

(1) XML Schema 基于 XML 语法。XML Schema 没有专门的语法，它是基于 XML 语法进行编写的。使用 XML 编写 XML Schema 有很多优势，如下所示：

- 不必再重新学习新的语言；
- 可以使用 XML 编辑器来编辑 XML Schema；
- 可以使用 XML 解析器来解析 XML Schema；
- 可以使用 XML DOM 来处理 XML Schema；

- 可以通过 XSLT 来转换 XML Schema。

(2) XML Schema 支持数据类型。XML Schema 最重要的功能之一就是对数据类型的支持,例如 int、float、boolean 等,通过对数据类型的支持带来如下优势:

- 可以更容易地描述允许的文档内容;
- 可以更容易地验证数据的正确性;
- 可以更容易地与来自数据库中的数据一并工作;
- 可以更容易地定义数据约束;
- 可以更容易地定义数据模型;
- 可以更容易地在不同的数据类型间转换数据。

(3) XML Schema 可以保护数据通信。当发送方将数据发送到接收方时,双方都应该有关于内容的相同的"期望值",通过 XML Schema,发送方可以用一种接收方能够明白的方式来描述数据。例如"02-04-2018",在某些国家被解释为 2018 年 4 月 2 日,而在另一些国家则被当作 2018 年 2 月 4 日。但是对于一个带有数据类型的 XML 元素,例如 <xs:element name="date" type="xs:date"/>,<date>2018-02-04</date>,可以确保对内容一致的理解,这是因为 XML 的数据类型 date 要求的是"YYYY-MM-DD"格式。

(4) XML Schema 是可扩展的。由于 XML Schema 是由 XML 编写的,因此 XML Schema 是可扩展的。通过可扩展的 XML Schema 定义,开发者可以在其他 XML Schema 文档中重复使用自己的 XML Schema,创建由标准数据类型派生出来的自己的数据类型,在相同的文档中引用多重的 XML Schema 文档。

(5) XML Schema 支持综合命名空间。

(6) XML Schema 支持属性组。

3.2 XML Schema 语法简介

3.2.1 XML Schema 文档结构

XML Schema 本身是一个 XML 文档,它符合 XML 语法结构,可以使用通用的 XML 解析器进行解析。一个 XML 文档引用一个 XML Schema 文档,该 XML Schema 文档定义了一个模式,遵循某个特定 XML Schema 模式的 XML 文档称为 XML Schema 的一个实例文档。

【例 3.1】 创建一个 XML Schema 文档,然后创建一个该文档的实例文档。

(1) 首先创建一个 XML Schema 文档,并且命名为 student.xsd,代码内容如下:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.mxxxb.com" xmlns="http://www.mxxxb.com"
elementFormDefault="qualified">
    <xs:element name="student">
```

```
        <xs:complexType>
            <xs:sequence>
                <xs:element name="name" type="xs:string"></xs:element>
                <xs:element name="age" type="xs:int"></xs:element>
                <xs:element name="birthday" type="xs:date"></xs:element>
                <xs:element name="score" type="xs:double"></xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

通过上述实例可以看出 XML Schema 自身就是一个 XML 文档。XML Schema 是用一套预先规定的 XML 元素和属性创建的，这些元素和属性定义了文档的结构和内容模式。＜schema＞元素是每一个 XML Schema 的根元素，其一般形式如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.wxxx3.org/2001/XMLSchema"
targetNamespace="http://www.mxxxb.com" xmlns="http://www.mxxxb.com
elementFormDefault"="qualified">
    <!--schema 文档中其他子元素的定义-->
</xs:schema>
```

其中，

- xmlns:xs="http://www.wxxx3.org/2001/XMLSchema"指出 schema 中用到的来自命名空间 "http://www.wxxx3.org/2001/XMLSchema" 的元素和数据类型应该使用前缀 xs；
- targetNamespace="http://www.mxxxb.com"指出本文档定义的元素、属性、类型等名称属于"http://www.mxxxb.com"命名空间；
- xmlns ="http://www.mxxxb.com "指出默认的命名空间是"http://www.myweb.com"；
- elementFormDefault="qualified"指出任何 XML 实例文档所使用的且在此 schema 中声明过的元素必须被命名空间限定。

(2) 创建 XML Schema 的一个实例文档，名称为 student.xml，在这个文档中引用 XML Schema 文档，代码内容如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<student xmlns="http://www.mxxxb.com"
xmlns:xsi="http://www.wxxx3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.mxxxb.com student.xsd">
    <name>tom</name>
    <age>20</age>
    <birthday>1999-03-25</birthday>
    <score>87.8</score>
```

```
</student>
```

要验证XML文档，必须指定Schema文档的位置。Schema文档的位置可以利用带有名称空间模式的xsi:schemaLocation属性以及不带名称空间模式的xsi:noNamespaceSchemaLocation属性来指定。

