XML基础学习

# XML语言基础

## XML的作用：传输数据

## XML文件的规则

元素必须配对：<name>…</name>

元素名称区分大小写。

每个xml文档有且只有一个根元素

Xml必须正确嵌套

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <books>  <book>  <name>JAVA自学手册</name>  <publisher>人民邮电出版社</publisher>  <author>张小侠</author>  <ISBN>9787115201027</ISBN>  <price unit="RMB">59.80</price>  </book>  </books> |

## Xml格式

注释：<!--注释内容-->

元素：<name></name>

属性：<price unit="RMB">59.80</price>

特殊字符显示：<![CDATA[ … ]]>。不应由 XML 解析器进行解析的文本数据。在 XML 元素中，"<" 和 "&" 是非法的。"<" 会产生错误，因为解析器会把该字符解释为新元素的开始。

特殊字符的转义：

|  |  |  |
| --- | --- | --- |
| 实体引用 | 符号 | 中文释义 |
| &lt; | < | 小于 |
| &gt; | > | 大于 |
| &amp; | & | 和号 |
| &apos; | ‘ | 单引号 |
| &quot; | “ | 引号 |

注意：xml可以通过浏览器打开来验证。

## Xml和CSS

XML可以通过CSS来修饰，通过引入CSS样式表：

<?xml-stylesheet type="text/css" href="样式表的相对/绝对路径"?>

# XML与XSLT

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">  <xsl:template match="/">  <html>  <body>  <xsl:element name="h2">my love</xsl:element>  <xsl:for-each select="books/book">  <p><xsl:value-of select="name"/><a href="#{generate-id(name)}">查看详情</a></p>  </xsl:for-each>  <table border="1">  <tr bgcolor="#55A0FF">  <th>名称</th>  <th>出版社</th>  <th>公司</th>  <th>作者</th>  <th>ISBN</th>  <th>价格</th>  <th>网站URL</th>  </tr>    <xsl:for-each select="books/book">  <xsl:sort select="price"/>  <xsl:if test="author='张小侠'">  <tr bgcolor="#7D7DFF">  <td><xsl:value-of select="name"/></td>  <td><xsl:value-of select="publisher"/></td>  <td><xsl:value-of select="company"/></td>  <td><xsl:value-of select="author"/></td>  <td><xsl:value-of select="ISBN"/></td>  <td><xsl:value-of select='format-number(price,"###,###.00")'/></td>  <td><xsl:value-of select="url"/></td>  </tr>  </xsl:if>  </xsl:for-each>  </table>  </body>  </html>  </xsl:template>  </xsl:stylesheet> |

SLT是 扩展样式表转换语言 的外语缩写，这是一种对XML（标准通用标记语言的子集）文档进行转化的语言，XSLT中的T代表英语中的"转换"（Transformation）。它是XSL（eXtensible Stylesheet Language）规范的一部分。

## XSLT的元素

XSLT转换XML时，XSLT中显示的元素与其在XML中的位置无关，还可以进行取舍。

### 声明XSLT

|  |
| --- |
| <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">或者  <xsl:transform version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"> |

### XSLT中的模板

<xsl:template match="/">

</xsl:template>

参数说明：match里面的路径就是XML元素的级别

### value-of

<xsl:value-of select="name"/>

参数说明：select获取节点的内容

### for-each

<xsl:for-each select="books/book">

</xsl:for-each>

参数说明：select获取要遍历的元素

### if

<xsl:if test="author='张小侠'">

</xsl:if>

参数说明：test中是判断条件

### choose

<xsl:choose>

<xsl:when test="条件">

…

</xsl:when>

<xsl:otherwise>

…

</xsl:otherwise>

</xsl:choose>

### sort

<xsl:sort select="price" order="desceding/descending"case-order="upper-first/lower-first"/>

