# day07-XML

## XML（50分钟）

###### XML的概述

1.1　什么是XML

XML全称为Extensible Markup Language，意思是可扩展的标记语言。XML语法上和HTML比较相似，但HTML中的元素是固定的，而XML的标签是可以由用户自定义的。

W3C在1998年2月发布1.0版本，2004年2月又发布1.1版本，但因为1.1版本不能向下兼容1.0版本，所以1.1没有人用。同时，在2004年2月W3C又发布了1.0版本的第三版。我们要学习的还是1.0版本！！！

1.2　XML的应用场景

　　保存关系型数据：

　　<student number="1001">

<name>zhangSan</name>

<age>23</age>

<sex>male</sex>

<teacher name=”liSi”>

<wife id=”xxx”><name>xxx</name></wife>

</teacher>

</student>

配置文件：

　　<servlet>

<servlet-name>MyServlet</servlet-name>

<servlet-class>cn.itcast.servlet.MyServlet</servlet-class>

</servlet>

###### 2　XML语法

来看一个XML文档

students.xml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" standalone="no"?>  <students>  <student number="1001">  <name>zhangSan</name>  <age>23</age>  <sex>male</sex>  </student>  <student number="1002">  <name>liSi</name>  <age>32</age>  <sex>female</sex>  </student>  <student number="1003">  <name>wangWu</name>  <age>55</age>  <sex>male</sex>  </student>  </students> |

2.1　XML文档声明

* 文档声明必须为<?xml开头，以?>结束；
* 文档声明必须从文档的0行0列位置开始；
* 文档声明只有三个属性：
* versioin：指定XML文档版本。必须属性，因为我们不会选择1.1，只会选择1.0；
* encoding：指定当前文档的编码。可选属性，默认值是utf-8；
* standalone：指定文档独立性。可选属性，默认值为yes，表示当前文档是独立文档。如果为no表示当前文档不是独立的文档，会依赖外部文件。

2.2　元素

元素是XML文档中最重要的组成部分：

* 普通元素的结构：开始标签、元素体、结束标签，例如：<hello>大家好</hello>；
* 元素体：元素体可以是元素，也可以是文本，例如：<b><a>你好</a></b>，其中<b>元素的元素体是<a>元素，而<a>元素的元素体是文本；
* 空元素：空元素只有开始标签，而没有结束标签，例如：<c/>，当元素必须自己闭合。

###### 3　属性

<student number="1001">

<name>zhangSan</name>

<age>23</age>

<sex>male</sex>

</student>

* 属性是元素的一部分，它必须出现在元素的开始标签中；
* 属性的定义格式：属性名=属性值，其中属性值必须使用单引或双引；
* 一个元素可以有0~N个属性，但一个元素中不能出现同名属性；

###### 4　注释

　　XML的注释与HTML相同，即以“<!--”开始，以“-->”结束。注释内容会被XML解析器忽略！

###### 5　转义字符和CDATA段

5.1　转义字符

XML中的转义字符与HTML一样。

因为很多符号已经被XML文档结构所使用，所以在元素体或属性值中想使用这些符号就必须使用转义字符，例如：“<”、“>”、“’”、“””、“&”。



例如：<a>&lt;hello&gt;</a>，<a>元素内部会被解释为：<hello>！

5.2　CDATA段

当大量的转义字符出现在xml文档中时，会使xml文档的可读性大幅度降低。这时如果使用CDATA段就会好一些。

在CDATA段中出现的“<”、“>”、“””、“’”、“&”，都无需使用转义字符。这可以提高xml文档的可读性。

|  |
| --- |
| <a><![CDATA[<a>]]></a> |

在CDATA段中不能包含“]]>”，即CDATA段的结束定界符。

###### 6　处理指令(了解即可)

处理指令，简称PI（Processing instruction）。处理指令用来指挥解析器如何解析XML文档内容。

例如，在XML文档中可以使用xml-stylesheet指令，通知XML解析器，应用css文件显示xml文档内容。

|  |
| --- |
| <?xml-stylesheet type="text/css" href="a.css"?> |

处理指令以“<?”开头，以“?>”结束，这一点与xml文档声明相同。

|  |
| --- |
| gj1 {font-size: 200px; color: red;}  gj2 {font-size: 100px; color: green;}  gj3 {font-size: 10px;}  gj4 {font-size: 50px; color: blue;} |
| <?xml version="1.0" encoding="gbk"?>  <?xml-stylesheet type="text/css" href="a.css" ?>  <gjm>  <gj1>中国</gj1>  <gj2>美国</gj2>  <gj3>日本</gj3>  <gj4>英国</gj4>  </gjm> |