当Schema文档包括targetNamespace属性时，应当通过XML文档根元素的xsi:schemaLocation属性来引用Schema文档，这个属性值包括由空格分开的两部分，前一部分是URI，这个URI与Schema文档的targetNamespace属性的URI是一致的；后一部分是Schema文档的完整路径及名称。另外，XML文档的根元素也必须声明Schema文档的名称空间(xmlns:xs="http://www.wxxx3.org/2001/XMLSchema)来引用XML Schema文档，文档代码如上例所示。

当Schema文档不包括targetNamespace属性时，应当通过XML文档根元素的xsi:noNamespaceSchemaLocation属性及W3C的Schema文档的名称空间(xmlns:xs="http://www.wxxx3.org/2001/XMLSchema)来引用XML Schema文档，针对上面的实例代码修改如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<student xmlns:xsi="http://www.wxxx3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="student.xsd">
    <name>Tom</name>
    <sex>male</sex>
    <age>19</age>
    <phoneno>88889999</phoneno>
</student>
```

3.2.2 XML Schema元素的声明

1. <schema>根元素

XML Schema文档必须要定义一个且只能定义一个<schema>根元素。<schema>根元素不但表明了文档类型，而且还包括了模式的约束、XML模式名称空间的定义、其他名称空间的定义等一些其他属性信息。其定义格式如下：

```
<xs:schema xmlns:xs="http://www.wxxx3.org/2001/XMLSchema"
           targetNamespace="URI"
           elementFormDefault="qualified|unqualified"
           attributeFormDefault="qualified|unqualified">
```

2. <element>元素

在W3C XML Schema中，元素通过使用<element>元素实现。元素声明用于给元素指定元素名称、元素内容和元素数据类型等属性。元素声明的基本语法如下所示：

```
<xs:element name="元素名称"
            type="数据类型"
            default="缺省值"
            ref="EName"
            minOccurs="nonNegativeInteger"
            maxOccurs="nonNegativeInteger|unbounded"/>
```

其中，

- name 指定要声明的元素的名称；
- type 指定该元素的数据类型；
- default 指定该元素的缺省值，此项可选；
- ref 指定使用引用的元素的名称，此项可选；
- minOccurs 指定该元素在 XML 文档中可以出现的最小次数，默认为 1，它的值是一个大于或等于 0 的整数，此项可选；
- maxOccurs 指定该元素在 XML 文档中可以出现的最大次数，默认为 1，它的值是一个大于或等于 0 的整数，可以将属性的值设置为 unbounded，表示对该元素出现的最多次数没有限制，此项可选。

3.2.3 XML Schema 数据类型

XML Schema 中的数据类型分为简单类型、复合类型和匿名类型。简单类型的元素是只能包含文本，但不能包含子元素或属性的元素；复合类型的元素是可以含有子元素或属性的元素；匿名类型在定义元素时，不必写 type 属性，即可以通过元素中是否包含 type 属性判断是否为匿名元素。

1. 简单类型

XML Schema 规范中定义了两类简单类型：内置类型和用户定义类型。

(1) 内置数据类型

XML Schema 中的内置数据类型，包括原始数据类型和派生数据类型，这些类型是在 XML Schema 中使用的每种数据类型的最基本构成块，可以用它们来描述元素的内容和属性值，也可以根据这些类型构造自定义类型。XML Schema 支持的常用内置数据类型如表 3.1 所示。

表 3.1 内置数据类型

数 据 类 型	描　　述	示　　例
string	XML 中任何合法的字符串	John Smith
boolean	逻辑判断，true 和 false	true、false
number	表示任意精度的十进制数，可使用缩写	−1.52，45，1.2E2
float	单精度浮点数	123.456

续表

数据类型	描　述	示　例
double	双精度浮点数	123.456
decimal	表示任意精度的十进制数	123456
long	表示－263～＋263－1的整数值	123456
int	表示－231～＋231－1的整数值	123456
nonNegativeInteger	表示大于或等于0的整数	123
positiveInteger	表示一个大于0的整数	123
dateTime	表示格式为YYYY-MM-DDThh:mm:ss的日期时间	2017-09-03T16:21:45
time	表示hh:mm:ss格式的时间	16:21:45
date	表示YYYY-MM-DD格式的日期	2017-09-03

(2) 用户定义类型

除了上述内置数据类型之外,还有一类简单数据类型是用户自定义类型。这种数据类型是编写模式文档的用户对内置类型或其他用户自定义类型加以限制或扩展而生成的。用户自定义类型使用＜xs:simpleType＞元素定义,其语法如下所示:

```
<xs:simpleType name="自定义数据类型的名称">
    <xs:restriction base="所基于的内置数据类型的名称">
        自定义数据类型的内容模式
    </xs:restriction>
</xs:simpleType>
```

其中restriction用于为XML元素或属性定义可接受的值,其可以使用的子元素及其含义如表3.2所示。

表3.2　restriction元素的常用子元素

元素名称	描　述
enumeration	在指定的数据集中选择,限定用户的选值
fractionDigits	限定最大的小数位,用于控制精度
length	指定数据的长度
maxExclusive	指定数据的最大值(小于)
maxInclusive	指定数据的最大值(小于或等于)
maxLength	指定长度的最大值
minExclusive	指定数据的最小值(大于)
minInclusive	指定数据的最小值(大于或等于)
minLength	指定长度的最小值
pattern	指定限定数据的正则表达式

例如，记录电话号码的标签格式为：＜phoneno＞0536-7778888＜/phoneno＞，此标签中的内容要求只能容纳 12 个字符长度的字符串值，并且匹配模式 dddd-ddddddd(d 表示 0～9 的数字)，针对 phoneno 元素的自定义数据类型如下：

```
<xs:simpleType name="phoneno">
    <xs:restriction base="xs:string">
        <xs:length value="12"/>
        <xs:pattern value="\d{4}-\d{7}"/>
    </xs:restriction>
</xs:simpleType>
```

正则表达式\d{4}-\d{7}的语义为：4 个数字后面是一个连字符，接着是 7 个数字。

注意：正则表达式的详细介绍超出了本书的范围，感兴趣的读者可以参阅相关的图书或资料。

下述代码片段将名为 age 的数据类型的值设定在 16～30：

```
<xs:simpleType name="age">
        <xs:restriction base="xs:positiveInteger">
            <xs:minInclusive value="16"/>
            <xs:maxInclusive value="30"/>
        </xs:restriction>
</xs:simpleType>
```

下述代码片段将名为 sex 的数据类型的值设定为只能是 male 或 female：

```
<xs:simpleType name="sex">
    <xs:restriction base="xs:string">
        <xs:enumeration value="male"/>
        <xs:enumeration value="female"/>
    </xs:restriction>
</xs:simpleType>
```

2. 复合类型

复合类型的元素是可以含有子元素或属性的元素。为了声明复合元素，应当首先定义一个复合数据类型，然后通过使该类型与元素相关联来声明复合元素。复合数据类型的声明语法如下所示：

```
<xs:complexType name="数据类型的名称">
    <!--内容模型定义(包括子元素和属性的声明)-->
</xs:complexType>
```

在 XML 模式中，可以将相关的元素结合为组。Schema 提供了能够用来组合用户定义的元素，常用的元素有以下几个。

(1) ＜sequence＞元素

＜sequence＞元素要求组中的元素必须按照模式中指定的顺序显示。其语法格式如

下所示：

```
<xs:sequence id="ID"
             minOccurs="nonNegativeInteger"
             maxOccurs="nonNegativeInteger|unbounded">
    <!--要组合的子元素的声明-->
</xs:sequence>
```

其中，

- id 规定该元素的唯一的 ID，该项可选；
- minOccurs 规定元素在父元素中可以出现的最小次数，该值可以是大于或等于 0 的整数，若设置为 0 则表示该组是可选的，缺省值为 1，该项可选。
- maxOccurs 规定元素在父元素中可以出现的最大次数，该值可以是大于或等于 0 的整数，若设置为 unbounded 则表示不限制最大次数，缺省值为 1；该项可选。

例如：

```
<xs:element name="company" type="comType"/>
<xs:complexType name="comType">
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:element name="employee" type="xs:string"/>
        <xs:element name="manager" type="xs:string"/>
    </xs:sequence>
</xs:complexType>
```

根据上述代码的定义，XML 文档中的＜company＞元素可以包含 0 个或多个＜employee＞元素和＜manager＞元素，并且要按照先＜employee＞后＜manager＞的顺序出现。如果元素的顺序不符合它们在 sequence 声明中的顺序时，验证器会进行报错。

需要注意的是上述代码中的 minOccurs 和 maxOccurs 限定的是复合类型整体出现的次数，也就是如图 3.1 所示的 XML 文档是有效的。

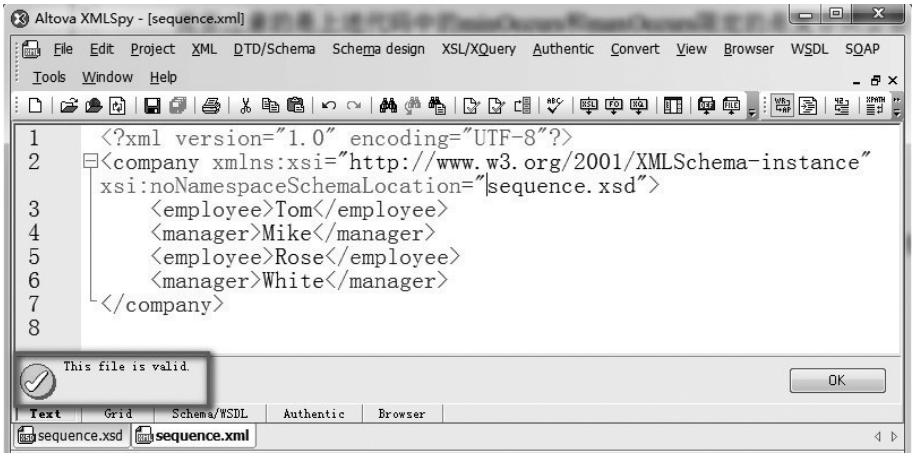

图 3.1　有效的 XML 文档演示

如图 3.2 所示的 XML 文档则是无效的。

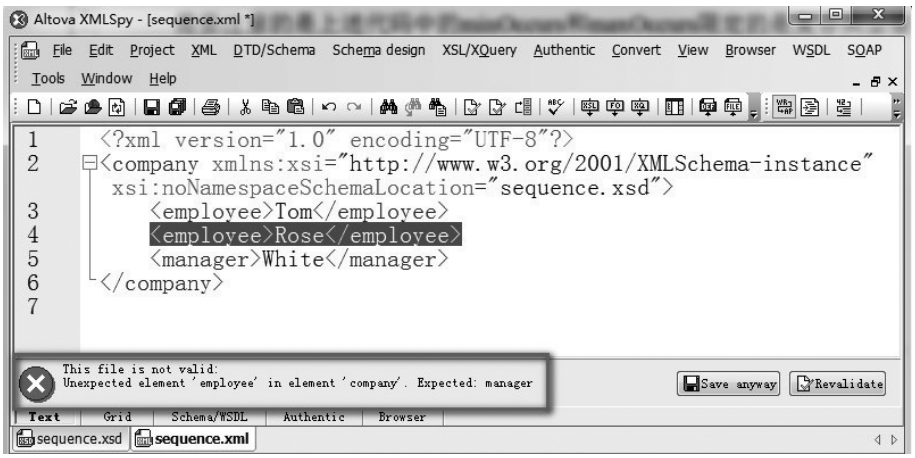

图 3.2 无效的 XML 文档演示

(2) <choise>元素

<choise>元素允许唯一的一个元素从<choise>声明组中被选择。其语法格式如下所示：