### copy-of

XSLT的variable可以为一部分变量定义变量，同时使用copy-of对变量的内容进行复制。

<xsl:variable name="title">

…

</xsl:variable>

<xsl:copy-of select=$title"/>

Select中也可以填写XML元素。

### apply-template定义模板

使用<apply-templates select=""/>引入模板

<apply-template match="name"/>创建模板

### attribute

<xsl:attribute name="title">…属性</xsl:attribute>

### element

<xsl:element name="h2(元素名称)">…</xsl:element>

## XSLT内建函数

### generate-id(name)

参数说明：name表示XML元素名称，相同参数产生的值相同。

### format-number(元素值，"###，###.00")

#代表数字的具体位置 .表示小数点的位置 ,表示千位的分隔符 %表示使用百分比来显示

### document(xml文档路径)

# DTD的介绍

文档类型定义(Document Type Definition)是一套关于标记符的语法规则。它是标准通用标记语言和[1] 可扩展标记语言1.0版规格的一部分，是文档的验证机制。

|  |
| --- |
| <!DOCTYPE books[  <!ELEMENT books (book+)>  <!ELEMENT book (name,publisher,company,author,ISBN,price,url)>  <!ELEMENT name (#PCDATA)>  <!ELEMENT publisher (#PCDATA)>  <!ELEMENT company (#PCDATA)>  <!ELEMENT author (#PCDATA)>  <!ELEMENT ISBN (#PCDATA)>  <!ELEMENT price (#PCDATA)>  <!ELEMENT url (#PCDATA)>  ]> |