###### 7　格式良好的XML文档

格式良好的XML就是格式正确的XML文档，只有XML的格式是良好的，XML解释器才能解释它。下面是对格式良好XML文档的要求：

* 必须要有XML文档声明；
* 必须且仅能有一个根元素；
* 元素和属性的命名必须遵循XML要求：
* XML命名区分大小写，例如<a>和<A>是两个不同的元素；
* 名称中可以包含：字母、数字、下划线、减号，但不能以数字、减号开头；
* 不能以xml开头，无论是大写还是小写都不可以，例如<xml>、<Xml>、<XML>都是错误的；
* 不能包含空格，例如<ab cd>是错误的。
* 元素之间必须合理包含，例如：<a><b>xxx</b></a>是合理的，而<a><b>xxx</a></b>就是错误的包含。

## DTD

###### 1　DTD概述

1.1　什么是DTD

DTD（Document Type Definition），文档类型定义，用来约束XML文档。或者可以把DTD理解为创建XML文档的结构！例如可以用DTD要求XML文档的根元素名为<students>，<students>中可以有1~N个<student>，<student>子元素为<name>、<age>和<sex>，<student>元素还有number属性。

DTD不是XML文档，它是XML文档的约束文件！就像法律与人一样！

展示DTD文档

|  |
| --- |
| <!ELEMENT students (student+)>  <!ELEMENT student (name,age,sex)>  <!ELEMENT name (#PCDATA)>  <!ELEMENT age (#PCDATA)>  <!ELEMENT sex (#PCDATA)> |

1.2　DTD分类

* 内部DTD：在XML文档内部嵌入DTD，只对当前XML文档有效；
* 外部DTD：独立的DTD文件，扩展名为.dtd；
* 本地DTD：DTD文件在本地，不在网络上。自己项目，或本公司内部使用的；
* 公共DTD：DTD文件在网络上，不在本地。都是大公司或组织发布的，共大家使用！

1.3　内部DTD

<!DOCTYPE 根元素名称 [

DTD约束代码

]>

内部DTD：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" standalone="yes" ?>  <!DOCTYPE students [  <!ELEMENT students (student+)>  <!ELEMENT student (name, age, sex)>  <!ELEMENT name (#PCDATA)>  <!ELEMENT age (#PCDATA)>  <!ELEMENT sex (#PCDATA)>  ]>  <students>  <student>  <name>zhangSan</name>  <age>23</age>  <sex>male</sex>  </student>  </students> |

* 位置：内部DTD在文档声明下面，在根元素上面；
* 语法格式：放到“<!DOCTYPE 根元素名称 [”和“]>”之间；
* 只对当前XML文档有效；

<!DOCTYPE students[

DTD约束内容

]>

1.4　SYSTEM DTD

<!DOCTYPE students SYSTEM “路径”>

本地DTD

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" standalone="no" ?>  <!DOCTYPE students SYSTEM "students.dtd">  <students>  <student>  <name>zhangSan</name>  <age>23</age>  <sex>male</sex>  </student>  </students> |

students.dtd

|  |
| --- |
| <!ELEMENT students (student+)>  <!ELEMENT student (name, age, sex)>  <!ELEMENT name (#PCDATA)>  <!ELEMENT age (#PCDATA)>  <!ELEMENT sex (#PCDATA)> |

* 位置：本地硬盘上；
* 语法格式：直接定义元素或属性即可；
* 本地所有XML文档都可以引用这个dtd文件；

<!DOCTYPE 根元素名称 SYSTEM “dtd路径”>

* 1. 公共DTD

<!DOCTYPE students PUBLIC “描述信息例如:版本作者语言等等” “网络上DTD的路径”>

公共DTD

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" standalone="no" ?>  <!DOCTYPE students PUBLIC "-//qdmmy6//DTD ST 1.0//ZH" "http://www.qdmmy6.com/xml/dtds/st.dtd">  <students>  <student>  <name>zhangSan</name>  <age>23</age>  <sex>male</sex>  </student>  </students> |

