1 XML 的诞生

首先，让我们来回顾一下可扩展置标语言 XML（eXtensible Markup Language）的发展简史。

XML 有两个先驱--SGML 和 HTML，这两个语言都是非常成功的置标语言，但是它们都在某些方面存在着与生俱来的缺陷。SGML（Standard Generalized Markup Language）的全称是标准通用置标语言，它为语法置标提供了异常强大的工具，同时具有极好的扩展性，因此在分类和索引数据中非常有用。但是，SGML 非常复杂，并且价格昂贵，几个主要的浏览器厂商都明确拒绝支持 SGML，使 SGML 在网上传播遇到了很大障碍。

相反，超文本置标语言 HTML（HyperText Markup Language）免费、简单，在世界范围内得到了广泛的应用。它侧重于主页表现形式的描述，大大丰富了主页的视觉、听觉效果，为推动 WWW 的蓬勃发展、推动信息和知识的网上交流发挥了不可取代的作用。可是，HTML 也有如下几个致命的弱点，这些弱点逐渐成为 HTML 继续发展应用的障碍。

- HTML 是专门为描述主页的表现形式而设计的，它疏于对信息语义及其内部结构的描述，不能适应日益增多的信息检索要求和存档要求。
- HTML 对表现形式的描述能力实际上也还非常不够，它无法描述矢量图形、科技符号和一些其他的特殊显示效果。
- HTML 的标记集日益臃肿，而其松散的语法要求使得文档结构混乱而缺乏条理，导致浏览器的设计越来越复杂，降低了浏览的时间效率与空间效率。

正因为如此，1996 年人们开始致力于描述一个置标语言，它既具有 SGML 的强大功能和可扩展性，同时又具有 HTML 的简单性。XML 就是这样诞生的。

正象 SGML 和 HTML 一样，可扩展置标语言 XML 也是一种置标语言，它通过在数据中加入附加信息的方式来描述结构化数据。不过，XML 并非象 HTML 那样，只提供一组事先已经定义好的标记。准确地说，它是一种元置标语言，允许程序开发人员根据它所提供的规则，制定各种各样的置标语言。在 XML 中，置标的语法是通过文档类型定义 DTD（Document Type Definition）来描述的，也就是说，通过 DTD 来描述什么是有效的标记，并进一步定义置标语言的结构。除了定义置标的语法外，为了明确各个标记的含义，XML 还使用与之相连的样式单（style sheet）来向应用程序，比如浏览器，提供如何处理显示的指示说明。一言以蔽之，XML 是通过数据文档、DTD、样式单三个分离的部分来描述数据的。

虽然 XML 貌似复杂，但它有一些突出的优点：

1. 良好的可扩展性。XML 允许各个不同的行业根据自己独特的需要制定自己的一套标记，同时，它并不要求所有浏览器都能处理这成千上万个标记，同样也不要求一个置标语言能够适合各个行业各个领域的应用，这种具体问题具体分析的方法更有助于置标语言的发展。

2. 内容与形式的分离。正如前面所说，XML 中信息的显示方式已经从信息本身中抽取出来，放在了"样式单"中。这样做便于信息表现方式的修改，便于数据的搜索，也使得 XML 具有良好的自描述性，能够描述信息本身的含义甚至它们之间的关系。

3. 遵循严格的语法要求。XML 不但要求标记配对、嵌套，而且还要求严格遵守 DTD 的规定。这增加了网页文档的可读性和可维护性，也大大减轻了浏览器开发人员的负担，提高了浏览器的时间空间效率。

4. 便于不同系统之间信息的传输。不同企业、不同部门中往往存在着许多不同的系统，XML 可以用作各种不同系统之间的交流媒介，是一种非常理想的网际语言。

5. 具有较好的保值性。XML 的保值性来自它的先驱之一--SGML 语言，可以为文档提供 50 年以上的寿命。

正是基于这些优点，国际标准化组织--万维网联盟 W3C（World Wide Web Consortium）推荐 XML 作为第二代网页发布语言。

最后，让我们来看一个完整的 XML 例子，以便对 XML 的整体机制有一个大致的了解。在下面的例子中，我们用 XML 来描述一个学生花名册的信息列表。我们先为这些数据定义一个 DTD：

Stuml.dtd

```
<?xml version="1.0" encoding="GB2312"?>

<!ELEMENT 学生花名册 (学生)*>
<!ELEMENT 学生 (学号, 姓名, 性别, 籍贯, 出生日期)>
<!ELEMENT 出生日期 (年, 月, 日)>
<!ELEMENT 学号 (#PCDATA)>
<!ELEMENT 姓名 (#PCDATA)>
<!ELEMENT 性别 (#PCDATA)>
<!ELEMENT 籍贯 (#PCDATA)>
<!ELEMENT 年 (#PCDATA)>
<!ELEMENT 月 (#PCDATA)>
<!ELEMENT 日 (#PCDATA)>
```

关于学生花名册信息的标准 XML 文档是这样的：

Student.xml

```
<?xml version = "1.0" encoding="GB2312" standalone = "no"?>
<!DOCTYPE 学生花名册 SYSTEM "stuml.dtd">
<?xml-stylesheet type="text/xsl" href="mystyle.xsl"?>
<学生花名册>
<学生>
<学号>001</学号>
```

```
<姓名>张三</姓名>
<性别>男</性别>
<籍贯>北京市</籍贯>
<出生日期>
<年>1980</年>
<月>3</月>
<日>1</日>
</出生日期>
</学生>
<学生>
<学号>002</学号>
<姓名>李四</姓名>
<性别>女</性别>
<籍贯>河北省</籍贯>
<出生日期>
<年>1979</年>
<月>5</月>
<日>12</日>
</出生日期>
</学生>
</学生花名册>
```

现在我们为它制定一个样式单，以描述这些数据的显示：

```
    Mystyle.xsl
<?xml version="1.0" encoding="GB2312"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl"
xmlns="http://www.w3.org/TR/REC-html40" result-ns="">
<xsl:template><xsl:apply-templates/></xsl:template>
<xsl:template match = "/">
<HTML>
<HEAD>
<TITLE>学生花名册</TITLE>
</HEAD>
<BODY>
<xsl:apply-templates select="学生花名册"/>
</BODY>
</HTML>
</xsl:template>
<xsl:template match = "学生花名册">
<xsl:for-each select="学生">
<UL>
<LI><xsl:value-of select="姓名"/></LI>
<UL>
```

```
<LI>学号: <xsl:value-of select="学号"/></LI>
<LI>性别: <xsl:value-of select="性别"/></LI>
<LI>籍贯: <xsl:value-of select="籍贯"/></LI>
<LI>出生年: <xsl:value-of select="出生日期/年"/></LI>
<LI>出生月: <xsl:value-of select="出生日期/月"/></LI>
<LI>出生日: <xsl:value-of select="出生日期/日"/></LI>
</UL>
</UL>
</xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

在 IE5 中看到的 XML 文件的显示结果是这样的：

```
l 张三
m 学号: 001
m 性别: 男
m 籍贯: 北京市
m 出生年: 1980
m 出生月: 3
m 出生日: 1
l 李四
m 学号: 002
m 性别: 女
m 籍贯: 河北省
m 出生年: 1979
m 出生月: 5
m 出生日: 12
```

有兴趣的读者可以用"记事本"录入上面这三个文件，分别存盘后放在一个目录中，然后用 IE5 打开文件 student.xml，看看结果是不是这样。

2 XML 语法

与 HTML 不同，XML 对于语法有着严格的规定，只有当一个 XML 文档符合"格式良好"的基本要求时，处理程序才能对它加以分析和处理。"格式良好的"这一标准通过对 XML 文档的各个逻辑成分和物理成分进行语法规定，保证了 XML 严密的条理性、逻辑性和良好的结构性，从而大大提高了 XML 应用处理程序处理 XML 数据的准确性和效率。实际上，格式良好的要求就是 XML 规范的语法要求，一个简单的检验方法就是用 Internet Explorer4.01 以上版本打开正在编辑的 XML 文档，如果报错，这个文档就不是"格式良好的"。

XML 文档的结构包括逻辑结构和物理结构。

一、逻辑结构

一个 XML 文档通常以一个 XML 声明开始，通过 XML 元素来组织 XML 数据。XML 元素包括标记和字符数据。为了组织数据更加方便、清晰，还可以在字符数据中引入 CDATA 数据块，并可以在文档中引入注释。此外，由于有时需要给 XML 处理程序提供一些指示信息，XML 文档中可以包含处理指令。具体说来，各个逻辑元素的作用和形式如下：

1. XML 声明

XML 声明是处理指令的一种，一个 XML 文档最好以一个 XML 声明作为开始。下面是一个完整的 XML 声明：

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

在一个 XML 的处理指令中必须包括 version 属性，指明所采用的 XML 的版本号，而且它必须在属性列表中排在第一位。standalone 属性表明该 XML 文档是否和一个外部文档类型定义 DTD 配套使用。encoding 属性则指明了数据所采用的编码标准。

2．元素

元素是 XML 文档内容的基本单元。从语法上讲，一个元素包含一个起始标记、一个结束标记以及标记之间的数据内容。其形式是：

```
<标记>数据内容</标记>
```

对于标记有以下语法规定：

(1) 标记必不可少。任何一个格式良好的 XML 文档中至少要有一个元素。

(2) 大小写有别。

(3) 要有正确的结束标记。结束标记除了要和起始标记在拼写和大小写上完全相同，还必须在前面加上一个斜杠"/"。当一对标记之间没有任何文本内容时，可以不写结束标记，而在起始标记的最后冠以斜杠"/"来确认。这样的标记称为"空标记"。

(4) 标记要正确嵌套。

(5) 标记命名要合法。标记名应该以字母、下划线"_"或冒号":"开头，后面跟字母、数字、句号". "、冒号、下划线或连字符"-"，但是中间不能有空格，而且任何标记名不能以"xml"（或者"xml"大小写的任何组合，如"XML"、"xML"、"xmL"等等）起始。

(6) 有效使用属性。标记中可以包含任意多个属性，属性以名称/取值对出现，属性名不能重复，名称与取值之间用等号"="分隔，且取值用引号引起来。

3．CDATA 节

在标记 CDATA 下，所有的标记、实体引用都被忽略，而被 XML 处理程序一视同仁地当作字符数据看待。CDATA 的形式如下：

```
<！ [CDATA[  文本内容  ]] >
```

CDATA 的文本内容中不能出现字符串"]]>"，另外，CDATA 不能嵌套，

4．注释

有些时候，人们希望在 XML 文档中加入一些用作解释的字符数据，并且希望 XML 处理器不对它们进行任何处理。这种类型的文本称作注释（COMMENT）文本，在 XML 中，注释的方法与 HTML 完全相同，用"<！--"和"-->"将注释文本引起来。对于注释还有以下规定：

(1) 在注释文本中不能出现字符"-"或字符串"--"

(2) 不要把注释文本放在标记之中，类似地，不要把注释文本放在实体声明之中或之前。

(3) 注释不能被嵌套。

5．处理指令 PI

处理指令是用来给处理 XML 文档的应用程序提供信息的，XML 分析器把这些信息原封不动地传给应用程序，由应用程序来解释这个指令，遵照它所提供的信息进行处理。处理指令应该遵循下面的格式：

```
<？处理指令名 处理指令信息 >
```

下面是一个例子，它是描述辞典信息的 XML 文档：

```
[1] <?xml version="1.0" encoding="GB2312" standalone="no"?>
[2] <?xml-stylesheet type="text/xsl" href="mystyle.xsl"?>
[3] <辞典>
[4] <词条>
[5] <词目>XML</词目>
[6] <解释>XML 是一种可扩展的元置标语言，它可用以规定新的置标规则，并根据这个规则组织数据。</解释>
[7] <示例>
[8] <!-- 一个 XML 的例子 -->
[9] <![CDATA[
[10] <学生>
[11] <学号>001</学号>
[12] <姓名>张三</姓名>
[13] </学生>
[14] ]]>
[15] </示例>
[16] </词条>
[17]</辞典>
```

这个例子中出现的逻辑要素有：

[1] 是 XML 声明。

[2] 是处理指令。

[3]--[17] 是文档中的各个元素。在[5]行的"<词目>XML</词目>"中，"<词　　目>""</词目>"是标记，"XML"是字符数据。

[8] 是注释。

[9]--[14] 是 CDATA 节。

二、物理结构

从物理结构上讲，XML 文档是由一个或多个存贮单元构成的，这些存贮单元就是所谓的实体。所有的 XML 文档都包含了一个"根实体"，又称作"文档实体"。这个实体是由 XML 本身给出的，无须显式定义就可以使用，它指的其实就是整个文档的内容，是 XML 语法分析器处理的起点。除此之外，可能还需要用到其他一些实体，这些实体都用名字来标识，在文件类型定义 DTD 中给出定义。

简单地说，实体充当着和别名类似的角色，即，一个简单的实体名称可以用来代表一大段文本内容。象任何计算机别名系统一样，实体引用简化了录入工作，因为每当要使用同样一大段文本时，只须使用它的别名就可以了，处理器会自动把这个别名替换为相应的文本。

实体分为两大类，如下表所示：

类型		通用实体	参数实体
使用场合		用在 XML 文档中	只用在 DTD 中元素和属性的声明中
声明方式	内部	<!ENTITY 实体名 "文本内容">	<!ENTITY % 实体名 "文本内容">
	外部	<!ENTITY 实体名 SYSTEM"外部文件 URL 地址">	<!ENTITY % 实体名 SYSTEM"外部文件 URL 地址">
引用方式		&实体名；	%实体名；

下面例子中定义了一个描述单位地址的通用实体：