## DTD的引入与验证

### Xml内部定义DTD

<!DOCTYPE books[

<!ELEMENT books (book+)>

]>

DOCTYPE :声明DTD关键字

ELEMENT：声明XML元素

注意：定义DTD文件的时候，注意空格。

### 引入外部DTD文件

<!DOCTYPE books SYSTEM "simple.dtd">

books是根元素，SYSTEM也可以定义为PUBLIC，定义为PUBLIC时候，引用的DTD是由权威机构制定的。

## DTD验证XML元素

### 声明元素

<!ELEMENT book (name,publisher,company,author,ISBN,price,url)>

<!ELEMENT company ANY)>：元素包含任意内容

<!ELEMENT company EMPTY)>：元素内容可以为空

### 声明重复元素

+：可以有多个子元素>=1；

\*：>=0个元素；

?：代表0或者1个元素

<!ELEMENT book (name,publisher,company,author,ISBN,price+,url)>

### 声明选择性元素

<!ELEMENT book (name,publisher,company,author,ISBN,(tel|phone)>

|：前后的两个元素在父元素中有且只能出现一次。

### 使用ENTITY

<!ENTITY info "提示信息">

## DTD验证XML属性

### 声明xml属性

<!ATTLIST price unit CDATA "RMB">

参数说明：price属性所在元素，unit表示属性，CDADA表示属性类型是字符数据，RMB表示的属性的默认值。

### 声明属性的#IMPLIED

<!ATTLIST price unit CDATA #IMPLIED >

说明：属性值可有可无，不能定义默认值

### 声明属性的#REQUIRED

<!ATTLIST price unit CDATA # REQUIRED >

说明：属性必须明确写出来，不能定义默认值。

### 声明属性的#FIXED

<!ATTLIST price unit CDATA #FIXED "RMB" >

说明：声明属性值时候必须用固定的值，不声明属性也不会报错。

### 声明DTD列举属性值

<!ATTLIST price unit (圆|角|分)"元">

说明：属性值只能使用列举值中的一个，默认值使用元

### 声明ID属性

<!ATTLIST book UID ID #REQUIRED >

说明：id属性可以作为元素的唯一标识，每个属性只能有一个ID类型的属性。

### 声明IDREF属性

一般用来引用ID类型的值，来表示两个元素之间的关系。

### 声明属性类型为NMTOKEN类型

<!ATTLIST price unit NMTOKEN #REQUIRED>

设置属性的内容的名称：字母，数字，冒号，点号等。里面存在多个属性值的时候，可以使用NMTOKENS

## DTD的缺陷

利用DTD验证有效性的解析器，就能够立即对文档的完整性进行可靠的检查。DTD虽然比较实用，但DTD也有不少的缺陷。

DTD有自己的特殊语法，其本身不是XML文档；

DTD只提供了有限的数据类型，用户无法自定义类型；

DTD不支持域名机制。

# XML Schema的基本语法

XSDL（XML Schema定义语言）由元素、属性、命名空间和XML文档种的其他节点构成的。

## XSD中的元素

XSD文档至少包含：schema根元素和XML模式命名空间的定义、元素定义。

### schema根元素语法

|  |
| --- |
| <xsd:schema xmlns:xsd=“http://www.w3.org/2001/XMLSchema”>  …  </xsd:schema>  在XSD中必须定义一个且只能定义一个schema根元素。根元素中包括模式的约束、XML模式命名空间的定义，其他命名空间的定义、版本信息、语言信息和其他一些属性。 |

### 元素

语法如下：<xsd:element name=”user” type=”xsd:string” />

XSD中的元素是利用element标识符来声明的。其中name属性是元素的名字，type属性是元素值的类型，在这里可以是XML Schema中内置的数据类型或其他类型。

|  |
| --- |
| <xsd:schema xmlns:xsd=“http://www.w3.org/2001/XMLSchema”>      <xsd:element name=”user” type=”xsd:string” />  </xsd:schema>  以上文档对应的有效XML文档如下：  <?xml version=”1.0”?>  <user>string</user> |

在元素的定义中还有2个属性：minOccurs和maxOccurs。其中minOccurs定义了该元素在父元素中出现的最少次数（默认为1，值为大于等于0的整数），maxOccurs定义了该元素在父元素中出现的最多次数（默认为1，值为大于等于0的整数）。在maxOccurs中可以把值设置为unbounded，表示对元素出现的最多次数没有限制。

|  |
| --- |
| <xsd:schema xmlns:xsd=http://www.w3.org/2001/XMLSchema>      <xsd:element name=”user” type=”xsd:string” minOccurs=”0” maxOccurs=”unbounded” />  </xsd:schema>  表示为元素user的类型为string，出现的次数最少为0（也就是可选），最多不限制。 |

### 引用元素和替代

|  |
| --- |
| <xsd:schema xmlns:xsd=”http://www.w3.org/2001/XMLSchema”>      <xsd:element name=”user” type=”xsd:string” />      <xsd:element name=”name”>          <xsd:complexType>              <xsd:sequence>                  <xsd:element ref=”user” />              </xsd:sequence>          </xsd:complexType>      </xsd:element>  </xsd:schema>  引用是利用element标记符的ref属性实现的。主要适用于避免在文档中多次定义同一个元素，应当将经常使用的元素定义为根元素的子元素，以便在文档的任何地方引用它。 |

在这里还可以为某个定义的元素起一个别名，方法如下：

|  |
| --- |
| <xsd:schema xmlns:xsd=”http://www.w3.org/2001/XMLSchema”>      <xsd:element name=”yonghu” type=”xsd:string” substitutionGroup=”user” />      <xsd:element name=”user” type=”xsd:string” />      <xsd:element name=”name”>          <xsd:complexType>              <xsd:sequence>                  <xsd:element ref=”user” />              </xsd:sequence>          </xsd:complexType>      </xsd:element>  </xsd:schema>  以上文档对应的有效XML文档如下：  <?xml version=”1.0”?>  <name>      <user>string</user>  </name>  或者：  <?xml version=”1.0”?>  <name>      <yonghu>string</yonghu>  </name>  主要是利用element标识符中的属性substitutionGroup实现的。 |