```
<xs: choice id="ID"
            minOccurs="nonNegativeInteger"
            maxOccurs="nonNegativeInteger|unbounded">
    <!--要组合的子元素的声明-->
</xs: choice>
```

其中，

- id 规定该元素的唯一的 ID。该项可选；
- minOccurs 规定元素在父元素中可以出现的最小次数，该值可以是大于或等于 0 的整数，若设置为 0 则表示该组是可选的，缺省值为 1。该项可选；
- maxOccurs 规定元素在父元素中可以出现的最大次数，该值可以是大于或等于 0 的整数，若设置为 unbounded 则表示不限制最大次数，缺省值为 1。该项可选。

例如：

```
<xs:element name="company" type="comType"/>
<xs:complexType name="comType">
    <xs:choice>
        <xs:element name="employee" type="xs:string"/>
        <xs:element name="manager" type="xs:string"/>
    </xs:choice>
</xs:complexType>
```

根据上述代码的定义，XML 文档中的<company>元素必须且只能包含一个<employee>元素或一个<manager>元素，否则，验证器会进行报错。

如图 3.3 所示的 XML 文档是有效的。

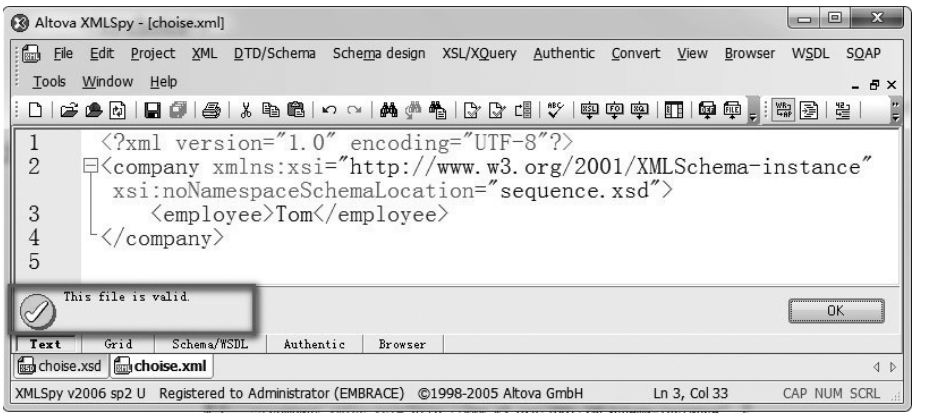

图 3.3　有效的 XML 文档演示

如图 3.4 所示的 XML 文档是无效的。

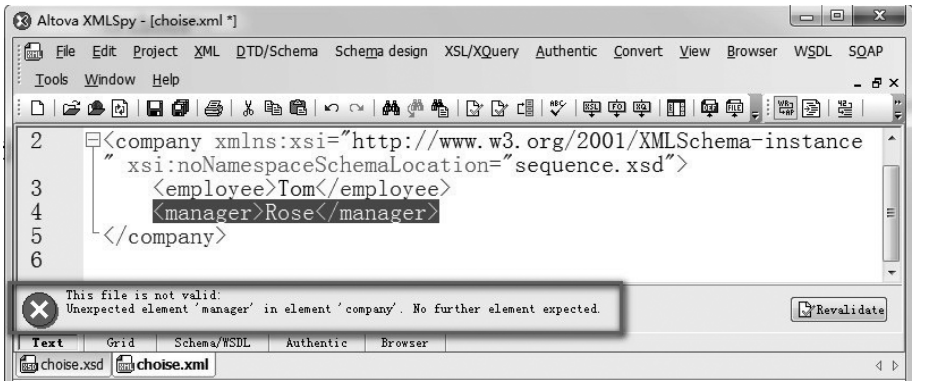

图 3.4　无效的 XML 文档演示

(3) <all>元素

<all>元素规定子元素能够以任意顺序出现，每个元素可以出现 0 次或 1 次，而且每次最多显示一次。其语法格式如下所示：