```
<!ENTITY address "北京市海淀区学府路 88 号，100000">
```

在 XML 文件中可以如下使用这个实体：

```
<客户> <姓名> 王五</姓名> <电话> （010）62626666</电话> <联系地址>
&address;</联系地址> </客户>
```

实体引用有以下几点规则：

- 除了 XML 标准规定的预定义实体以外，在 XML 文档引用一个实体之前，必须已经对此实体进行过声明。
- 实体引用中不能出现空格。
- 尽管在一个实体中可以再引用其他实体，但是不能出现循环引用。
- 实体引用的文档必须符合 XML 语法的种种要求
- 任何一个独立的逻辑要素，比如元素、标记、注释、处理指令、实体引用等等，都不能开始于一个实体，而结束于另一个实体。

如果在属性中出现实体引用，不但要遵守前面所述的实体引用的种种规则，还要注意以下两点：

- 在标记属性中不能引用一个外部实体。

引用的文本中不能出现字符"<"，否则替换后就不再是一个"格式良好的"XML 文件了。

3 XML DTD 的建立与使用（上）

XML 的精髓是允许文档的编写者制定基于信息描述、体现数据之间逻辑关系的自定义标记，确保文档具有较强的易读性、清晰的语义和易检索性。因此，一个完全意义上的 XML 文档不仅仅是"格式良好的"，而且还应该是使用了一些自定义标记的"有效的"XML 文档，也就是说，它必须遵守文档类型定义 DTD 中已声明的种种规定。

DTD 描述了一个置标语言的语法和词汇表，也就是定义了文档的整体结构以及文档的语法。简而言之，DTD 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。

DTD 的使用

一个 DTD 可以是内部的，包含在 XML 文档的前导说明部分；也可以是外部的，作为一个外部文档被引用。一个包含内部 DTD 的 XML 文档的结构为：

```
<?xml version = "1.0" encoding="GB2312" standalone = "yes"?>
<!DOCTYPE 根元素名[
元素描述
]>
文档体.....
```

外部 DTD 的好处是：它可以方便高效地被多个 XML 文档所共享。只要写一个 DTD 文档，就可以被多个 XML 文档所引用。使用外部 DTD 的 XML 文档的结构为：

```
<?xml version = "1.0" encoding="GB2312" standalone = "no"?>
<!DOCTYPE 根元素名 SYSTEM "外部 DTD 文件的 URL">
文档体.....
```

实际上，SYSTEM 不是引用外部 DTD 的唯一方法，这个关键字主要用于引用一个作者或组织所编写的众多 XML 文档中通用的 DTD。还存在另一种外部 DTD：一个由权威机构制订的、提供给特定行业或公众使用的 DTD。因此，引用外部 DTD 的另一个方法是使用关键字 PUBLIC，引用公开给公众使用的 DTD，在 DOCTYPE 中说明的形式为：

```
<!DOCTYPE 根元素 PUBLIC "DTD 名称" "外部 DTD 的 URL">
```

DTD 中元素类型的声明

DTD 中使用元素类型声明 ETD（Element Type Declaration）来声明所有有效的文档元素。ETD 应该采用如下的结构：

```
<!ELEMENT 元素名 元素内容描述>
```

XML 的标准将元素按内容划分为四类：

1. 空元素类型。定义方式为：
   <!ELEMENT 元素名 EMPTY>
   这类元素在 XML 文档中使用空元素标记，元素中没有内容。

2. ANY 元素类型。定义方式为：
   <!ELEMENT 元素名 ANY>
   XML 文档里该元素中可以包含任何内容。建议一般只把文档的根元素规定为 ANY 类型。

3. 父元素类型
   这类元素中可以包含子元素，在 DTD 中通过正则表达式规定子元素出现的顺序和次数。语法分析器将这些正则式与 XML 文档内部的数据模式相匹配，判别出一个文档是否是"有效的"。 在下表中，我们列出了正则表达式中可能出现的元字符：

元字符	含义	举例
元素 A 元素 B 元素 C	元素列表，无须遵从顺序要求	<!ELEMENT 联系人 (姓名 EMAIL)>
,	并（AND），要求严格遵从顺序要求	<!ELEMENT 联系人 (姓名,EMAIL)>
+	出现一次或多次	<!ELEMENT 联系人 (姓名,EMAIL+)>
*	出现零次或多次	<!ELEMENT 联系人 (姓名,EMAIL*)>
()	一组要共同匹配的表达式	<!ELEMENT 联系人 (姓名,EMAIL)+>
|	或（OR）	<!ELEMENT 联系人 (姓名,(电话 |EMAIL))>
?	可选，不出现或出现一次	<!ELEMENT 联系人 (姓名, (电话 |EMAIL), 地址?)>

4. 混合元素类型

这类元素中可以包含文本，同时文本之间可以有选择地插入子元素，但子元素出现的顺序和次数不受限制。它的定义方式是：
<!ELEMENT 元素名 （#PCDATA | 子元素名 1 | 子元素名 2 | ...）*>

4 XML DTD 的建立与使用(下)

DTD 中元素属性的声明

在 DTD 中定义属性时，我们使用下面的格式来给一个元素定义一组合适的属性，同时指定这些属性的类型和缺省值：

<!ATTLIST 元素名 （属性名 属性类型 缺省值）*>

- 元素名是属性所属的元素的名字。
- 属性名是属性的命名。
- 缺省值是属性的初值。有以下四种类型：

名 称	定 义	说明方式
必须赋值的属性	XML 文档中必须为这个属性给出一个属性值	<!ATTLIST 元素名 属性名 属性类型 #REQUIRED>
属性值可有可无的属性	不要求在 XML 文档中给该属性赋值，而且也无须在 DTD 中为该属性提供缺省值	<!ATTLIST 元素名 属性名 属性类型 #IMPLIED>
固定取值的属性	需要为一个特定的属性提供一个缺省值，并且不希望 XML 文档中另外给出元素值把这个缺省值替代掉	<!ATTLIST 元素名 属性名 属性类型 #FIXED "缺省值">
事先定义了缺省值的属性	需要在 DTD 中提供一个缺省值，在 XML 文档中可以为该属性给出新的属性值，也可以不另外给出属性值，采用 DTD 中给出的缺省值。	<!ATTLIST 元素名 属性名 属性类型 "缺省值">

这里，给出一个完整的例子：

```
<!ATTLIST 页面作者
   姓名 CDATA #IMPLIED
  年龄 CDATA #IMPLIED
  联系信息 CDATA #REQUIRED
  网站职务 CDATA #FIXED "页面作者"
  个人爱好 CDATA "上网"
  >
```

属性类型用来指定该属性是属于十个有效属性类型中的哪种类型。
这十种类型是：

类 型	含 义
CDATA	纯文本，由可显示字符组成的字符串
Enumerated	取值来自一组可接受的取值的列表
ID	以属性值的方式为文档中的某个元素定义唯一的标识，用以区分具有相同结构相同属性的不同元素
IDREF	属性值引用已定义的 ID 值，方法是把那个元素的 ID 标识值作为该属性的取值；IDREFS 是 IDREF 的复数形式，取值可以是若干个 ID 标识
IDREFS	
ENTITY	取值为一个已定义的实体
ENTITIES	
NMTOKEN	面向处理程序的类型
NMTOKENS	
NOTATION	取值为一个 DTD 中声明的符号，这个类型对于使用非 XML 格式的数据非常有用。

这十种类型，又可分为三个大类。第一大类是字符串类型，指的是 CDATA，它的值可以是任何合法的字符串。第二大类是枚举类型，包括 Enumerated 和 NOTATION，需要在 DTD 中为它们声明可取值的列表。剩下的七个类型都属于第三大类，又称为记法类型，含有不同的词法及语意限定。

参数实体

最后再来说一说参数实体。前面说过，参数实体专门用于 DTD，它可以方便元素和属性的声明。在下面这个例子中，参数实体简化了本科生信息和研究生信息中相同部分的录入工作：

```
<!ENTITY % GENERAL_INFO "姓名 | 学号 | 性别 | 出生日期 ">
<!ELEMENT 本科生 (%GENERAL_INFO;)>
<!ELEMENT 研究生 (%GENERAL_INFO; | 导师)>
```

虽然从理论上讲参数实体所代表的内容可以为任何文本，但由于它专门用于 DTD 中，出现在元素定义的正则表达式里，所以对它有一些特殊的规定。参数实体的替换文本中括号必须成对出现，并且，它不能以连接符"|"或","结尾，因为这样常常会留有错误隐患。

5 XML Schema

Schema 的由来

DTD 作为 XML 1.0 规范的重要组成部分，对于 XML 文档的结构起到很好的描述作用。但是，它也具有一些缺点，比如，它采用了非 XML 的语法规则、不支持数据类型、扩展性较差等等。Schema 正好解决了这些问题。从总体上讲，Schema 具有以下优点：

- 一致性：Schema 使得对 XML 的定义不必再利用一种特定的形式化的语言，而是直接借助 XML 自身的特性，利用 XML 的基本语法规则来定义 XML 文档的结构，使得 XML 达到了从内到外的完美统一，也为 XML 的进一步发展奠定了坚实的基础。
- 扩展性：Schema 对 DTD 进行了扩充，引入了数据类型、命名空间，从而使其具备较强的可扩展性。
- 互换性：利用 Schema，我们能够书写 XML 文档以及验证文档的合法性。另外，通过特定的映射机制，还可以将不同的 Schema 进行转换，以实现更高层次的数据交换。
- 规范性：同 DTD 一样，Schema 也提供了一套完整的机制以约束 XML 文档中置标的使用，但相比之下，后者基于 XML，更具有规范性。Schema 利用元素的内容和属性来定义 XML 文档的整体结构，如哪些元素可以出现在文档中、元素间的关系是什么、每个元素有哪些内容和属性、以及元素出现的顺序和次数等等，都可一目了然。

Schema 的发展

Schema 是伴随着 XML1.0 规范的制订而推出的，从 Schema 的第一个方案到现在为止，W3C 成员共提交了五个 Schema 规范，分别是 XML-Data、DCD(Document Content Description for XML)、SOX(Schema for Object-Oriented XML)、DDML(Document Definition Markup Language)和 XML Schema。直到现在，关于 Schema 还没有一个正式推荐标准，它仍处于不断修改完善的过程当中。

初识 Schema

下面以一个简单的示例对 Schema 进行介绍（注：本例以及后面的 Schema 语法均以微软 Internet Explorer 5.0 的 Schema 实现为蓝本）：