### 设置默认值和固定值

语法如下：

|  |
| --- |
| <xsd:element name=”city” type=”xsd:string” default=”xian” />  <xsd:element name=”country” type=”xsd:string” fixed=”china” /> |

通过default属性的设置，可以在XML文档中没有对city元素定义时赋予默认值。而使用fixed属性，可以给元素country设定一个固定的值china，并且不允许改变！

### 利用组合器控制结构

sequence组合器，定义了一列元素必须按照模式中指定的顺序显示（如果是可选的，也可以不显示）。语法如下：

|  |
| --- |
| <xsd:schema xmlns:xsd=”http://www.w3.org/2001/XMLSchema”>      <xsd:element name=”name”>          <xsd:complexType>              <xsd:sequence>                  <xsd:element name=”first” type=”xsd:string” />                  <xsd:element name=”middle” type=”xsd:string” />                  <xsd:element name=”last” type=”xsd:string” />             </xsd:sequence>          </xsd:complexType>      </xsd:element>  </xsd:schema> |

all组合器，允许所定义的元素可以按照任意顺序显示，all元素的子元素在默认情况下是必须的，而且每次最多显示一次。语法如下：

|  |
| --- |
| <xsd:schema xmlns:xsd=”http://www.w3.org/2001/XMLSchema”>      <xsd:element name=”name”>          <xsd:complexType>              <xsd:all minOccurs=”0”>                  <xsd:element name=”first” type=”xsd:string” />                  <xsd:element name=”middle” type=”xsd:string” />                  <xsd:element name=”last” type=”xsd:string” />              </xsd:all>          </xsd:complexType>      </xsd:element>  </xsd:schema> |

choice组合器，允许指定多组声明中的一个，用于互斥情况。语法如下：

|  |
| --- |
| <xsd:schema xmlns:xsd=”http://www.w3.org/2001/XMLSchema”>      <xsd:element name=”name”>          <xsd:complexType>              <xsd:choice>                  <xsd:element name=”first” type=”xsd:string” />                  <xsd:element name=”middle” type=”xsd:string” />                  <xsd:element name=”last” type=”xsd:string” />              </xsd:choice>          </xsd:complexType>      </xsd:element>  </xsd:schema> |

## 定义属性

在XML Schema文档中可以按照定义元素的方法定义属性，但受限制的程度较高。它们只能是简单类型，只能包含文本，且没有子属性。可以应用在attribute元素定义中的属性如下：

* default ：初始默认值
* fixed：不能修改和覆盖的属性固定值
* name：属性的名称
* ref：对前一个属性定义的引用
* type ：该属性的XSD类型或者简单类型
* use ： 如何使用属性
* form ：确定attributeFormDefault的本地值
* id：模式文档中属性唯一的ID

Default、fixed、name、ref和type属性与在element标记中定义的对应属性相同，但type只能是简单类型。Use属性的值可以是：optional（属性不是必须的，此为默认属性）、prohibited或者required（属性是强制的）。

### 创建属性

语法如下：

<xsd:attribute name=”age” type=”xsd:integer” />

该语句定义了一个名为age的属性，它的值必须是一个整数。把它添加到模式中时，它必须时schema元素、complexType元素或者attributeGroup元素的子元素。例子如下：

|  |
| --- |
| <xsd:schema xmlns:xsd=”http://www.w3.org/2001/XMLSchema”>      <xsd:element name=”name”>          <xsd:complexType>              <xsd:sequence>                  <xsd:element name=”first” type=”xsd:string” />              </xsd:sequence>              <xsd:attribute name=”age” type=”xsd:integer” use=”optional” />          </xsd:complexType>      </xsd:element>  </xsd:schema>  以上文档对应有效的XML文档如下：  <?xml version=”1.0”?>  <name age=”27”>      <first>string</first>  </name>  如上所示，要把属性附加在元素上，属性应该在complexType定义中的组合器之后定义或引用。 |