公共DTD是说，DTD由某个公司或权威组织发布的，供大家使用的DTD。其格式如下：

<!DOCTYPE 根元素 PUBLIC "DTD名称" "DTD网址">

当然你需要知道要使用的DTD的名称和网址。如果某个机构公布了DTD，那么一定也会公布DTD的名称和网址。

###### 2　DTD语法之定义元素(了解)

2.1　定义元素语法

定义元素语法：<!ELEMENT 元素名 元素描述>

* <!ELEMENT name (#PCDATA)>，定义名为name的元素，内容为文本类型。
* <!ELEMENT student (name,age,sex)>，定义名为student元素，内容依次为name、age、sex元素；
* <!ELEMENT student ANY>，定义名为student元素，内容任意；
* <!ELEMENT student EMPTY>，定义名为student元素，不能有内容，即空元素，注意空元素是可以有属性的。<student />

2.2　子元素出现次数

可以使用\*、+、?来指定子元素出现的次数

\*：可以出现0~N次；

* +：可以出现1~N次；
* ?：可以出现0~1次。

例如：<!ELEMENT student(name,age?,hobby\*,grade+) >，定义student元素，第一子元素为name，必须且仅能出现一次，age是可有可无的，hobby可以出现0~N次，grade可以出现1~N次。

2.3　枚举类型子元素

<!ELEMENT student (name | age | sex)>，表示student子元素为name、age、sex其中之一，必须仅且能选择其一。

2.4　练习

* 根元素为students，可以包含1~N个student元素；
* student元素依次包含：name、age、sex元素；
* name、age、sex元素的内容类型为文本内容；
* 要求SYSTEM外部DTD。

###### 3　DTD语法之定义属性

3.1　定义属性的语法

<!ATTLIST 元素名 属性名 属性类型 设置说明>

例如：<!ATTLIST student number CDATA #REQUIRED>，给student元素定义属性number，类型为文本，这个默认是必须的。

3.2　属性设置说明

* #REQUIRED：说明属性是必须的；
* #IMPLIED：说明属性是可选的；
* 默认值：在不给出属性值时，使用默认值。

3.3　属性的类型

* CDATA：文本类型；
* Enumerated：枚举类型；
* ID：ID类型，ID类型的属性用来标识元素的唯一性，即元素的ID属性值不能与其他元素的ID属性值相同；
* IDREF：ID引用类型，用来指定另一个元素，与另一个元素建立关联关系，IDREF类型的属性值必须是另一个元素的ID。

|  |
| --- |
| <!ELEMENT students (student+) > <!ELEMENT student EMPTY> <!ATTLIST student number ID #REQUIRED> <!ATTLIST student name CDATA #REQUIRED> <!ATTLIST student sex (male | female) "male" > <!ATTLIST student friend IDREF #IMPLIED> |
| <?xml version="1.0" ?> <!DOCTYPE students SYSTEM "students.dtd">  <students>     <student number="itcast\_001" name="zhangSan"/>     <student number="itcast\_002" name="liSi" sex="male"/>     <student number="itcast\_003" name="wangWu" sex="female" friend="itcast\_002"/> </students> |

###### 4　DTD语法之定义实体（了解）

4.1　什么是实体

有时在XML中会出现很多相同的内容，例如“北京传智播客教育科技有限公司”，这个名称太长了，我们希望把这个值与一个“符号”绑定，然后在需要这个名称时使用它绑定的“符号”即可。这个符号就是实体了。例如：“&传智;”！

其中“传智”是实体名，而“北京传智播客教育科技有限公司”是实体值，XML被解析时，所有实体会被替换成实体名。

4.2　实体分类(了解)

实体分为两种：一般实体和参数实体。

* 一般实体：在XML文档中使用；
* 参数实体：在DTD使用。

4.3　一般实体(了解)

* 定义一般实体：<!ENTITY 实体名 “实体值”>，例如：<!ENTITY 大美女 “白冰”>；
* 一般实体引用：&实体名;，例如<xxx>&大美女;</xxx>。

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" standalone="no" ?>  <!DOCTYPE students SYSTEM "a.dtd">  <students>  <student number="itcast\_001" name="zhangSan"/>  <student number="itcast\_002" name="liSi" sex="male"/>  <student number="itcast\_003" name="wangWu" sex="female" friend="itcast\_002"/>  <student number="itcast\_004" name="&itcast;"/>  </students> |
| <!ELEMENT students (student+) >  <!ELEMENT student EMPTY>  <!ATTLIST student number ID #REQUIRED>  <!ATTLIST student name CDATA #REQUIRED>  <!ATTLIST student sex (male | female) "male">  <!ATTLIST student friend IDREF #IMPLIED>  <!ENTITY itcast "北京传智播客教育科技有限公司"> |

4.4　参数实体(了解)

* 定义参数实体：<!ENTITY % 实体名 "实体值">，“%”与实体名之间的空格是必须的；
* 例如：<!ENTITY % friend "student friend IDREF #IMPLIED">
* 参数实体引用：%实体名;；
* 例如：<!ATTLIST %friend;>