```
1. <?xml version="1.0" encoding="GB2312" ?>
2. <Schema xmlns="urn:schemas-microsoft-com:xml-data"
   xmlns:dt="urn:schemas-microsoft-com:datatypes">
3. <AttributeType name="序号"/>
4. <AttributeType name="性别"/>
5. <ElementType name= "姓名"/>
6. <ElementType name= "年龄"/>
7. <ElementType name= "电话" dt:type="fixed.14.4"/>
8. <ElementType name= "地址" />
9. <ElementType name= "联系人" content="eltOnly">
10. <element type= "姓名" />
11. <element type= "年龄" />
12. <element type= "电话" />
13. <element type= "地址" />
14. </ElementType>
15. <ElementType name= "通讯录" content="eltOnly">
16. <element type= "联系人" />
17. <attribute type="序号"/>
18. <attribute type="性别"/>
19. </ElementType>
20.</Schema>
```

第 1 行是 XML 类型声明语句，指明该文档是一个 XML 文档，并且符合版本 1.0 规范；该文档采用 GB2312 编码。

第 2 行是 Schema 声明语句，它包含了 Schema 命名空间的声明。本例中用到了两个命名空间：一是 xmlns="urn:schemas-microsoft-com:xml-data"，它指定本文档是一个 XML Schema 文档；另一个是 xmlns:dt="urn:schemas-microsoft-com:datatypes"，它定义了在本文档中可以使用的数据类型。

第 3、4 行是属性定义语句，它定义了两个属性：序号和性别。

第 5、6、7、8 行是元素定义语句，它定义了四个元素：姓名、年龄、电话、地址。其中为"电话"元素定义了数据类型：fixed.14.4。

第 9-14 行定义了本 XML Schema 的二级元素：联系人，指明该元素包含四个子元素：姓名、年龄、电话、地址。

第 15-19 行定义了本 XML Schema 的顶级元素：通讯录，指明该元素包含一个子元素：联系人，以及两个属性：序号、性别。

第 20 行是结束标记语句，它表明该 XML Schema 的描述到此为止。

Schema 语法

Schema 有着自己的一套完整的语法，涉及到的关键元素包括：Schema、ElementType、element、AttributeType、attribute、group、datatype、description。

Schema元素是XML Schema中第一个出现的元素，用于声明该XML文档是一个Schema文档。Schema 具有两个属性：name 指定该 Schema 的名称，而 xmlns 则指定该 Schema 包含的命名空间。

ElementType 元素是 XML Schema 中重要元素之一，用于定义该 XML Schema 文档中出现的元素。通过属性 content 来表明 ElementType 所声明的元素是否为空、是否包含文本、是否包含子元素、还是既包含文本又包含子元素，通过 dt:type 指定该元素的数据类型，通过 order 指定该元素的子元素的排列规则，相应取值有：one、seq、many。element 元素是用于声明在 ElementType 中出现的元素，它需要同 ElementType 配合使用。

AttributeType 元素用于定义在 Schema 文档中出现的属性类型。其属性 dt:type 指定所声明属性类型的数据类型，可支持的数据类型包括：entity, entities, enumeration, id, idref, idrefs, nmtoken, nmtokens, notation, 和 string。default 属性可用于指定该属性类型的缺省取值。required 属性指定该属性对于引用它的元素是否是必须的。 attribute 元素实际上是对 AttributeType 声明的属性的引用，它也需要同 AttributeType 配合使用。

group 元素是用于将 XML 文档中的元素分组。通过属性 order 可指定该分组中的元素或子分组的顺序，通过 minOccurs 和 maxOccurs 分别指定该分组在 XML 实例文档中出现的最少和最多次数。

datatype 是 XML Schema 中一个重要元素，也是 XML Schema 的一大特色，它用于为 ElementType 和 AttributeType 指定数据类型。XML Schema 支持两种数据类型，一种是 XML 1.0 标准中定义的十种基本数据类型：entity, entities, enumeration, id, idref, idrefs, nmtoken, nmtokens, notation, 和 string；另外，还支持一些扩展数据类型，包括：bin.base64, bin.hex, boolean, char, date, dateTime, dateTime.tz, fixed.14.4, float, int, number, time, time.tz, i1, i2, i4, r4, r8, ui1, ui2, ui4, uri, uuid。

description 只能算是 XML Schema 中一个配角，它的主要作用是为 ElementType 和 AttributeType 元素提供描述信息。

Schema 的应用

由于 XML Schema 的种种优点，现在 Schema 取代 DTD 已成大势所趋。在这种情况下，国际上一些知名企业和组织审时度势纷纷在战略上向 XML Schema 倾斜，提供对 XML Schema 的支持。其中最为典型的当然要数微软的 BizTalk 和 xml.org 组织的注册/资源库。更加值得一提的是，微软在其浏览器软件 Internet Explorer 5.0 中率先提供对 Schema 的支持，当然，因为最终的 XML Schema 标准尚未正式推出，因此微软支持的 Schema 也只是过渡性"标准"，今后还会不断修改。

6 XML 中的命名空间

XML 命名空间并不是 XML1.0 标准的一部分，而是一个被称为"Namespace in XML"的独立标准。W3C 组织于 1998 年 2 月提出 XML 命名空间标准的第一个草案，直到 1999 年 1 月 14 日才正式发布为推荐标准。

一、 XML 命名空间的由来

简单说来，制定 XML 命名空间标准的初衷是为了解决 XML 文档中命名的冲突问 题。那么何为命名冲突问题呢？请看下面这个例子。

假设我们已有两个 XML 文档，"学生.xml"和"老师.xml"，如下所示：

```
<?xml version ="1.0" encoding = "GB2312"?>
<学生>
<姓名>李明</姓名>
<班级>三年级二班</班级>
<住址>135 楼 210 室</住址>
</学生>

<?xml version ="1.0" encoding = "GB2312"?>
<教师>
<姓名>李华</姓名>
<住址>432 楼 133 室</住址>
<电话>(021)32566178</电话>
</教师>
```

下面我们希望把这两个XML文档的内容结合成一个新的XML文档--"新学生.xml"，新XML文档如下所示：

```
<?xml version ="1.0" encoding = "GB2312"?>
<学生>
<姓名>李明</姓名>
<班级>三年级二班</班级>
<住址>135 楼 210 室</住址>
<班主任>
<教师>
<姓名>李华</姓名>
<住址>432 楼 133 室</住址>
<电话>(021)32566178</电话>
</教师>
</班主任>
</学生>
```

在这个新 XML 文档--"新学生.xml"中即出现了命名冲突的问题。"学生"元素的"姓名"子元素，其语义是"学生的姓名"，而"教师"元素的"姓名"子元素的语义是"班主任教师的姓名"。"住址"的元素名也有同样的命名冲突问题。

解决命名冲突问题的一个直接的方法是，给重名的元素或属性重新命名。例如将上例中"教师"的"姓名"元素改为"教师姓名"，然而这不是一种长期解决问题办法。在 XML 的实际应用中，人们常常为不同行业和领域用 XML 制定不同的语言标准，比如电子商务、远程教育、电子书都分别用 XML 制定了语言标准，然后针对不同的语言编写不同的模块化处理程序。通过重用现存的语言标准和处理程序，人们可以很快地定义出新的语言标准和处理程序。假

如我们通过重新命名的方法解决名称冲突问题，那么我们将面临着，针对原名称开发的应用程序不可再利用的危险。

解决名称冲突的一个比较好的解决方案是，给不同的语言赋以不同的名称空间，应用程序通过名称空间来区分一个元素到底来自于那一个语言。XML 命名空间就是对这种方案的具体实现。

二、 XML 命名空间的定义

XML 命名空间解决命名冲突问题采用的方法是所谓"两段式命名法"，其中第一段是代表特定命名空间的"命名空间前缀"，第二段是元素或属性原来的名字，两段之间用冒号"："分开。用 XML 命名空间重写后的"新学生.xml"文档如下：

```
<?xml version ="1.0" encoding = "GB2312"?>
<学生:学生 xmlns:学生 = http://www.xml.net.cn/学生
xmlns:班主任 = http://www.xml.net.cn/班主任>
<学生:姓名>李明</学生:姓名>
<学生:班级>三年级二班</学生:班级>
<学生:住址>135 楼 210 室</学生:住址>
<学生:班主任>
<班主任:教师>
<班主任:姓名>李华</班主任:姓名>
<班主任:住址>432 楼 133 室</班主任:住址>
<班主任:电话>(021)32566178</班主任:电话>
</班主任:教师>
</学生:班主任>
</学生:学生>
```

修改后的"新学生.xml"文档中，"姓名"和"住址"元素的名称前增加了"学生"和"班主任"这样的前缀，因此不再冲突。下面我们就对 XML 命名空间的定义作一说明。

XML 命名空间的定义由命名空间的声明、"合法名称"的定义及应用、命名空间的作用域三部分组成。

（1） XML 命名空间的声明

XML 命名空间的声明是通过保留属性"xmlns"来实现的。上例中的

```
<学生:学生 xmlns:学生 = http://www.xml.net.cn/学生
xmlns:班主任 = http://www.xml.net.cn/班主任>
```

就是命名空间声明。

命名空间声明有两种方式，即直接定义方式和缺省定义方式：

直接定义方式： xmlns: [命名空间前缀] = [命名空间名]

缺省定义方式：　xmlns = [命名空间名]

命名空间声明中，等号右边的属性值部分是一个 URI（Uniform Resource Identifier 统一资源标识符）引用，其功能是区分不同的命名空间。因此，这个 URI 引用被称为命名空间名，它应该具有唯一性和持久性。虽然该属性值使用了 URI，但其目的并不是要直接得到一个 schema 或 DTD，主要的目的在于标识特定的命名空间。

命名空间声明中，等号左边的属性名部分，如果有用冒号"："分隔开的"命名空间前缀"，就是直接定义方式，其中"命名空间前缀"是一个合法的 XML 名称。没有"命名空间前缀"的命名空间声明，就是缺省的命名空间声明。

命名空间声明将"命名空间名"与"命名空间前缀"绑定在一起。

（2） "合法名称"的定义和应用

在定义了命名空间的声明以后，对如何引用<命名空间前缀>构成新的元素名和属性名，需要再作进一步的统一规范，这就是所谓"合法名称"定义的由来。

"合法名称"由用西文冒号":"分开的前缀部分和本地部分组成，其中前缀部分和本地部分都是一个合法的 XML 名称。如："班主任:姓名"。

"合法名称"的前缀部分，规定必须是一个"命名空间前缀"，且这个命名空间前缀必须已经经过命名空间声明声明过，语法分析器会自动将其与声明中的 URI 引用相联系。冒号后的部分是该命名空间中定义的元素或属性名，提供了"合法名称"的本地部分。在用缺省方式声明命名空间时，由于"命名空间前缀"为空，因此，这时的"合法名称"只剩下本地部分。"新学生.xml"文档中，"http://www.xml.net.cn/学生"命名空间改为缺省命名空间后，如下所示：

```
<?xml version ="1.0" encoding = "GB2312"?>
<学生 xmlns = http://www.xml.net.cn/学生
xmlns:班主任 = http://www.xml.net.cn/班主任>
<姓名>李明</姓名>
<班级>三年级二班</班级>
<住址>135 楼 210 室</住址>
<班主任>
<班主任:教师>
<班主任:姓名>李华</班主任:姓名>
<班主任:住址>432 楼 133 室</班主任:住址>
<班主任:电话>(021)32566178</班主任:电话>
</班主任:教师>
</班主任>
</学生>
```

"合法名称"的应用主要有三种情况：

- 用于起始元素标记、结束元素标记和空元素标记。如前面例子所示。

- 用于属性的定义。
  例如：
  <?xml version ="1.0" encoding = "GB2312"?>
  <学生:学生 xmlns:学生 = http://www.xml.net.cn/学生>
  <学生:姓名>李明</学生:姓名>
  <学生:班级 学生:数字类型 = "中文">三年级二班</学生:班级>
  <学生:住址 学生:数字类型 = "阿拉伯">135 楼 210 室</学生:住址>
  </学生:学生>
- 用于 DTD 中的元素名和属性类型。
  例如：
  <?xml version="1.0" encoding="GB2312"?>
  <!ELEMENT 学生:学生 (学生:姓名, 学生:班级，学生:住址)>
  <!ATTLIST 学生:学生 xmlns:学生
  CDATA #FIXED "http://www.xml.net.cn/学生">
  <!ELEMENT 学生:姓名 (#PCDATA)>
  <!ELEMENT 学生:班级 (#PCDATA)>
  <!ELEMENT 学生:住址 (#PCDATA)>

（3） 命名空间的作用域

所谓命名空间的作用域范围是指，一个命名空间声明可以作用到哪些元素和属性。一般可以认为命名空间声明，能够作用到说明它的元素和该元素的所有内容元素，除非被其他命名空间声明所覆盖。再一次修改"新学生.xml"文档，将"http://www.xml.net.cn/班主任"命名空间移到"教师"元素中，如下所示：

```
<?xml version ="1.0" encoding = "GB2312"?>
<学生 xmlns = http://www.xml.net.cn/学生>
<姓名>李明</姓名>
<班级>三年级二班</班级>
<住址>135 楼 210 室</住址>
<班主任>
<班主任:教师 xmlns:班主任 = http://www.xml.net.cn/班主任>
<班主任:姓名>李华</班主任:姓名>
<班主任:住址>432 楼 133 室</班主任:住址>
<班主任:电话>(021)32566178</班主任:电话>
</班主任:教师>
</班主任>
</学生>
```

缺省命名空间"http://www.xml.net.cn/学生"的作用域在"教师"元素以外的地方，而"http://www.xml.net.cn/班主任"命名空间的作用域在"教师"元素内，包括"教师"元素本身。

与 XML 命名空间相关的主要概念，讨论到这里基本上可以结束了。由于篇幅的关系，与 XML 命名空间相关的一些曾经引起争论的问题，如"命名空间与 DTD"，就不在这里讨论了。XML 命名空间已经在 XSLT、Xlink 等标准中得到应用，它已经成为 XML 标准家族不可或缺的一员。

7 文档显示与样式单（上）

概述

XML 关于文档浏览的基本思想是将数据与数据的显示分别定义，XML 文档本身不涉及各种数据的具体显示方式，文档的显示实际上是通过一个外部样式表，又称为样式单来描述的。

样式单(Style Sheet)是一种描述结构文档表现方式的文档，它既可以描述这些文档如何在屏幕上显示，也可以描述它们的打印效果，甚至声音效果。与传统使用的<font>等标记相比，样式单有许多突出的优点：

- 表达效果丰富。
- 文档体积小。
- 便于信息检索。
- 可读性好。

迄今为止，W3C 已经给出了两种样式单语言的推荐标准，一种是层叠样式单 CSS（Cascading Style Sheets），另一种是可扩展样式单语言 XSL（eXtensible Stylesheet Language）。

样式单一般不包含在 XML 文档内部，以独立的文档方式存在。如果对一个 XML 文档施加某一个样式单，可在 XML 文档中使用标记:

<?xml-stylesheet type="..." href="..."?>

予以声明，表示该 XML 文档的显示效果由所引用的样式单决定。例如：

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

表明 mystyle.css 决定 XML 文档的显示样式。

而

<?xml-stylesheet type="text/xsl" href="mystyle.xsl"?>

表明该 XML 文档使用 XSL 样式单 mystyle.xsl。

内容与形式相分离使 XML 文档更偏重于数据本身，而不受显示方式的细枝末节的影响。而且通过定义不同的样式单可以使相同的数据呈现出不同的显示外观，实现 XML 数据的可重用性。

层叠样式单 CSS

层叠样式单 CSS 是一种样式描述规则，目前 W3C 有两个推荐标准，CSS1 和 CSS2。CSS2 是在 CSS1 的基础上制定的，基本上涵盖了 CSS1，并在 CSS1 的基础上增加了媒体类型、特性选择符、声音样式等功能，并对 CSS1 原有的一些功能进行了扩充。

CSS 最初在 HTML 中得到应用，它的基本思想是为结构文档中的各个标记定义出相对应的一组显示样式。

利用 CSS 人们可以改变文档中元素的显示属性，如位置、颜色、背景、边空、字体、排版格式等等。

定义 CSS 的基本格式如下：

Selector {property:value; property:value; ...}
Selector： 选择符，被施加样式的元素，可以是标记（tag）、类(class)、标识(id)等。
Property： 样式属性，可以是颜色、字体、背景等等。
value： 样式属性取值，决定样式结果。

下面举一个简单的 XML 文档和它的样式单例子：