## XML Schema数据类型

### 基本数据类型

基本数据类型是在XML Schema中使用的每种数据类型的最基本构成块。可以根据这些类型构造自定义的类型。这些类型包括：

* boolean：可以是1（true）或者0（false）。
* dateTime：表示时间的部分可选，格式：CCYY-MM-DDThh:mm:ss

例如：2005-3-18T14:48:12

* decimal ：表示任意精度的十进制数字。
* string：字符数据。
* int：表示从-2,147,483,648到2,147,483,648之间一个整数。
* nonNegativeInteger：表示大于或者等于0的一个整数。
* nonPositiveInteger：表示小于或者等于0的一个整数。
* short：表示从-32768到32767之间的一个整数。

|  |
| --- |
| <xsd:schema xmlns:xsd=http://www.w3.org/2001/XMLSchema>  <xsd:element name=”ContactDetails”>          <xsd:complexType>              <xsd:sequence>                  <xsd:element ref=”name” />                  <xsd:element name=”rate” type=”xsd:decimal” />              </xsd:sequence>              <xsd:attribute name=”lastUpdated” type=”xsd:dateTime” />          </xsd:complexType>  </xsd:element>      <xsd:element name=”name”>          <xsd:complexType>              <xsd:sequence>                  <xsd:element name=”first” type=”xsd:string” />                  <xsd:element name=”middle” type=”xsd:string” />                  <xsd:element name=”last” type=”xsd:string” />              </xsd:sequence>              <xsd:attribute name=”local” type=”xsd:boolean” />          </xsd:complexType>  </xsd:element>  </xsd:schema>  以上文档对应有效的XML文档如下：  <?xml version=”1.0”?>  <ContactDetails lastUpdated=” 2005-3-18T14:48:12”>  <name local=”true”>         <first>santld</first>          <middle/>          <last>wang</last>  </name>  <rate>10.27</rate>  </ContactDetails> |

### 简单类型

虽然从内置数据类型中得到了许多的功能，但是在许多情况下，只有数据类型来限制数据的值是远远不够的。这里在学习简单类型之前，先看看关于约束。

**--约束方面**

enumeration ：用空格分开的一组指定的数值。它把数据类型约束为指定的值。

fractionDigit：指定小数点后的最大位数。

length：长度单位。

minExclusive：下限值，所有的值都必须大于该值。

maxExclusive ：上限值，所有的值都应该小于该值。

minLength：长度单位的最小个数。

maxLength ：长度单位的最大数。

minInclusive：最小值，所有的值都应该大于或者等于该值。

maxInclusive：最大值，所有的值都应该小于或者等于该值。

pattern ：数据类型的值必须匹配的指定模式，pattern值必须是一个正则表达式。

totalDigits：指定小数最大位数的值。

whiteSpace 其值为：preserve（值中的空格不能改变），replace（所有的制表符、换行符和回车符都用空格代替），collapse（执行replace，删除相邻的、结尾处和开头处的空格）。

要应用上述的约束，就要利用元素restriction。这个元素中有2个属性：id属性是模式文档中restriction元素的唯一标识符，base属性设置为一个内置的XSD数据类型或者现有的简单类型定义，他是一种被限制的类型。

|  |
| --- |
| <xsd:restriction base=”xsd:string”>  <xsd:minLength value=”4” />      <xsd:maxLength value=”10” />  </xsd:restriction>  如上，将字符串最小长度限定为4个字符，将最大长度限定为10个字符。  <xsd:restriction base=”xsd:int”>  <xsd:minInclusive value=”1” />  <xsd:maxInclusive value=”100” />  </xsd:restriction>  如上，将一个整数的取值范围设置为1到100之间。  <xsd:restriction base=”xsd:string”>  <xsd:enumeration value=”FistValue” />  <xsd:enumeration value=”SecondValue” />  <xsd:enumeration value=”ThirdValue” />  </xsd:restriction> 如上，字符串只能为枚举出的三种值。 |