```
<xs: all id="ID"
          minOccurs="0|1"
          maxOccurs="1">
   <!--要组合的子元素的声明-->
</xs:all>
```

其中，

- id 规定该元素的唯一的 ID，该项可选；
- minOccurs 规定元素在父元素中可出现的最小次数，该值可以是 0 或 1，若设置为 0 则表示该组是可选的，缺省值为 1，该项可选；

- maxOccurs 规定元素在父元素中可出现的最大次数，该值必须是 1。该项可选。

例如：

```
<xs:element name="company" type="comType"/>
<xs:complexType name="comType">
    <xs:all>
        <xs:element name="employee" type="xs:string"/>
        <xs:element name="manager" type="xs:string"/>
    </xs:all>
</xs:complexType>
```

根据上述代码的定义，在 XML 文档中<employee>元素和<manager>元素能够以任何顺序出现，两个元素都必须且只能出现一次，否则，验证器会进行报错。

如图 3.5 所示的 XML 文档是有效的。

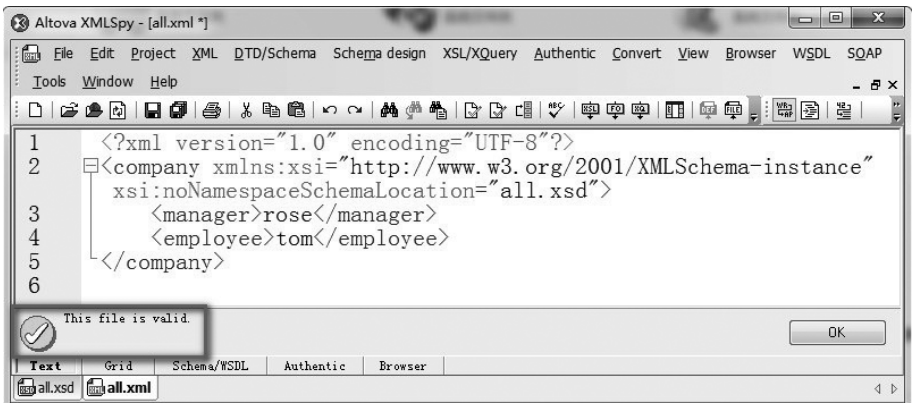

图 3.5 有效的 XML 文档演示

如图 3.6 所示的 XML 文档是无效的。

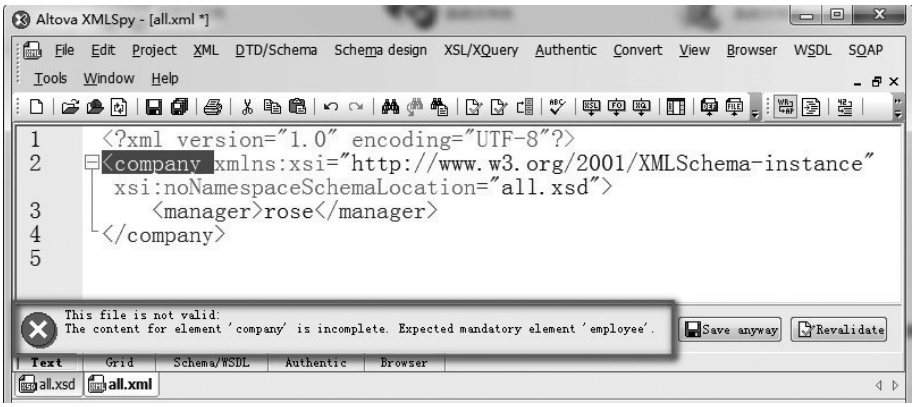

图 3.6 无效的 XML 文档演示

【例 3.2】 根据给定 XML 文档(student.xml)以及相关要求创建相应的 XML Schema 文档(student.xsd)。