```
<?xml version="1.0" encoding="gb2312" ?>
<?xml-stylesheet type="text/css" href="mystyle.css"?>
<information xmlns:student="customer.dtd">
客户信息表
<customer>
<name>王大卫</name>
<id>001</id>
<address>海淀大街</ address >
<telephone>62875566</telephone>
</customer>
<customer>
<name>李小丽</name>
<id>002</id>
<address>东四十条</address>
<telephone>62173425</telephone>
</customer >
</information>
```

上面是一段 XML 文档，描述的是一个客户信息表，其中有两个客户的资料。有关客户信息表的 DTD 定义如下：

```
<!ELEMENT information(customer*) >
<!ELEMENT customer(name+, id, address, telephone?) >
<!ELEMENT name(#PCDATA) >
<!ELEMENT id(#PCDATA) >
<!ELEMENT address(#PCDATA) >
<!ELEMENT telephone(#PCDATA) >
```

下面看一下如何利用 CSS 将上述资料显示在浏览器中。

样式一：

```
information,customer
{
font-size:10.5pt;
```

```
font-weight:bold;
color:black;
display:block;
margin-bottom:5pt;
}

id,address,telephone
{
font-weight:normal;
font-size:10.5pt;
display:block;
color:blue;
margin-left:20pt;
}

name
{
font-weight:bold;
font-size:10.5pt;
display:block;
color:red;
margin-top:5pt;
margin-left:8pt;
}
```

样式一的浏览结果见图 1。

图 1 样式一的浏览效果（IE）

样式二：

```
information
{
display: table-caption;
text-align: center;
padding:25px;
}

customer
{
display:table-row;
}

name,id,address,telephone
```

```
{
display:table-cell;
padding:5px;
}

name
{
color:red;
font-weight:bold;
}
```

样式二的浏览结果见图 2。

图 2 样式二浏览结果（Mozilla）

从例中可以看出，同样的数据资料，CSS 样式单却赋予了它不同的表现方式。 样式二的浏览结果是用浏览器 NetScape 的升级版本 Mozilla 得到的，使用 IE 得不到这样的结果，原因是 IE 对 CSS2 的支持不完全。

8 XML 文档显示与样式单（下）

可扩展样式单语言 XSL

XSL（eXtensible Stylesheet Languge）是描述 XML 文档样式信息的一种语言，是由 W3C 制定的。上述的层叠样式单 CSS，是一种静态的样式描述格式，其本身不遵从 XML 的语法规范。而 XSL 本身就是一个 XML 文档，系统可以使用同一个 XML 解释器对 XML 文档及其相关的 XSL 文档进行解释处理。XSL 最近的一个草案于 2000 年 3 月提出，尚未成为正式标准。

XSL 由两大部分组成：一部分描述如何将一个 XML 文档转换为可浏览或可输出的格式；另一部分则定义格式对象 FO（fomatted object）。在输出时，首先根据 XML 文档构造源树，然后根据给定的 XSL 将这个源树转换为可以显示的结果树，这个过程称作树转换，最后再按照 FO 解释结果树，产生一个可以在屏幕上、纸上、语音设备或其他媒体中输出的结果，这个过程称作格式化。

到目前为止，W3C 还未能出台一个得到多方认可的 FO，但是描述树转换的这一部分协议却日趋成熟，已从 XSL 中分离出来，另取名为 XSLT（XSL Transformations），其正式推荐标准于 1999 年 11 月 16 日推出，现在一般所说的 XSL 大都指的是 XSLT。与 XSLT 一同推出的还有其配套标准 XPath，这个标准用来描述如何识别、选择、匹配 XML 文档中的各个构成元件，包括元素、属性、文字内容等。

使用 XSL 显示 XML 的基本思想是通过定义模板将 XML 源文档转换为带样式信息的可浏览文档。最终的可浏览文档可以是 HTML 格式、带 CSS 的 XML 格式及 FO 格式。

在 XML 中使用如下语句声明 XSL 样式单：

<?xml-stylesheet type="text/xsl" href="mystyle.xsl"?>

XSL 在网络中的应用大体分为两种模式：

1．服务器端转换模式

在这种模式下，XML 文件下载到浏览器前先转换成 HTML，然后再将 HTML 文件送往客户端进行浏览。有两种方式：

- 动态方式；即当服务器接到转换请求时再进行实时转换，这种方式无疑对服务器要求较高。
- 批量方式；实现将 XML 用 XSL 转换好一批 HTML 文件，接到请求后调用转换好的 HTML 文件即可。

2．客户端转换模式

这种方式是将 XML 和 XSL 文件都传送到客户端，由浏览器实时转换。前提是浏览器必须支持 XML+XSL。

下面是 XSLT 的一个简单的例子，通过剖析这个例子，即可掌握 XSLT 的基本功能。

例子仍然使用前述的顾客信息表的例子，其 XML 文档与 DTD 文档请参照 CSS 部分，只是 XML 文档的引用样式要改为： 使用 Java 构建处理 XML 可扩展性的应用.doc

<?xml-stylesheet type="text/xsl" href="mystyle.xsl"?>

mystyle.xsl 文档如下：

```
<?xml version="1.0" encoding="gb2312" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl"
xmlns="http://www.w3.org/TR/REC-html40">
<xsl:template>
<xsl:apply-templates/>
</xsl:template>

<xsl:template match="/">
<HTML>
<HEAD>
<TITLE>顾客信息表</TITLE>
<STYLE>
CAPTION{font-size:15pt; font-weight:bold; color:blue }
</STYLE>
</HEAD>
<BODY>
<xsl:apply-templates select="information"/>
</BODY>
```

```
</HTML>
</xsl:template>

<xsl:template match="information">
<TABLE BORDER="1">
<CAPTION>顾客信息表</CAPTION>
<THEAD>
<TD> <B>姓名</B> </TD>
<TD> <B>籍贯</B> </TD>
<TD> <B>年龄</B> </TD>
<TD> <B>电话</B> </TD>
</THEAD>
<xsl:for-each select="customer" order-by="name">
<TR>
<TD><B><xsl:value-of select="name"/></B></TD>
<TD><xsl:value-of select="id"/></TD>
<TD><xsl:value-of select="address"/></TD>
<TD><xsl:value-of select = "telephone"/></TD>
</TR>
</xsl:for-each>
</TABLE>
</xsl:template>
</xsl:stylesheet>
```

将例中的 XML 文档用 XSL 样式转换为 HTML 文档的步骤是：先用 XML 解释器将 XML 文档解释成 DOM 对象，相当于建立了原文档的一个节点树。然后用 XML 解释器解释 XSL 文档，用模板匹配的方法去遍历 XML 节点树，将树中的节点按模板的设定转换为模板指示的显示语言。

为看懂例中的 XSL 源码，首先介绍一下 XSL 主要的几条语句：

Xsl:stylesheet
声明语句

Xsl:template
相当于编程中函数的概念

Xsl:template match = ""
相当于函数调用，去匹配引号中指定的节点

Xsl:apply-templates
应用模板函数

Xsl:apply-templates select =""
应用模板函数的调用，跳转到引号中指定的模板

xsl:for-each select = ""
循环语句，遍历与引号中的属性值相同的节点

xsl:value-of select = ""
赋值语句，取出引号中指定的属性值

知道了上面的语句含义，现在就可以分析一下上面的源代码了：

1.用<xsl:template> <xsl:apply-templates/> </xsl:template>声明 XSL 模板，并调用该模板。

2.系统最先匹配 XML 原树的根节点，根节点用"/"表示。将根节点转换为带有样式信息的 HTML 文档的开头的一段代码：

<HTML>
< HEAD>
< TITLE>顾客信息表< /TITLE>
< STYLE>
... ...
< /STYLE>
< /HEAD>
< BODY>

然后再调用另一个节点模板，相当于根节点模板的子函数，处理根节点下的子节点"顾客信息表"的模板转换。之后再把 HTML 的结尾部分添在根节点模板的后面，使其成为一个完整的 HTML 文档。

3.匹配学生花名册子节点。建立一个表头为"姓名"、"序号"、"地址"、"电话"的表格，并循环遍历"customer"节点，将每个"customer"节点的上述四个属性值取出，放在表格中。读者可自己仔细研究一下这几句，笔者不再赘述。

如何选择样式单

CSS 和 XSL 同属于样式单，它们的区别表现在于 CSS 既可用于 HTML，也可用于 XML。但 XSL 是专门针对 XML 提出的，它不能处理 HTML 文档。

另外 XSL 是一种转换的思想，它最终将 XML 文档转换为另一种可用于输出的文档，而 CSS 则没有任何转换动作，在整个过程中没有任何新码产生。

XSL 中 90%的样式规定在 CSS 中都有定义，但仍然有一些效果是 CSS 无法描述的，必须使用 XSL 不可。这些功能包括文本的置换、根据文本内容决定显示方式、文档内容排序等，都是 XSL 所独有的。 再者，XSL 遵从 XML 的语法，而 CSS 的语法自成体系。

选择样式单还要考虑不同浏览器对样式单的支持程度。目前 IE5 与 Mozilla（即 Netscape5.0）都支持 CSS，但支持的程度都有限。至今为止，IE5 尚不能完全支持 CSS1，即便是支持的部分也存在很多错误，对于 CSS2 也只提供部分支持。Mozilla 在对 CSS 的支持上已经优于 IE5，它采用新一代的 Raptor/Gecko 引擎技术，已经能够完全支持 CSS1，但对 CSS2 的支持计划尚不明朗。对 XSLT 而言，只有 IE5 支持，Netscape5 并不支持。

另外，为弥补传统页面在表现力上的不足，W3C 又公布了很多利用 XML 定义的新标准，如 SVG，MathML。其中 SVG 针对于矢量图形的显示与传输，MathML 针对于数学公式的显示与传输，对于这一类 XML 文档，需要另外开发单独的浏览器或插件，目前还没有达到实用的阶段。

综上所述，一个 XML 文档的显示方式可以归纳为三种方式：即利用 CSS 显示、利用 XSL 转化为 FO 显示、、利用 XSL 转化为带有 CSS 的 HTML 文档显示。

9 XML 的字符编码

构成书面语言的基本元素，在拼音文字中被称为"字母"，在象形文字中被称为"单字"。在计算机信息处理中，"字母"、"单字"和一些印刷及科学计算符号一起被统称为"字符"。计算机中的"字符"都是用一个二进制数字来表示的，这个二进制数字称为"字符"的编码。针对不同需求选择的"字符"集合，被称为某某字符集。对这些字符集进行不同的二进制编码，便形成了各种各样字符集编码。

XML 标准规定，XML 文件使用 Unicode 字符集编码。虽然 Web 是国际性的，但由于 HTML 对跨文种编码支持的不足，极大地限制了跨文种网页的制作和浏览。XML 通过对双字节 Unicode 字符集和其压缩表示方式 UTF-8、UTF-16 提供完全支持，使 Web 的国际性在数据表现和数据交换层上真正得以实现，因为 Unicode 几乎包括了世界上所有现代语言通常使用的每一种字符。

下面我们就对 Unicode 以及 XML 对多文种的支持做一简单介绍。

Unicode 简介

由于在计算机应用领域中存在着几十种互不相同的字符集，当在使用不同字符集的计算机系统之间进行信息交换时，人们可能会得到一些莫名其妙的东西（那些经常上网，经常不得不在各种字符编码之间来回切换的人对此可能深有感触）。紊乱的字符编码给信息交换以及软件开发商等带来了极大的不便。人们急需一种得到大家认可的并且涵盖了全世界各种文字的字符集。显然，建立这样的一种字符集是十分困难的。不过，这方面的努力一直在进行，Unicode 就是这种努力的结果之一。

为了将成千上万的文字统一到同一个编码机制之下，在兼顾经济的原则下，不管是东方文字还是西方文字，在 Unicode 中一律用两个字节来表示。也就是说，Unicode 是一种双字节编码机制的字符集，使用 0-65535 之间的双字节无符号整数对每个字符进行编码。这样，在 Unicode 字符集中，至少可以定义 65536 个不同的字符，足以应付目前绝大多数场合的需要。

目前，常用的 Unicode 编码方式有两种：UTF-8 以及 UTF-16。

UTF-8 是一种不等幅的编码方式，UTF-8 编码的字节长度取决于所要编码的字符在 ISO 10646 中的编码值。在 UTF-8 中，不同的字符，可能需要 1-6 个字节来进行编码。对于单字节的 UTF-8 编码，该字节的最高位为 0，其余 7 位用来对字符进行编码（等同于 ASCII 码）。对于多字节的 UTF-8 编码，如果编码包含 n 个字节，那么第一个字节的前 n 位为 1，

第一个字节的第 n+1 位为 0，该字节的剩余各位用来对字符进行编码。在第一个字节之后的所有的字节，都是最高两位为"10"，其余 6 位用来对字符进行编码。

UTF-16 也是 Unicode 的变形表示方式的一种。它的目的是维持双八位的编码方式，同时也用一些特殊的双八位来表示 ISO 10646 中非基本多文种平面（BMP）中的一些字符。这种用来表示非 BMP 字符的方法在 Unicode 中称作代理对机制。

代理对的编码机制以及原先不需要代理对的六万三千多个基本 Unicode 码，合起来叫做 UTF-16。也就是说 UTF-16 基本上就是 Unicode 双字节编码的实现，再加上一个应付未来扩充需要的编码机制。

UTF-16 编码遵循下述原则：

- 对于字符编码值小于 0x10000 的字符，则直接把编码值转化为一个相等的 16 位整数。
- 对于字符编码值在 0x10000 到 0x10FFFF 之间的字符，将用一个 0xD800 到 0xDBFF（代理高部）之间的 16 位数，后面紧跟一个 0xDC00 到 0xDFFF（代理低部）之间的 16 位数来表示。
- 字符编码值大于 0x10FFFF 的字符将不能用 UTF-16 来表示。

XML 对多文种的支持

在 XML1.0 标准中规定，XML 分析器必须支持以 UTF-8 或 UTF-16 编码的 Unicode 字符串，当 XML 声明中没有特别指明当前文档的编码方式（encoding）时，则一律以 Unicode 看待。分析器会自动识别出当前文档是 UTF-8 的还是 UTF-16。也就是说，XML 使用的缺省字符集是 Unicode 字符集。由于 Unicode 是一种通用字符集编码，因此，能够提供对多文种的支持。也就是说，在同一个 XML 文档中，几乎可以包含全世界的各种文字。

除非明确指出，否则 XML 处理器假设 XML 文档是用 UTF-8 编码的。因为 UTF-8 包括作为子集的 ASCII 码，所以 XML 处理器也可以轻易地对 ASCII 文本进行语法分析。

除了 UTF-8, XML 处理器还要能够识别 UTF-16。通过在文本实体的开端处寻找字节顺序记号，XML 处理器可以辨认 XML 文档是 UTF-8 编码的还是 UTF-16 编码的。

如果不能把文本转变成 UTF-8 或者 UTF-16，那么可以把文本用本地字符集代替，但是这时要告诉 XML 处理器是哪一种字符集。因为各种 XML 处理器不能保证处理其它编码形式，所以应该把这种方法作为一个最后的手段来使用。

为了提醒 XML 处理器正在使用非 Unicode 编码，可以在该文件开端的 XML 声明中包括一个 encoding 属性。例如，为了指定在缺省情况下整个文档使用 GB2312（除非在嵌套实体中被另一个进程指令覆盖）编码，使用如下的 XML 声明：