--**简单类型**

简单类型是对一个节点的可能值进一步限制的自定义数据类型。创建简单类型需要利用simpleType元素，其定义如下：

|  |
| --- |
| <simpleType id=”ID” name=”NCName” final=”(#all|(list|union|restriction))” />  id属性应唯一的标明文档内的simpleType元素，name不能使用冒号字符。simpleType不能包含元素，也不能有属性，根据在simpleType中定义的规则，它基本上是一个值，或者是一个值的集合。 |
| <xsd:simpleType name=”personsName”>  <xsd:restriction base=”xsd:string”>          <xsd:minLength value=”4” />          <xsd:maxLength value=”8” />  </xsd:restriction>  </xsd:simpleType>  <xsd:element name=”first” type=”personsName” />  以上文档对应有效的XML文档如下：  <first>Santld</first>  以下就是无效的XML文档：  <first>SS</first> 或者 <first>Santld wang</first>  <xsd:simpleType name=”personsTitle”>  <xsd:restriction base=”xsd:string”>          <xsd:enumeration value=”Mr.” />          <xsd:enumeration value=”Mrs.” />          <xsd:enumeration value=”Miss.” />  </xsd:restriction>  </xsd:simpleType>  <xsd:element name=”title” type=”personsTitle” />  上面定义的类型personsTitle是一个字符串类型，但它的值只能是Mr.、Mrs.或者Miss.其中的一个。 |

### 复杂类型

复杂类型的定义必须使用complexType元素，在这里可以包含属性和元素。在复杂类型的使用中，主要是complexType和simpleType配合使用。

格式：<xsd:complexType name=”name” />

|  |
| --- |
| <xsd:element name="name" type="FullName" />  <xsd:complexType name="FullName"> <xsd:sequence>  <xsd:element name="first" type="PersonsFirstname" minOccurs="0" maxOccurs="1"  default="John" />  <xsd:element name="middle" type="xsd:string" minOccurs="0"  maxOccurs="unbounded" nillable="true" />  <xsd:element name="last" type="xsd:string" minOccurs="1" maxOccurs="1"  default="Doe" />  </xsd:sequence>  <xsd:attribute name="title" type="PersonsTitle" default="Mr." />  </xsd:complexType>  <xsd:simpleType name="PersonsFirstname">  <xsd:restriction base="xsd:string">  <xsd:minLength value="4" />  <xsd:maxLength value="10" />  </xsd:restriction>  </xsd:simpleType>    <xsd:simpleType name="PersonsTitle">  <xsd:restriction base="xsd:string">  <xsd:enumeration value="Mr." />  <xsd:enumeration value="Mrs." />  <xsd:enumeration value="Miss." />  </xsd:restriction>  </xsd:simpleType>  如上就实现了一个复杂类型，该例子实现了一个复杂类型FullName，其中包含了两个简单类型PersonsFirstname和PersonsTitle。 |

**-- 分组和属性**

在为定义XML文档而创建的一些更为复杂的定义中，会有一些元素集、属性集的组合，这是我们就使用了分组的概念。分组定义中使用的是group元素。

|  |
| --- |
| <xsd:group name=”CityChoice”>  <xsd:choice>          <xsd:element name=” Beijing ” type=”xsd:string” />          <xsd:element name=” Shanghai ” type=”xsd:string” />          <xsd:element name=”Xian” type=”xsd:string” />  </xsd:choice>  </xsd:group>  <xsd:element name=”City”>  <xsd”group ref=”CityChoice” minOccurs=”1” maxOccurs=”1” />  </xsd:element>  对属性的分组，应该使用attributeGroup元素，例如：  <xsd:attributeGroup name=”contactAttribs”>  <xsd:attribute name=”city” type=”xsd:string” />  <xsd:attribute name=”country” type=”xsd:string” />  <xsd:attribute name=”age” type=”xsd:string” />  </xsd:attributeGroup>  <xsd:element name=”contact”>  <xsd:complexType>          <xsd:attributeGroup ref=”contactAttribs” />  </xsd:complexType>  </xsd:element>  以上文档对应有效的XML文档如下：  <contact city=” Beijing ” country=” China ” age=”25” /> |