参数实体是在DTD内部使用，而不是在XML中使用。

在内部DTD中使用参数实体会有诸多限制。

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" standalone="no" ?>  <!DOCTYPE students SYSTEM "a.dtd">  <students>  <student number="itcast\_001" name="zhangSan"/>  <student number="itcast\_002" name="liSi" sex="male"/>  <student number="itcast\_003" name="wangWu" sex="female" friend="itcast\_002"/>  <student number="itcast\_004" name="&itcast;"/>  </students> |
| <!ELEMENT students (student+) >  <!ELEMENT student EMPTY>  <!ATTLIST student number ID #REQUIRED>  <!ATTLIST student name CDATA #REQUIRED>  <!ATTLIST student sex (male | female) "male">  <!ENTITY % friend "<!ATTLIST student friend IDREF #IMPLIED>">  %friend;  <!ENTITY itcast "北京传智播客教育科技有限公司"> |

## Schema

###### 1　Schema概述

1.1　什么是Schema

* Schema是新的XML文档约束；DTD出现的比较早.
* Schema要比DTD强大很多；
* Schema本身也是XML文档，但Schema文档的扩展名为xsd，而不是xml。

1.2　Schema简介

　　本课程中不对Schema深入探讨，我们只对Schema有个了解即可。

students.xsd

|  |
| --- |
| <?xml version="1.0"?> <xsd:schema xmlns="http://www.itcast.cn/xml"         xmlns:xsd="http://www.w3.org/2001/XMLSchema"         targetNamespace="http://www.itcast.cn/xml" elementFormDefault="qualified">     <xsd:element name="students" type="studentsType"/>     <xsd:complexType name="studentsType">         <xsd:sequence>             <xsd:element name="student" type="studentType" minOccurs="0" maxOccurs="unbounded"/>         </xsd:sequence>     </xsd:complexType>     <xsd:complexType name="studentType">         <xsd:sequence>             <xsd:element name="name" type="xsd:string"/>             <xsd:element name="age" type="ageType" />             <xsd:element name="**sex**" type="sexType" />         </xsd:sequence>         <xsd:attribute name="number" type="numberType" use="required"/>     </xsd:complexType>     <xsd:simpleType name="sexType">         <xsd:restriction base="xsd:string">             <xsd:enumeration value="male"/>             <xsd:enumeration value="female"/>         </xsd:restriction>     </xsd:simpleType>     <xsd:simpleType name="ageType">         <xsd:restriction base="xsd:integer">             <xsd:minInclusive value="0"/>             <xsd:maxInclusive value="120"/>         </xsd:restriction>     </xsd:simpleType>     <xsd:simpleType name="numberType">         <xsd:restriction base="xsd:string">             <xsd:pattern value="ITCAST\_\d{4}"/>         </xsd:restriction>     </xsd:simpleType> </xsd:schema> |

students.xml

|  |
| --- |
| <?xml version="1.0"?>  <students xmlns="http://www.itcast.cn/xml"         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"          xsi:schemaLocation="http://www.itcast.cn/xml students.xsd" >     <student number="ITCAST\_1001">         <name>zhangSan</name>         <age>20</age>         <sex>male</sex>     </student>     <student number="ITCAST\_1002">         <name>liSi</name>         <age>25</age>         <sex>female</sex>     </student> </students> |

###### 2　Schema名称空间

2.1　什么是名称空间

如果一个XML文档中使用多个Schema文件，而这些Schema文件中定义了相同名称的元素时就会出现名字冲突。这就像一个Java文件中使用了import java.util.\*和import java.sql.\*时，在使用Date类时，那么就不明确Date是哪个包下的Date了。

总之名称空间就是用来处理元素和属性的名称冲突问题，与Java中的包是同一用途。如果每个元素和属性都有自己的名称空间，那么就不会出现名字冲突问题，就像是每个类都有自己所在的包一样，那么类名就不会出现冲突。

2.2　目标名称空间

在XSD文件中为定义的元素指定名称，即指定目标名称空间。这需要给<xsd:schema>元素添加targetNamespace属性。

* <xsd:schema targetNamespace="http://www.itcast.cn/xml">

名称空间可以是任意字符串，但通常我们会使用公司的域名作为名称空间，这与Java中的包名使用域名的倒序是一样的！千万不要以为这个域名是真实的，它可以是不存在的域名。