```
< ?xml version = "1.0" encoding = "GB2312"? >
```

一个完整的小例子如下所示：

```
< ?xml version = "1.0" encoding = "GB2312"? >
<  测试  >
```

XML 可以支持多文种。这个例子使用的是中文。

< /测试 >

下表列出了当今使用的大多数常用字符集的规范名称，它们可以在XML的encodeing属性中给出。对于在列表中没有找到的编码形式，可以查询由Internet分配数字管理处（IANA）维护的正式列表http://www.isi.edu/in-notes/iana/assignments/character-sets。

规范名称	语言国家
US-ASCII	英语
UTF-8	压缩的 Unicode
UTF-16	压缩的 UCS
ISO-10646-UCS-2	原 Unicode
ISO-10646-UCS-4	原 UCS
ISO-8859-1	Latin-1，西欧语言
ISO-8859-2	Latin-2，东欧语言
ISO-8859-3	Latin-3，南欧语言
ISO-8859-4	Latin-4，北欧语言
ISO-8859-5M	ASCII 加西里尔语言
ISO-8859-6	ASCII 加阿拉伯语言
ISO-8859-7	ASCII 加希腊语言
ISO-8859-8	ASCII 加希伯来语言
ISO-8859-9	Latin-5，土耳其语言
ISO-8859-10	Latin-6，西欧
ISO-8859-13	Latin-7，ASCII 码加波罗的海 周边语言和独特的拉托维亚语
ISO-8859-14	Latin-8，ASCII 码
ISO-8859-15	Latin-9，Latin-10，西欧
ISO-2022-JP	日语
Shift-JIS	日文 Windows
EUC-JP	日文 Unix
Big5	汉语，中国台湾
GB2312	汉语，中国大陆
KOI8-R	俄语
ISO-2022-KR	朝鲜语
EUC-KR	朝鲜语，Unix
ISO-2022-CN	汉语

10 XML 链接语言

本讲和下一讲将讨论 XML 规范的另一个重要组成部分--XML 链接规范，XML 链接规范分为三个部分：XLink 语言、XML Base 和 XPointer 语言。其中，XLink 语言用于建立资源之间的链接；XML BASE 提供与 HTML 中 BASE 标记相似的功能，用于指定相对 URL 的绝对路径；XPointer 用于定位 XML 文档中的片断。需要指出的是，XML Base 和 XPointer 语言的使用并不局限于链接，它们的主要功能在于资源的定位。本讲主要介绍 XLink 语言和 XML Base，XPointer 语言将在下一讲介绍。

当前的主流浏览器 Netscape Navigator 和 Internet Explorer 对 XLink 的支持非常弱，在 Netscape Navigator 6.0 的预览发行版中可以发现对简单 XML 链接的支持，但是其支持的部分并不完全符合最新的规范；Internet Explorer 5.5 没有提出对 XLink 的支持。如果要在浏览器中显示链接，可以利用 XSLT 将 XML 链接转化为 HTML 链接加以显示。

XML 链接与 HTML 链接

在 HTML 中，常用标记表示链接。通过标记，可以从一个文件链接到另一个文件，或者链接到文件的某一部分。另外，标记<IMG>和<OBJECT>允许图形等对象直接嵌入文件。XML 链接完全不同，它没有专门的链接元素，需要通过指定元素属性来表示链接，只要元素包含 xlink:type 属性，且取值为"simple"或"extended"，该元素就是链接元素，其中 xlink 是代表 XLink 命名空间的前缀，当前版本的 URI 是"http://www.w3.org/1999/xlink"，根据 xlink:type 属性的取值，可以将 XML 链接划分为简单 XML 链接和扩展 XML 链接。简单 XML 链接的 xlink:type 固定取值为"simple"，扩展 XML 链接的 xlink:type 固定取值为"extended"。

简单 XML 链接

简单 XML 链接与 HTML 链接非常相似，它在链接元素和目标资源间建立链接。下面是一个简单 XML 链接的元素定义，其中包含 xlink 前缀的属性都是链接属性，在链接属性外，链接元素还可以有任意属性和任意内容。

```
<!ELEMENT SimpleLink (#PCDATA)>
<!ATTLIST SimpleLink
    xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink"
    xlink:type (simple) #FIXED "simple"
    xlink:href CDATA #REQUIRED
    xlink:role CDATA #IMPLIED
    xlink:title CDATA #IMPLIED
    xlink:show (embed | replace | new) "replace"
    xlink:actuate (onLoad | onRequest) "onRequest"
>
```

xlink:href 属性是目标资源的 URL，可以是绝对 URL、相对 URL 或文件片段，相对 URL 必须接收 XML Base 中指定的绝对路径，XLink 处理程序将 XML Bae 和相对 URL 合并起来形成目标资源的 URL，而文件片段则由 XPointer 指定。xlink:role 和 xlink:title 是用于描述目标资源的属性，统称为语义属性。xlink:role 用于机器阅读，如搜索引擎的爬虫等，而 xlink:title 用于人工阅读；xlink:show 和 xlink:actuate 用于描述链接激活时的行为，统称为行为属性。xlink:show 表示链接激活时的目标资源的显示环境，取值"embed"表示在当前窗口嵌入显

示，"replace"表示在当前窗口显示目标资源，替换原来的显示内容，"new"表示新开窗口显示目标资源，xlink:actuate 是指链接的激活时机，"onLoad"是指文件加载时直接激活链接资源，"onRequest"是指在文件加载后，用户发出链接激活的命令才激活，如用户点击了链接等。

利用上面的元素声明，定义元素 SimpleLink 的实例如下。

```
<SimpleLink >
xmlns:xlink = "http://www.w3.org/1999/xlink"
xlink:href = "http://www.xlinksample.com/simplelink.xml"> This is a simple xlink!
</SimpleLink>
```

扩展 XML 链接

扩展 XML 链接可以在多个资源（尤其是只读资源）之间建立多向的链接。扩展 XML 链接元素的构造，很大程度取决于用户，但通常包括一个资源集合和一个连接集合，连接集合元素表示资源集合元素间的连接。资源集合中可以包括本地资源和远程资源，如果资源是 XML 链接元素的组成部分，该资源是本地资源；否则就是远程资源，如其他的 XML 文件或本文件的其它元素。

下例是一个扩展 XML 链接的 DTD 声明，其中 ExtendedLink 元素是扩展 XML 链接元素，它包含零到多个 Local 元素、Remote 元素和 Arc 元素。其中 Local 元素表示本地资源；Remote 元素表示远程资源；Arc 元素表示这些资源间的连接，称为链接弧。需要指出的是，这些元素的名称并不重要，真正对链接起作用的并不是元素名称，而是这些元素所具有的 XLink 属性。

```
<!ELEMENT ExtendedLink (Local | Remote | Arc)*>
<!ATTLIST ExtendedLink
----xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink"
----xlink:type (extended) #FIXED "extended"
----xlink:role CDATA #IMPLIED
----xlink:title CDATA #IMPLIED>
<!ELEMENT Local ANY>
<!ATTLIST Local
----xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink"
----xlink:type (resource) #FIXED "resource"
----xlink:role CDATA #IMPLIED
----xlink:title CDATA #IMPLIED
----xlink:label NMTOKEN #IMPLIED>
<!ELEMENT Remote ANY>
<!ATTLIST Remote
----xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink"
----xlink:type (locator) #FIXED "locator"
----xlink:href CDATA #REQUIRED
----xlink:role CDATA #IMPLIED
```

```
----xlink:title CDATA #IMPLIED
----xlink:label NMTOKEN #IMPLIED>
<!ELEMENT Arc (#PCDATA)>
<!ATTLIST Arc
----xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink"
----xlink:type (arc) #FIXED "arc"
----xlink:from NMTOKEN #IMPLIED
----xlink:to NMTOKEN #IMPLIED
----xlink:arcrole CDATA #IMPLIED
----xlink:title CDATA #IMPLIED
----xlink:show (embed | replace | new ) "replace"
----xlink:actuate (onLoad | onRequest) "onRequest">
```

从上面我们可以看出，无论是链接元素自身、资源元素还是链接弧元素，都包含了 XLink 命名空间的属性，另外它们也可以包含其他内容和属性。对于本地资源元素，xlink:type 属性固定取值为"resource"，远程资源元素的 xlink:type 属性固定取值为"locator"，链接弧的 xlink:type 属性固定取值为"arc"。其中 xlink:role（xlink:arcrole）和 xlink:title 属性的含义与简单 XML 链接中相类似，只不过描述的是自己所在的资源元素或链接弧。Remote 元素中的 xlink:href 属性是指远程资源的 URL。Local 和 Remote 中的 xlink:label 是指资源的标识，在 Arc 元素中的 xlink:from 和 xlink:to 属性的取值必须是某个 xlink:label 的取值，表示从 xlink:from 标识的资源到 xlink:to 标识的资源的链接。由于 xlink:label 并不是唯一标识，所以一个 Arc 元素可能表示了多个资源间的链接。xlink:from 和 xlink:to 属性都可以省略，表示链接包含的所有资源都参与链接，xlink:from、xlink:to 和 xlink:label 称为遍历属性。

利用上面的元素声明，定义元素 ExtendedLink 的实例如下。