```
<?xml version="1.0" encoding="UTF-8"?>
<student xmlns:xsi="http://www.wxxx3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="student.xsd">
    <name>Tom</name>
    <sex>male</sex>
    <age>19</age>
    <phoneno>0536-8888777</phoneno>
</student>
```

此文档中元素<name>元素：<sex>元素、<age>元素和<phoneno>元素的取值需满足如下要求：

- <name>、<sex>、<age>和<phoneno>是简单类型的元素；
- <name>元素和<sex>元素的值必须是字符串；
- <sex>元素的取值只能是 male 和 female；
- <age>元素的值必须是 16～30 的整数；
- <phoneno>元素允许的组合：11 位手机号、3 位区号-8 位号码、4 位区号-8 位号码、4 位区号-7 位号码。

针对上述要求，该 XML 的完整模式文档定义如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.wxxx3.org/2001/XMLSchema" elementFormDefault=
"qualified" attributeFormDefault="unqualified">
    <xs:element name="student" type="studentDef"/>
    <!--定义 studentDef 复合类型-->
    <xs:complexType name="studentDef">
        <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="sex" type="sexDef"/>
            <xs:element name="age" type="ageDef"/>
            <xs:element name="phoneno" type="phonenoDef"/>
        </xs:sequence>
    </xs:complexType>
    <!--定义 ageDef 简单类型-->
    <xs:simpleType name="ageDef">
        <xs:restriction base="xs:positiveInteger">
            <xs:minInclusive value="16"/>
            <xs:maxInclusive value="30"/>
        </xs:restriction>
    </xs:simpleType>
    <!--定义 sexDef 简单类型-->
    <xs:simpleType name="sexDef">
```

```
        <xs:restriction base="xs:string">
            <xs:enumeration value="male"/>
            <xs:enumeration value="female"/>
        </xs:restriction>
    </xs:simpleType>
    <!--定义 phonenoDef 简单类型-->
    <xs:simpleType name="phonenoDef">
        <xs:restriction base="xs:string">
            <xs:pattern value="\d{11}|\d{3}-\d{8}|\d{4}-\d{8}|\d{4}-\d{8}"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

在上述定义中,由于 XML 文档<student>元素为复合类型,故在 Schema 中为其指定类型时需要使用复合类型。studentDef 是按照 XML 文档结构定义的复合数据类型,被通过<student>的 type 属性指定为<student>元素的数据类型。

将<age>、<sex>和<phoneno>元素的数据类型都扩展为用户自定义数据类型,并分别命名为 ageType、sexType 和 phonenoType,此种形式可以方便地实现数据类型共用。

3. 匿名类型

使用 XML Schema,可以定义一系列具有名称的数据类型,通过使用元素的 type 属性来引用这些数据类型。这种类型的模式构造非常直观,可以方便地实现数据类型复用,但在有些情况下不是很实用。例如,在模式文档中定义了许多只应用一次并且包含非常少的约束的数据类型,使用上文介绍的方法模式,文档就会变得非常烦琐。对于此类情况,可以使用一种新的定义方式——匿名类型定义。

使用匿名类型定义时元素声明不必再写 type 属性,因为匿名类型定义在某个元素的内部,为该元素专用。

【例 3.3】 使用匿名类型定义实现例 3.2 的要求,定义一个 XML Schema 文档。

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.wxxx3.org/2001/XMLSchema" elementFormDefault=
"qualified" attributeFormDefault="unqualified">
    <xs:element name="student">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="name" type="xs:string"/>
                <xs:element name="sex">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="male"/>
                            <xs:enumeration value="female"/>
```

```
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="age">
            <xs:simpleType>
                <xs:restriction base="xs:positiveInteger">
                    <xs:minInclusive value="16"/>
                    <xs:maxExclusive value="30"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="phoneno">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:pattern
                    value="\d{11}|\d{3}-\d{8}|\d{4}-\d{8}|\d{4}-\d{7}"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
    </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

3.2.4 XML Schema 属性声明

属性声明用于命名属性并指定属性值的类型。在 Schema 中实现的方法是使用 Attribute 标记，可以按照定义元素的方式定义属性，但受限制的程度更高。它们只能是简单类型，只能包含文本，且没有子属性；属性是没有顺序的，而元素是有顺序的。其语法格式如下所示：

```
<xs:attribute name="属性名"
              default="缺省值"
              fixed="固定值"
              type="数据类型"
              use="optional|required"/>
```

其中，

- name 用来指定自定义属性名称；
- default 用来指定自定义属性的一个缺省值，该项可选；
- fixed 用来为自定义属性提供一个固定的值，且不能和 default 属性同时出现，该项可选；
- type 指定该属性的数据类型，此处只能是简单数据类型；

- use 指定该属性值是 required(必需)还是 optional(可选的),默认为 optional。

注意:要把属性附加在元素上,属性应该在 complexType 定义过程中最后进行定义。

例如,下面代码显示了给<student>元素增加一个 no(学号)属性的方法:

```
<xs:element name="student" type="studentDef"/>
    <!--定义 studentDef 复合类型-->
<xs:complexType name="studentDef">
    <xs:sequence>
        <xs:element name="name" type="xs:string"/>
        <xs:element name="sex" type="sexDef"/>
        <xs:element name="age" type="ageDef"/>
        <xs:element name="phoneno" type="phonenoDef"/>
    </xs:sequence>
    <!--定义 student 元素的 no 属性-->
     < xs: attribute  name =" no"  type =" xs: string"  use =" required" > </xs:
attribute>
</xs:complexType>
```

3.3 模式重用

XML Schema 支持高度重用性,即在一个模式中声明的组件能够被另一个模式重用。XML Schema 推荐标准中提供了<include>元素和<import>元素来实现模式的重用。

1. <include>元素

<include>元素用于向一个文档添加带有相同目标命名空间的多个 XML Schema 文档。其语法格式如下:

```
<xs:include id="ID" schemaLocation="filename"/>
```

其中,

- id 用来指定元素的 ID,ID 必须是唯一的,该项可选;
- schemaLocation 指定所要包含的 XML Schema 文档的 URI。

<include>元素在一个 xsd 文档中可以多次出现。<schema>元素是<include>元素的父元素。在使用<include>元素时,唯一要注意的事情是:要包含和已包含的模式文件必须属于同一个目标命名空间,如果 schema 目标命名空间不匹配,则包含不会有效。

【例 3.4】 定义一个 XML Schema 文档,演示如何使用<include>元素实现模式重用。

定义第一个模式文件 first.xsd,内容如下:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.wxxx3.org/2001/XMLSchema"
xmlns="http://www.mxxxb.com" targetNamespace="http://www.mxxxb.com">
    <xs:simpleType name="bid">
        <xs:restriction base="xs:string">
            <xs:pattern value="[A]\d{6}"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

定义第二个模式文件 second.xsd，内容如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.wxxx3.org/2001/XMLSchema"
xmlns="http://www.mxxxb.com" targetNamespace="http://www.mxxxb.com">
    <xs:simpleType name="aid">
        <xs:restriction base="xs:string">
            <xs:pattern value="[C]\d{6}"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

定义第三个模式文件 third.xsd，内容如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.wxxx3.org/2001/XMLSchema"
xmlns="http://www.mxxxb.com" targetNamespace="http://www.mxxxb.com">
    <xs:include schemaLocation="first.xsd"/>
    <xs:include schemaLocation="second.xsd"/>
    <xs:element name="books" type="infotype"/>
    <xs:complexType name="infotype">
        <xs:sequence>
            <xs:element name="book" type="booktype"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="booktype">
        <xs:sequence>
            <xs:element name="title" type="xs:string"/>
            <xs:element name="author" type="atype"/>
        </xs:sequence>
        <xs:attribute name="bookid" type="bid"/>
    </xs:complexType>
    <xs:complexType name="atype">
        <xs:sequence>
            <xs:element name="firstname" type="xs:string"/>
            <xs:element name="lastname" type="xs:string"/>
```

```
        </xs:sequence>
        <xs:attribute name="authorid" type="aid"/>
    </xs:complexType>
</xs:schema>
```

在 third.xsd 模式文件中使用了前面两个模式文件中所定义的用户自定义的简单数据类型，并且它们位于同一个目标命名空间：targetNamespace="http://www.mxxxb.com"。对于命名空间"http://www.mxxxb.com"中定义的类型的引用不需要加前缀，因为此命名空间使用的是默认引用的方式；对于命名空间"http://www.wxxx3.org/2001/XMLSchema"中定义的类型的引用必须加上前缀 xs。

2. <import>元素

<import>元素和<include>元素具有同样的功能，但<import>元素允许访问来自多个不同目标名称空间的模式的组件。其语法格式如下：

```
<import id="ID" namespace="namespace" schemaLocation="filename"/>
```

其中，

- id 用来指定元素的 ID，ID 必须是唯一的，该项可选；
- namespace 指定被引入模式所属的命名空间 URI，它也指定前缀，该前缀用来使一个元素或属性和一个特定的命名空间相关联；
- schemaLocation 指定模式文件的物理地址。

【例 3.5】 定义一个 XML Schema 文档，演示如何使用<import>元素实现模式重用。

定义第一个模式文件 first_i.xsd，该 XSD 文件对于命名空间使用默认方式，目标命名空间：targetNamespace="http://www.xxx.com"，内容如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.wxxx3.org/2001/XMLSchema"
xmlns="http://www.xxxt.com" targetNamespace="http://www.xxxt.com">
    <xs:simpleType name="bid">
        <xs:restriction base="xs:string">
            <xs:pattern value="[A]\d{6}"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

定义第二个模式文件 second_i.xsd，该 XSD 文件对于命名空间使用默认方式，目标命名空间：targetNamespace="http://www.xxxd.com"，内容如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.wxxx3.org/2001/XMLSchema"
xmlns="http://www.xxxd.com" targetNamespace="http://www.xxxd.com">
```

```
    <xs:simpleType name="aid">
        <xs:restriction base="xs:string">
            <xs:pattern value="[c]\d{6}"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