如果每个公司发布的Schema都随意指定名称空间，如a、b之类的，那么很可能会出现名称空间的名字冲突，所以还是使用域名比较安全，因为域名是唯一的。

当使用了targetNamespace指定目标名称空间后，那么当前XSD文件中定义的元素和属性就在这个名称空间之中了。

2.3　XML指定XSD文件

在XML文件中需要指定XSD约束文件，这需要使用在根元素中使用schemaLocation属性来指定XSD文件的路径，以及目标名称空间。格式为：schemaLocation=”目标名称空间 XSD文件路径”

* <students schemaLocation="http://www.itcast.cn/xml students.xsd">

schemaLocation是用来指定XSD文件的路径，也就是说为当前XML文档指定约束文件。但它不只要指定XSD文件的位置，还要指定XSD文件的目标名称空间。

其中http://www.itcast.cn/xml为目标名称空间，students.xsd为XSD文件的位置，它们中间使用空白符（空格或换行）分隔。

也可以指定多个XSD文件，格式为：

* schemaLocation=”目标名称空间1 XSD文件路径1 目标名称空间2 XSD文件路径2”

下面是spring配置文件的例子，它一共指定两个XSD文件

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <beans     xsi:schemaLocation="http://www.springframework.org/schema/beans                        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd                        http://www.springframework.org/schema/aop                        http://www.springframework.org/schema/aop/spring-aop-3.0.xsd"> </beans> |

下面是JavaWeb项目的配置文件：

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <web-app xsi:schemaLocation="http://java.sun.com/xml/ns/javaee                              http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"> </web-app> |

2.4　定义名称空间

现在我们已经知道一个XML中可以指定多个XSD文件，例如上面Spring的配置文件中就指定了多个XSD文件，那么如果我在<beans>元素中给出一个子元素<bean>，你知道它是哪个名称空间中的么？

|  |
| --- |
| <beans     xsi:schemaLocation="http://www.springframework.org/schema/beans                        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd                        http://www.springframework.org/schema/aop                        http://www.springframework.org/schema/aop/spring-aop-3.0.xsd">  <bean></bean> </beans> |

所以只是使用schemaLocation指定XSD是不够的，它只是导入了这个XSD及XSD的名称空间而已。schemaLocation的作用就相当于Java中导入Jar包的作用！最终还是在Java文件中使用import来指定包名的。

xmlns是用来指定名称空间前缀的，所谓前缀就是“简称”，例如中华人发共和国简称中国一样，然后我们在每个元素前面加上前缀，就可以处理名字冲突了。

格式为：xmln:前缀=”名称空间”

注意，使用xmlns指定的名称空间必须是在schemaLocation中存在的名称空间。

|  |
| --- |
| <beans  xmlns:b="http://www.springframework.org/schema/beans"  xmlns:aop="http://www.springframework.org/schema/aop"     xsi:schemaLocation="http://www.springframework.org/schema/beans                        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd                        http://www.springframework.org/schema/aop                        http://www.springframework.org/schema/aop/spring-aop-3.0.xsd">  <b:bean></b:bean>  <aop:scoped-proxy/>  </beans> |

2.5　默认名称空间

在一个XML文件中，可以指定一个名称空间没有前缀，那么在当前XML文档中没有前缀的元素就来自默认名称空间。

|  |
| --- |
| <beans  xmlns="http://www.springframework.org/schema/beans"  xmlns:aop="http://www.springframework.org/schema/aop"     xsi:schemaLocation="http://www.springframework.org/schema/beans                        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd                        http://www.springframework.org/schema/aop                        http://www.springframework.org/schema/aop/spring-aop-3.0.xsd">  <bean></bean>  <aop:scoped-proxy/>  </beans> |

2.6　W3C的元素和属性

如果我们的XML文件中需要使用W3C提供的元素和属性，那么可以不在schemaLocation属性中指定XSD文件的位置，但一定要定义名称空间，例如：

|  |
| --- |
| <beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"       xsi:schemaLocation="http://www.springframework.org/schema/beans                        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd                        http://www.springframework.org/schema/aop                        http://www.springframework.org/schema/aop/spring-aop-3.0.xsd">  </beans> |

上面定义了一个名称空间，前缀为xsi，名称空间为http://www.w3.org/2001/XMLSchema-instance。这个名称空间无需在schemaLocation中不存在这个名称空间。

你可能已经发现了，schemaLocation这个属性其实是w3c定义的属性，与元素一定，属性也需要指定“出处”，xsi:schemaLocation中的xsi就是名称空间前缀。也就是说，上面我们在没有指定xsi名称空间时，就直接使用schemaLocation是错误的。