```
<ExtendedLink xmlns:xlink = "http://www.w3.org/1999/xlink">
<Local
xlink:role="department" xlink:title = "系别" xlink:label="department">
计算机系
/Local>
Remote
xlink:href = "http://www.xlinksample.com/teacher.xml"
xlink:role = "teacher" xlink:title = "老师：张三" xlink:label = "teacher">
张三
</Remote>
<Remote
xlink:href = "http://www.xlinksample.com/DataStructure.xml"
xlink:role = "course" xlink:title = "数据结构" xlink:label = "course">
数据结构
</Remote>
<Remote
xlink:href = "http://www.xlinksample.com/OperatingSystem.xml"
xlink:role = "course" xlink:title = "操作系统" xlink:label = "course">
操作系统
```

```
</Remote>
<Arc
xlink:from = "teacher" xlink:to = "department"
xlink:arcrole = "belonged" xlink:title = "属于">
教师所属系
</Arc>
<Arc
xlink:from = "teacher" xlink:to = "course"
xlink:arcrole = "teach" xlink:title = "教师开课">
教师开课安排
</Arc>
</ExtendedLink>
```

XML Base

XML Base 是用于指定相对 URL 的绝对路径，与 HTML 中的 BASE 标记功能相似，不同的是，XML Base 的指定是通过元素的 xml:base 属性指定的，前缀 xml 是用于表示命名空间"http://www.w3.org/XML/1998/namespace"。XML Base 的作用范围是包括其所有后代元素的整个元素，除非在后代元素指定了新的 XML Base。

11 XPointer 语言

XPointer 语言的主要功能是在 XML 文件中定位片段（fragement），在 XML 链接中，通常将它加到 URL 的结尾，更明确地表示目标资源。但是 XPointer 语言的使用并不局限于 XML 链接，它可以用于在需要文档内部定位的任何地方，比如在可视化的 XML 编辑器中用于描述用户选择的节点或字符串。

XPointer 语言概述

HTML 中的文件内部定位非常简单，就是在目标文件中插入一个命名锚（named anchor），然后用链接元素<A>的 href 属性指定链接的位置即可。但这种机制存要求你同时控制目标文件和开始文件，而这并不总是成立。由于 XML 是结构化的文件，借助文件结构进行内部定位就成为可能，这就是 XPointer 语言，它支持在 XML 文件中定位元素、属性、字符串等内部结构。

XPointer 语言基于 XSLT 中的 XPath，支持完整形式（Full XPointers）、无修饰名称（bare names）或子节点序列（Child Sequences）三种形式以定位片段。完整形式的 XPointer 的形式为 xpointer(表达式)，其中表达式用于定位计算，得到需要的资源片断，所有的定位计算都基于一个已知节点，该节点称为上下文节点。一般而言，最初始的上下文节点总是文档中的确定位置，如文档的根节点（注意：根节点不是根元素，它是一个抽象的节点，是根元素的父节点，用"/"表示）、文档中具有确定 ID 值的元素（用函数 id()表示）、当前元素（用函数 here()表示）等。无修饰名称只有一个名称，表示文件中 ID 等于指定名称的元素，它提供了与 HTML 文件兼容的方式以定位文档片段。子节点序列由名称、一系列数字和"/"组成的序列，其中"/"用于分隔名称和数字，数字 n 表示前面定位元素的第 n 个直接子元素，

子节点序列必须以"/1"或名称打头，表示该序列从根元素或指定 ID 的元素开始。实际上，后面两种形式都是第一种形式的简写，下面是几种形式的 XPointer 的例子。

xpointer(/child::lang/descendent::body[position()<=2])
xpointer(id("location")) 简写为：location
xpointer(/child::*[position()=1]/child::*[position()=2]) 简写为： /1/2
xpointer(id("loc")/child::*[position()=3]/child::*[position()=4]) 简写为：loc/3/4

XPath 位置步

与 XPath 语言一样，XPointer 语言中的表达式是多个位置步形成的一个位置路径，每个位置步间用"/"分隔，前面位置步的运算结果构成后面位置步的上下文节点。位置步的形式为 axis::node-test[prediates]，它包括三个部分：

- 关键字，有且仅有一个，它表示结果节点与上下文节点的关系；
- 节点测试，有且仅有一个，它表示结果节点的类型或名称；
- 谓词，零到多个，它可以是用于限制结果的任意表达式。

在下面的例子中，第一个"/"表示文档的根节点，"child::lang"为第一个位置步，"descendent::body[position()<=2]"为第二个位置步，这两个位置步链接形成一个位置路径。在第二个位置步中，"descendent"是关键字，"body"是节点测试，"position()<=2"是谓词。

/child::lang/descendent::body[position()<=2]

关键字的运算基于上下文节点，如果上下文节点有多个，它将对上下文节点逐一进行计算。下面列出了用于 XPointer 中的 XPath 关键字。

（1） ancestor：上下文节点的所有祖先节点，即包含上下文节点的所有节点。显然，根节点没有祖先节点。

（2） ancestor-or-self：上下文节点及其所有祖先节点。

（3） attribute：上下文节点的所有属性，如果上下文节点不是元素类型，结果集为空，attribute 关键字可以简写为"@"。

（4） child：上下文节点的直接子节点，包括元素、处理指令、注释、文本，但不包括属性。child 关键字是缺省的。

（5） descendant：上下文节点的所有后代节点，不包括属性和命名空间节点。简写为"//"。

（6） descendant-or-self：上下文节点及其所有后代节点。

（7） following：处于上下文节点后的节点。第一个节点从上下文节点结束标记后的第一个开始标记开始。返回元素顺序按文件顺序排列。

（8） following-sibling：上下文节点后，并与上下文节点具有相同父节点的节点。按文件顺序计算。

（9） parent：上下文节点的父节点，即直接包含上下文节点的节点。如果没有节点测试，可以简写为".."

（10） preceding：开始于整个上下文节点前的所有节点，不包括祖先节点。返回元素顺序按文件逆序排列。

（11） preceding-sibling：开始于上下文节点前，并与上下文节点具有相同父节点的节点。

（12） self：上下文节点。可以简写为"."

节点测试用于测试节点类型或名称，只有那些满足节点测试的节点才会保留在初始的结果集中，以待下一步的谓词过滤。在很多情况下，节点测试是一个名称，但并不总是如此，其它的节点测试还有：

（1） *。通配符"*"表示所有的元素，它不考虑元素的名称，通配符"*"只选择元素类型。如果你希望选择所有节点，可以用 node()表示。

（2） node()。节点测试 node()表示所有类型的节点，包括元素、注释、文本、处理指令等。

（3） comment()。表示类型是注释的节点。

（4） processing-instruction()。表示类型是处理指令的节点。它还可以接收参数，表示指定名称的处理指令。

（5） text()。表示类型是文本的节点。

谓词是对结果集进行过滤的表达式，一般是布尔型，其他类型的表达式可以转化为布尔型。XPointer 中，表示位置的谓词是最常用的谓词，包括几个谓词函数 position()、last()等。position()表示节点在结果集中的序号，序号从 1 开始，一般而言，结果集中的节点是按照相对于上下文节点的距离排序。position()函数常与关系运算符（=、<>、<、<=、>、>=）、数字连用，构成关系表达式，表示结果结点满足此关系表达式。谓词[position()=n]常简写为[n]。last()表示集合中的节点数目，也可与 position()函数连用。

XPointer 扩展

前面介绍的定位方式来源于 XPath，它们同时适用于 XPath 和 XPointer，但是 XPointer 对 XPath 进行了扩展。其扩展主要包括：

（1） 将 XPath 的节点(node)扩展为位置(location)，位置可以是节点、点和区域；点是某个节点或字符前后的位置，区域是两个点之间的所有结构和数据。

（2） 增加了函数 here() 和 origin()用于进行位置定位，表示当前元素，其中 here()用于本地资源，origin()用于远程资源；

（3） 增加函数 point()以定位点。如果点的包含结构可以包含节点，这种类型的点称为节点点（node point），此时索引按照节点进行，0 表示第一个节点前的位置，n 表示第 n 个节点前的位置；如果点的包含结构不能包含节点，这种类型的点称为字符点（character point），此时索引按照字符进行。

（4） 增加了区域表达式，用于生成区域；其形式为 xpointer(start-point/range-to(end-point))，其中 start-point 是一个位置路径，表示区域的起始点，end-point 是另一个位置路径，表示区域的结束点，range-to()是一个函数，其参数是位置路径，表示整个位置路径的计算结果是一个区域。

（5） 增加函数 string-range()，用于生成字符串；其形式为 string-range(location-set,substring,index,length)，其中 location-set 表示字符串匹配的范围，substring 表示匹配字符串，index 表示相对于匹配位置的起始点 length 表示返回字符串的长度；

（6） 增加函数 start-point() 和 end-point()，用于定位节点或区域的起始点和结束点；

（7） 增加谓词函数 unique()，用于测试 XPointer 表达式是否只返回一个位置，而不是多个位置或没有返回位置。

XPointer 转义

由于 XPointer 不是单独出现，而一些字符不能直接出现在包含 XPointer 的上下文中，因此这些字符必须转义出现。首先，XML 文件中不能直接出现的"<"、"&"、"'"、"'"必须转义出现，转义方法与 XML 规范相同；其次，XPointer 出现在 URI 中时，不能直接出现在 URI 中的字符必须转义出现，主要包括非 ASCII 字符，转义为"%HH"形式，HH 是该字符的十六位编码；另外，XPointer 语言要求不匹配的括号必须转义，转义方法是在不匹配的括号前面加上"^"，如果表达式中出现"^"，则在前再加一个"^"。

12 应用程序接口 DOM & SAX

作为一个程序开发人员，可能需要通过程序代码，访问一个 XML 文档中的内容。由于 XML 文档实际上就是一个文本文件，我们必须书写一个能够识别 XML 文档中信息的文本文件阅读器--XML 语法分析器，用来解释 XML 文档并提取其中的内容。显然，这不仅是一项非常耗时的工作，而且要求每个应用 XML 的人都要自己去处理 XML 中的语法细节。而且，在不同的应用程序或开发环境中，如果多种应用程序都需要访问 XML 文档中的数据，这样的分析器代码就要被重写多次。这时，你或许会产生这样一个疑问：把 XML 分析器做成一个 DLL 不就可以解决代码重写的问题了吗？回答当然是肯定的，但是，如果所有现成的 XML 分析器的接口都不相同，那么我们的开发就必须是针对某一个 XML 分析器的，当我们希望更换一个性能更高的分析器时，程序就必须被重写。因此，一个统一的 XML 数据接口也是必需的。也就是说，要真正实现代码的重用，就必须解决 DLL 的接口标准问题。

针对上述问题，W3C 以及 XML_DEV 邮件列表的成员分别提出了两个标准的应用程序接口：DOM 和 SAX。

DOM 和 SAX 在 XML 应用程序开发过程中的作用如图 1 所示：

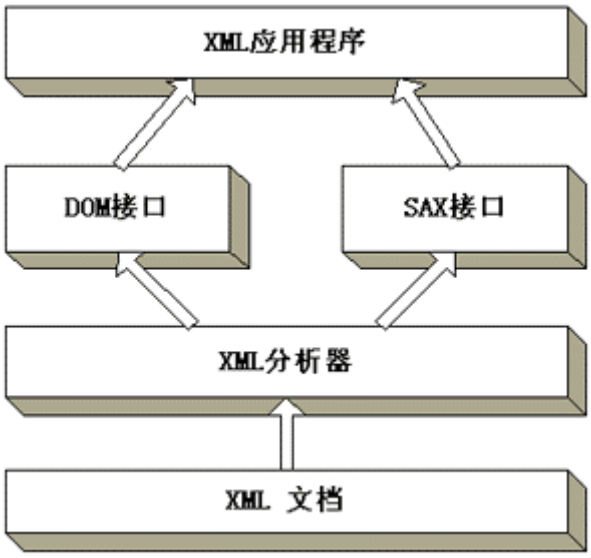


图 1 DOM 和 SAX 在应用程序开发过程中的作用

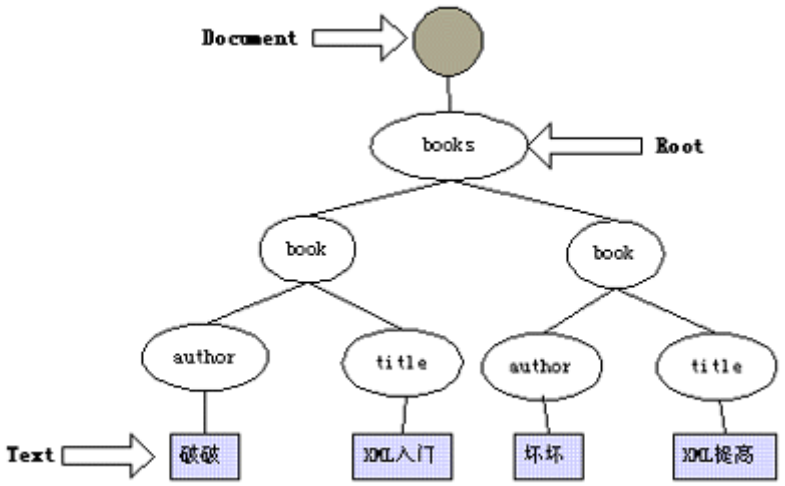
DOM 简介

DOM 的全称是 Document Object Model，也即文档对象模型。它是由 W3C 制定的一套标准接口规范。在应用程序中，基于 DOM 的 XML 分析器将一个 XML 文档转换成了一个对象模型的集合（这个集合通常被称为 DOM 树），应用程序可以通过对该 DOM 树的操作，实现对 XML 文档中数据的操作。通过 DOM 接口，应用程序可以在任何时候访问 XML 文档中的任何一部分数据，因此，这种利用 DOM 接口的机制也被称作随机访问机制。

XML 的一个显著特征就是它是结构化的。在结构化文档中，信息是按层次化的树形结构组织的。所以结构化文档模型的组织也必然是树形的。一个 DOM 接口的 XML 分析器，在对 XML 文档进行分析之后，不管这个文档有多简单还是有多复杂，文档中的信息都会被转化成一棵对象节点树。在这棵节点树中，有一个根节点--Document 节点，所有其他的节点都是根节点的后代节点。DOM 节点树生成之后，就可以通过 DOM 接口访问、修改、添加、删除、创建树中的节点和内容。例如，对于下面的 XML 文档：