定义第三个模式文件 third_i.xsd，该 XSD 文件目标命名空间 targetNamespace＝"http://www.xxxd.com"，内容如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.wxxx3.org/2001/XMLSchema"
xmlns="http:/www.mxxxb.com" targetNamespace="http:/www.mxxxb.com" xmlns:
prd1="http://www.xxxt.com" xmlns:prd2="http://www.xxxd.com">
    <xs:import namespace="http://www.xxxt.com" schemaLocation="first_i.xsd"/>
    <xs:import namespace="http://www.xxxd.com" schemaLocation="second_i.xsd"/>
    <xs:element name="books" type="infotype"/>
    <xs:complexType name="infotype">
        <xs:sequence>
            <xs:element name="book" type="booktype"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="booktype">
        <xs:sequence>
            <xs:element name="title" type="xs:string"/>
            <xs:element name="author" type="atype"/>
        </xs:sequence>
        <xs:attribute name="bookid" type="prd1:bid"/>
    </xs:complexType>
    <xs:complexType name="atype">
        <xs:sequence>
            <xs:element name="firstname" type="xs:string"/>
            <xs:element name="lastname" type="xs:string"/>
        </xs:sequence>
        <xs:attribute name="authorid" type="prd2:aid"/>
    </xs:complexType>
</xs:schema>
```

在 third_i.xsd 模式文件中使用了前面两个模式文件中所定义的用户自定义的简单数据类型，因为它们并不位于同一个目标命名空间，所以需要借助于＜import＞元素将其他目标命名空间中的模式文件引入当前的模式文件中。

对于命名空间"http://www.xxxt.com"中所定义的类型的引用必须加上前缀 prd1；对于命名空间"http://www.xxxd.com"中所定义的类型的引用必须加上前缀 prd2；对于

命名空间"http://www.wxxx3.org/2001/XMLSchema"中所定义的类型的引用必须加上前缀 xs;而对于命名空间"http://www.mxxxb.com"中所定义的类型的引用就不需要加前缀了,因为此命名空间使用的是默认引用的方式。

3.4 XML Schema 应用实例

【例 3.6】 对于下列 XML 文档:

```
order.xml
<?xml version="1.0" encoding="UTF-8"?>
<orders>
    <order orderID="A001" orderDate="2017-09-12">
        <name>玩具</name>
        <number>16</number>
        <city>上海</city>
        <zip>200000</zip>
        <phoneno>13577778888</phoneno>
    </order>
    <order orderID="A002" orderDate="2017-09-16">
        <name>文具</name>
        <number>20</number>
        <city>青岛</city>
        <zip>266000</zip>
        <phoneno>0532-77778888</phoneno>
    </order>
</orders>
```

创建一个 Schema 文档,并应用于给定的 XML 文档。要求如下:

- <order>元素在 XML 文档中可以出现多次,但至少要求出现一次;
- <orderID>元素值的格式必须是 AXXX,其中 X 为 0～9 的数字,为必选项;
- <number>元素的值为 1～999;
- <zip>元素的内容的格式必须是 XXXXXX,其中 X 为 0～9 的数字,该元素可选;
- <phoneno>元素允许如下组合:11 位手机号、3 位区号+8 位号码、4 位区号+8 位号码、4 位区号+7 位号码。

1. 建立 XML Schema 文档

(1) 单击 File|New 按钮,在弹出的对话框中选择 W3C XML Schema 选项。

(2) 首先定义 XML 文档的根元素<orders>,该元素拥有两个<order>子元素,因此将它认定为复合类型元素,并且可以被<sequence>元素修饰,用作定义子元素出现的次数。其次<order>元素拥有两个属性和若个干子元素,因此也将它认定为复合类型元素,并且用<sequence>元素修饰,用作定义子元素的出现顺序。<name>、<number>、

<zip>、<phoneno>这几个元素只包含文本信息，因此可以将其定义为简单类型元素。相关代码如下所示：

order.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.wxxx3.org/2001/XMLSchema"
elementFormDefault="qualified">
    <xs:element name="orders">
        <xs:complexType>
            <xs:sequence maxOccurs="unbounded">
                <xs:element name="order" type="OrderType"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:complexType name="OrderType">
        <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="number" type="numberType"/>
            <xs:element name="city" type="xs:string"/>
            <xs:element name="zip" type="zipType" minOccurs="0"/>
            <xs:element name="phoneno" type="phonenoType"/>
        </xs:sequence>
        <xs:attribute name="orderID" type="orderIDType" use="required"/>
        <xs:attribute name="orderDate" type="xs:date"/>
    </xs:complexType>
    <xs:simpleType name="numberType">
        <xs:restriction base="xs:int">
            <xs:minInclusive value="1"/>
            <xs:maxInclusive value="999"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="zipType">
        <xs:restriction base="xs:string">
            <xs:length value="6"/>
            <xs:pattern value="\d{6}"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="phonenoType">
        <xs:restriction base="xs:string">
            <xs:pattern value="\d{11}|\d{3}-\d{8}|\d{4}-\d{8}"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="orderIDType">
        <xs:restriction base="xs:string">
            <xs:length value="4"/>
            <xs:pattern value="A\d{3}"/>
        </xs:restriction>
```