**-- 内容模型**

内容模型可以对在XML文档内使用的元素、属性和类型进行限制，确定用户可以在XML实例的哪些等级添加自己的元素和属性。

|  |
| --- |
| **Any**：当在XML中声明元素时any是默认的内容模型，该模型可以包含文本、元素和空格。如果允许添加元素的内容，且无需屋里修改模式文件，就可以使用该模型。  <xsd:xchema xmlns:xsd=”http://www.w3.org/2001/XMLSchema”>  <xsd:element name=”name”>          <xsd:complexType>              <xsd:sequence>                  <xsd:element name=”first” type=”xsd:string” />                  <xsd:element name=”middle” type=”OtherNames” />                  <xsd:element name=”last” type=”xsd:string” />              </xsd:sequence>          </xsd:complexType>  </xsd:element>  <xsd:complexType name=”OtherNames”>          <xsd:sequence>              <xsd:any namespace=”##any” processContents=”lax” minOccurs=”0”     maxOccurs=”unbounded” />          </xsd:sequence>  </xsd:complexType>  </xsd:schema>  例子中的xsd:any元素说明该类型允许添加内容。  namespace属性允许的值为：##any（元素可以来自任何命名空间）   * ##other（元素可以来自除了该元素的父元素所在的目标命名空间之外的命名空间） * ##local（元素不受命名空间的限制） * ##targetNamespace（元素来自父元素的目标命名空间）   processContents属性说明对这里所创建的元素进行验证时所执行的操作，取值有如下3种：   * strict（标明XML处理器必须获得和那些命名空间相关联的模式，并验证元素和属性） * lax（与strict相似，只是如果处理器找不到模式文档，也不会出现错误） * skip（不利用模式文档验证XML文档）   上述模式的一个有效实例如下：   <?xml version=”1.0”?>  <name>  <first>santld</first>  <middle>          <nameInChina>San</nameInChina>  </middle>  <last>wang</last>  </name> |
| **Empty**：empty元素禁止把文本或者元素作为一个声明为空的元素的子元素，如果要保证该元素不包括子元素、文本甚至空格，就可以使用它。在XSD中时利用xsd:anyType类型来限定它的，这样就意味元素只能包含属性  <xsd:schema xmlns:xsd=”http://www.w3.org/2001/XMLSchema”>  <xsd:element name=”contact”>          <xsd:complexType>              <xsd:complexContent>                  <xsd:restriction base=”xsd:anyType”>                      <xsd:attribute name=”age” type=”xsd:integer” />                  </xsd:restriction>              </xsd:complexContent>          </xsd:complexType>  </xsd:element>  </xsd:schema>  以上就是一个复杂类型，只允许有一个age属性。在例子里complexContent元素可以表示complexType的内容要进行扩充或者限制，在这里，我们对其内容进行限制，因此使用了restriction元素，如果扩成则使用exension元素。 |
| **Mixed**：最后一个内容模型就是mixed，它包含文本、内容和属性。在complexType元素上把mixed属性的值设为true，就一个声明一个mixed内容模型了。例如：  <xsd:schema xmlns:xsd=”http://www.w3.org/2001/XMLSchema”>  <xsd:element name=”contact”>          <xsd:complexType mixed=”true”>              <xsd:sequence>                  <xsd:element name=”first” type=”xsd:string” />              </xsd:sequence>          <xsd:complexType>  </xsd:element>  </xsd:schema>  上述模式的一个有效实例如下：  <?xml version=”1.0”?>  <contact>My first name is<first>Santld</first>.</contact>  在例子中contact元素就包含了文本和元素first。 |