```
<?xml version="1.0" encoding="gb2312" ?>
<books>
<book>
<author>破破</author>
<title>XML 入门</title >
</book>
<book>
<author>坏坏</author>
<title>XML 提高</title>
</book>
</books>
```

用 DOM 分析器分析之后，就会得到如图 2 所示的树结构。



在这棵文档对象树中，文档中所有的内容都是用节点来表示的。一个节点又可以包含其他节点，节点本身还可能包含一些信息，例如节点的名字、节点值、节点类型等。

由图 2 可以看出，在 DOM 中，文档的逻辑结构类似一棵树。文档、文档中的根、元素、元素内容、属性、属性值等都是以对象模型的形式表示的。文档中的根实际上也是一个元素，之所以要把它单独列出来，是因为在 XML 文档中，所有其他元素都是根元素的后代元素，而且根元素是唯一的，具有其他元素所不具有的某些特征。

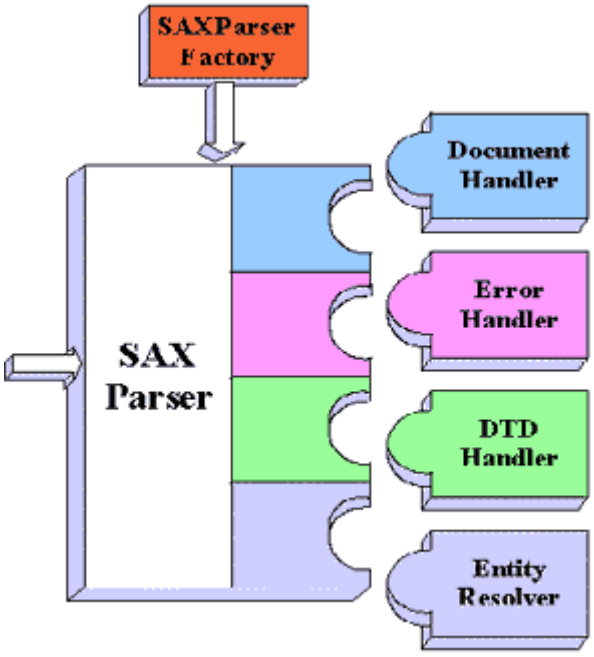
图 2 中给出的例子比较简单，事实上，DOM 中还包含注释、处理指令、文档类型、实体、实体引用、命名空间、事件、样式单等多种对象模型。

SAX 简介

SAX 的全称是 Simple APIs for XML，也即 XML 简单应用程序接口。它是 XML_DEV 邮件列表中的成员根据应用的需求自发地定义的一套对 XML 文档进行操作的接口规范。SAX 提供了一种对 XML 文档进行顺序访问的模式，这是一种快速读写 XML 数据的方式。当使用 SAX 分析器对 XML 文档进行分析时，会触发一系列事件，并激活相应的事件处理函数，从而完成对 XML 文档的访问，所以 SAX 接口也被称作事件驱动接口。

SAX 接口之所以叫做"简单"应用程序接口，是因为这个接口确实非常简单，绝大多数事情分析器都没有做，需要应用程序自己去实现。它的基本原理是由接口的使用者提供符合定义的处理器，XML 分析时遇到特定的事件，就去调用处理器中特定事件的处理函数。一般 SAX 接口都是用 JAVA 实现的，但事实上 C++也可以用于实现 SAX 接口，只是 C++的分析器比较少。

SAX 分析器的大体构成框架如图 3 所示：



对于下面的 XML 文档：

```
<?xml version = "1.0" encoding="gb2312"?>
<books>
<book>
<title>XML 入门</title>
<author>小喆喆</author>
<publisher>北京大学出版社</publisher>
<email>xml@icst.pku.edu.cn</email>
</book>
</books>
```

当利用 SAX 分析器中的 DocumentHandler 接口中的方法对文档进行分析时，方法的调用时序如图 4 所示：

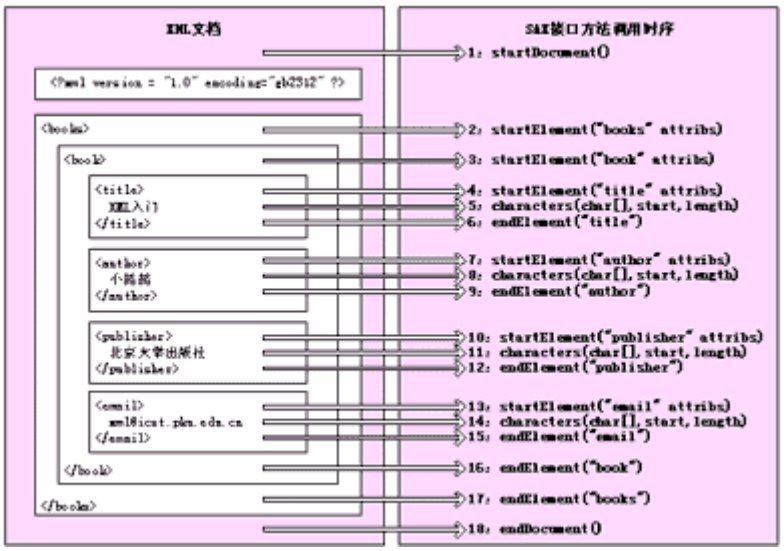


图 4 SAX 接口方法调用时序图

总结

DOM 分析器通过对 XML 文档的分析，把整个 XML 文档以一棵 DOM 树的形式存放在内存中，应用程序可以随时对 DOM 树中的任何一个部分进行访问与操作，也就是说，通过 DOM 树，应用程序可以对 XML 文档进行随机访问。这种访问方式给应用程序的开发带来了很大的灵活性，它可以任意地控制整个 XML 文档中的内容。然而，由于 DOM 分析器把整个 XML 文档转化成 DOM 树放在了内存中，因此，当 XML 文档比较大或者文档结构比较复杂时，对内存的需求就比较高。而且，对于结构复杂的树的遍历也是一项比较耗时的操作。所以，DOM 分析器对机器性能的要求比较高，实现效率不十分理想。不过，由于 DOM 分析器的树结构的思想与 XML 文档的结构相吻合，而且，通过 DOM 树机制很容易实现随机访问，因此，DOM 分析器还是有很广泛的使用价值的。

SAX 分析器在对 XML 文档进行分析时，触发一系列的事件，应用程序通过事件处理函数实现对 XML 文档的访问。由于事件触发本身是有时序性的，因此，SAX 分析器提供的是一种对 XML 文档的顺序访问机制，对于已经分析过的部分，不能再倒回去重新处理。SAX 之所以被叫做"简单"应用程序接口，是因为 SAX 分析器只做了一些简单的工作，大部分工作还要由应用程序自己去做。也就是说，SAX 分析器在实现时，它只是顺序地检查 XML 文档中的字节流，判断当前字节是 XML 语法中的哪一部分，检查是否符合 XML 语法并触发相应的事件。对于事件处理函数本身，要由应用程序自己来实现。同 DOM 分析器相比，SAX 分析器对 XML 文档的处理缺乏一定的灵活性，然而，对于那些只需要访问 XML 文档中的数据而不对文档进行更改的应用程序来说，SAX 分析器的效率则更高。由于 SAX 分析器实现简单，对内存要求比较低，因此实现效率比较高，同样具有广泛的使用价值。

13　数据库与信息交换

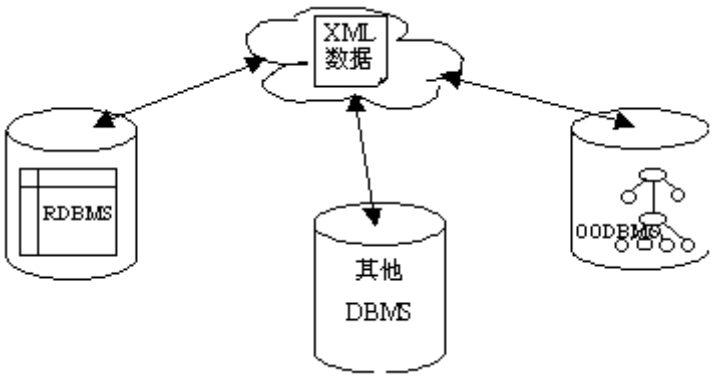
XML 信息交换的类型

从整体上讲，XML 定义了应用间传递数据的结构，而且这种结构的描述不是基于二进制的、只能由程序去判读的代码，而是一种简单的、能够用通用编辑器读取的文本。利用这种

机制，程序员可以制订底层数据交换的规范，而各模块之间传输的数据将是规范的符合既定规则的数据。从应用的角度来看，XML 信息交换大致可分为下面几种类型：数据发布、数据集成和交易自动化。通过 XML 可以实现跨媒体、多介质的数据发布。传统的信息发布方式多是基于纸介质和 CD-ROM 的信息发布，而 XML 的出现，使得跨媒体数据发布技术又向前发展了一步。2000 年 5 月 18 日，一个由数字印刷领域的知名厂家组成的所谓"按需印刷"组织（PODi）发布了"个性化印刷置标语言"（Personalized Print Markup Language, PPML）规范。这是一种基于 XML 的技术规范，主要用于带有可再利用内容文档的快速印刷。如果说数据发布涉及到的是服务器-浏览器形式的数据交换，那么，数据集成则是一种服务器-服务器之间的数据交换，这对于 B2B 电子商务系统尤其重要。另外，XML 也有助于提高应用的自动化程度。遵循共同的标准，使得应用程序开发商开发出具有一定自动处理能力的代理程序，从而提高工作效率。

XML 数据存取机制

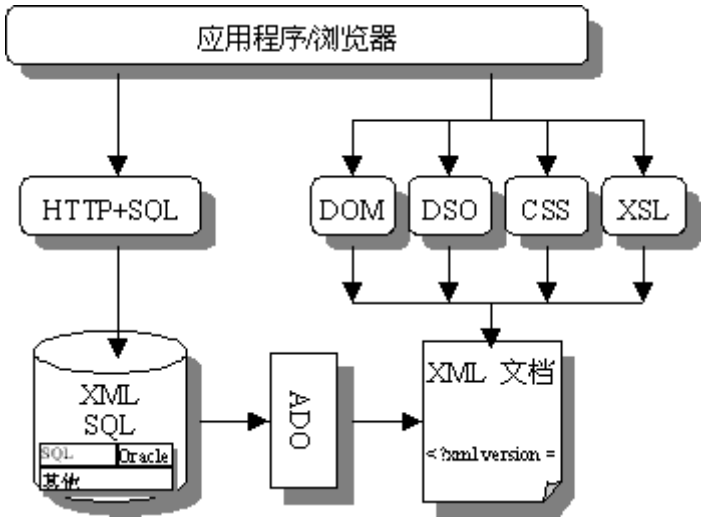
XML 数据源多种多样，根据具体的应用，大概可分为下面三种：一种是 XML 纯文本文档，第二种是关系型数据库，第三种则来源于其它各种应用数据，如邮件、目录清单、商务报告等。其中，第一种来源是最基本的也是最为简单的，将数据存储于文件中的优点在于可以直接读取，或者加以样式信息在浏览器中显示，或者通过 DOM 接口编程同其它应用相连。第二种数据来源是对第一种来源的扩展，其目的是便于开发各种动态应用，其优点则在于通过数据库系统对数据进行管理，然后再利用服务器端应用（如 ASP、JSP、Servlet）等进行动态存取。这种方式最适合于当前最为流行的基于三层结构的应用开发。第三种数据由于来源广泛，因此需要具体情况具体对待。下图示出了典型的 XML 数据存取机制。



XML 在数据库中的应用模式

通常，XML 在数据库中的应用模型需要借助三层架构来实现。这种模式下，在用户接口层，利用 CSS 或 XSL 技术，XML 可以实现基于 Web 浏览器的多样式可视化显示。而中间层则需要有一个代理程序运行于其中，通过它来访问数据库管理系统中的数据和输出 XML 文档。另外，这种代理程序还可以进行双向的基于事件的数据更新，也就是说，客户端的数据变化（如数据的插入、删除、修改等）可以通过代理程序反映到底层数据库，而数据库的更新也能够通知到客户端。表面上看，这种机制同传统的三层架构没有什么区别，但实际上是不同的，因为此时在传输过程中的数据都是已经 XML 化了的。 微软在其 Windows 分布式 Internet 应用（即 Windows DNA）架构中集成了 XML 技术。通过中间层的代理程序，可获取的数据来源可以不必局限于某台固定的数据库服务器，而可以是分布于企业内，甚至于遍及全球各地的数据库服务器。另外，借助于 XML Schema，开发者就能更为精确地描述和交换数据，因而大大地提高这种应用的效率。

XML 提供了一种连接关系数据库和面向对象数据库以及其它数据库管理系统之间的纽带（参见下图）。XML 文档本身是一种由若干节点组成的结构，这种特点使得数据更适宜于用面向对象格式来存储，同时也有利于面向对象语言（C++、Java 等）调用 XML 编程接口访问 XML 节点。关系数据库和面向对象数据库首先需要将数据从数据库中提取出来，经过转换或直接以 XML 数据形式发布到网上（局域网或 Internet 网），然后相互交换数据，经应用层系统处理后再转存入库。



XML 数据交换技术及应用

到目前为止，已有大量关于 XML 数据交换技术和应用面世。其中，有的只是将现有技术扩展 XML 支持，有的属于 XML 中间件产品，还有的是比较完整的 XML 应用。它们大多数都提供了对数据库的支持，这不能不从一个侧面反映出 XML 与数据库的密切关系以及基于 XML 数据库应用的潜力。不同的编程语言和脚本语言需要不同的 SQL API 和 XML 语法分析器组合。例如，对于一个 C++程序员来说，编写一套访问数据库的 XML 应用程序可能需要利用 ODBC 和 C++ XML 语法分析器；而对于一个 Java 程序员来说，可能只需要 JDBC 和 Java XML 语法分析器就够了；更为特殊地，如果你对微软的 Visual Basic 和 VBScript 脚本语言比较熟，那么很有可能你会用它们来开发 XML 应用，此时，你只要再学习一下 ADO，然后借助微软的 XML 语法分析器进行编程。除此之外，DB2XML、InterAccess、ODBC2XML、XML Servlet、XOSL、ASP2XML 都提供了基于 XML 的数据存取机制，它们有的是以组件的形式提供，有的属于转换工具，还有的则是完整的软件包。下面是一个 ASP 示例，通过在调用 ADO 和 DOM 接口从数据库中提取数据以动态生成 XML 文档。

```
<% @language = "VBScript" %>
<% Response.ContentType = "text/xml" %>
<?xml version="1.0" encoding="GB2312" ?>
<%

' 动态构建 XML 文档
set xmlDoc = Server.CreateObject(Microsoft.XMLDOM)
set root = xmlDoc.createNode("element","通讯录","")
xmlDoc.appendChild(root)
```

```
' 查询数据库
sqlStr = "select * from roster"
set cConn = Server.CreateObject("ADODB.Connection")
cConn.Open "ROSTER","sa", ""
set rsData = cConn.Execute(sqlStr)

rsData.MoveFirst()
while (not rsData.EOF )
'构建联系人子节点
set tmpNode = xmlDoc.createNode("element","联系人","")
xmlDoc.documentElement.appendChild(tmpNode)

'构建姓名、年龄、电话、地址子节点
for i = 0 to rsData.Fields.Count - 1
set childNode =
xmlDoc.createNode("element",rsData.Fields(i).Name,"")
childNode.text = rsData.Fields(i)
tmpNode.appendChild(childNode)
next

rsData.MoveNext()
wend

Response.Write(xmlDoc.xml)
%>
```

编程实现基于 XML 的数据存贮当然是一种手段，但略显繁琐，因此一些支持 XML 的数据库产品又提供了一种更为方便的途径，即通过 URL 的方式直接进行 XML 的数据存贮。比较典型的示例如下：

- http://localhost/northwind?sql=SELECT+FirstName,LastName+FROM+Employees+FOR+XML+AUTO&xsl=employee.xsl&contenttype=text/html&root=root (注：Microsoft SQL Server 2000 技术实现)

http://localhost/tamino/xmldb?_XQL=patient/name (注：Software AG Tamino Server 技术实现)

14 XML 相关标准

从本系列讲座或其他资料中会发现，虽然 XML 标准本身简单，但与 XML 相关的标准却种类繁多，W3C 制定的相关标准就有二十多个，采用 XML 制定的重要的电子商务标准就有十多个。这一方面说明 XML 确实是一种非常实用的结构化语言，并且已经得到广泛应用；另一方面，这又为学习了解这些标准带来一定得困难，除了标准种类繁多外，标准之间通常还互相引用，特别是应用标准，它们的制定不仅仅使用的是 XML 标准本身，还常常用到了

其他很多标准。对某一特定的应用领域哪些标准是重要的呢？哪些标准是基础的，被其他标准广泛引用呢？这些标准之间的相互关系如何呢？XML 标准的体系是怎样的呢？

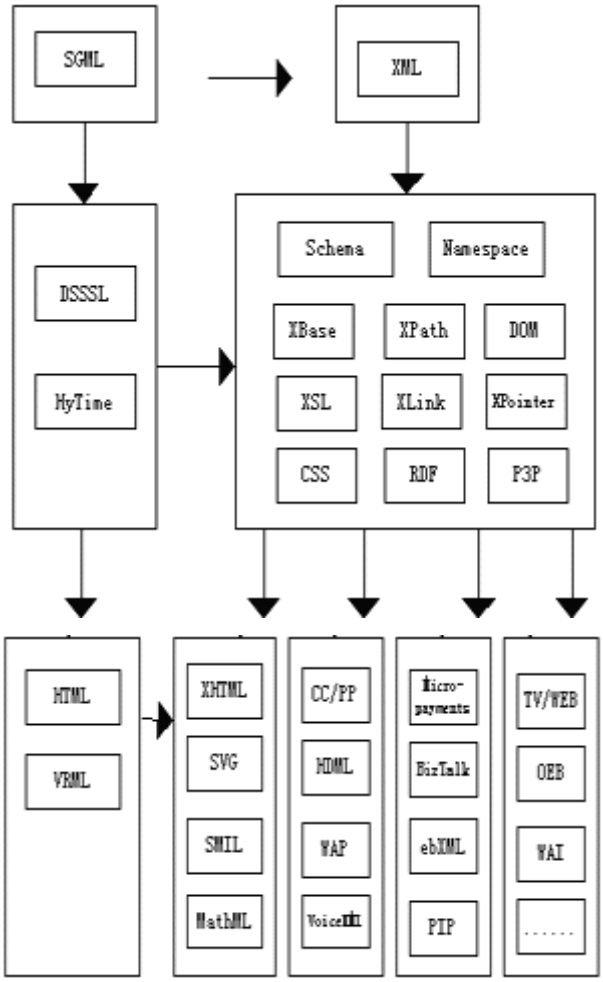
1 XML 标准体系

XML 相关标准也可分为元语言标准、基础标准、应用标准三个层次，如图 11-1 所示。

元语言标准(meta-Language)：描述的是用来描述标准的元语言。在 XML 标准体系中就是 XML 标准，是整个体系的核心，其他 XML 相关标准都是用它制定的或为其服务的。

基础标准（Foundation Standards）：这一层次的标准是为 XML 的进一步实用化制定的标准，规定了采用 XML 制定标准时的一些公用特征、方法或规则。如：XML Schema 描述了更加严格地定义 XML 文档的方法，以便可以更自动化的处理 XML 文档；XML Namespace 用于保证 XML DTD 中名字的一致性，以便不同的 DTD 中的名字在需要时可以合并到一个文档中；等等。这些基础标准主要有哪些？它们之间的相互关系又如何？在本讲座的后半部分将作更为详细的讨论。

应用标准（Application Standards）：XML 已开始被广泛接受，大量的应用标准，特别是针对 INTERNET 的应用标准，纷纷采用 XML 进行制定。有人甚至认为，XML 标准是 Internet 时代的 ASCII 标准。在这 Internet 时代，几乎所有的行业领域都与 Internet 有关。而这些行业一旦与 Internet 发生关系，都必然要有其行业标准。而这些标准往往采用 XML 来制定。当前较为重要的应用标准主要包括：用于 XML 显示的标准：XHTML（采用 XML 对 HTML 的重新定义）、SVG（有关矢量图形的）、SMIL（有关多媒体同步显示的）、MathML（有关数学公式符号的）；用于电子商务领域的标准：Micropayments（W3C 制定的）、BizTalk（Microsoft 发起的电子商务的 schema 库）、ebXML（联合国 UN/CEFACT 小组和 OASIS 共同发起的）、PIP（由诸多 IT 业的巨子组成的一个标准化组织 RosettaNet 的应用网络标准。），cXML、xCBL、tpaML 等等；以及其他领域的。



2 XML 基础标准及其相互关系

从 XML 标准体系中可以看到 XML 基础标准是相当多的，而且这些标准又是非常重要的，因为，这些标准是 XML 应用标准的基础。它们是在 XML 标准的基础上，进一步对 XML 中一些公共的特性、方法、规则作了更为详细明确的规定。应用标准通常都要使用到这些标准的内容或者遵照其中的约定。在使用 XML，阅读利用 XML 应用标准时也需要理解这些标准的内容。在本系列讲座中，这些标准几乎都已涉及到了。在本次讲座中，将对它们进行分类，主要讨论这些标准之间的相互关系。以便读者能从整体上对这些标准有个更清晰的认识。

图 11-2 是对 XML 应用标准的框架图。

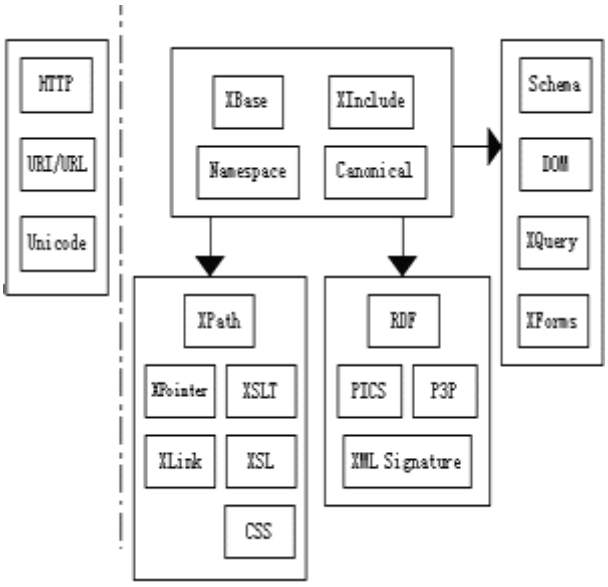


图 11-2

首先，看看 XML 相关标准外围的一些标准，那些对 WEB 应用具有确定体系框架意义的几个标准，如图 11-2 中虚线左侧的三个标准。这些标准并不是针对 XML 标准应用或采用 XML 标准制定的，但它们是 WEB 应用的基础，几乎在 WEB 应用的任何地方都会使用到它们。HTTP：超文本传输协议。URI/URL：统一资源标识符/统一资源定位器，是用来定位 Internet 上资源，以便在庞大的 WEB 信息系统中能唯一地标识任何一个资源。Unicode 是在 WEB 应用中广泛采用的一个字符编码标准，它将几乎世界上所有的文字都包括进去了。XML 标准要求 XML 分析器必须至少支持 UTF-8/16 编码的 Unicode 字符。

如图 11-2 中上部所示的标准是仅次于 XML 标准本身，居于核心地位，并且几乎被其他所有 XML 相关标准采用的一组标准。这些标准是由 XML 核心工作组（XML Core Working Group）制定的，为 XML 标准提供最为本质的支持。XML Base，用于定义 XML 文档的 URI 的基础部分标准，与 HTML BASE 相似。XML Inclusions (XInclude)，用于规定文档中包含物的处理模型与语法规则。Namespaces in XML，提供了一种简单的方式，用来在 XML 文档中通过与由 URI 引用标识的命名空间相关联来限定元素和属性，提供了解决多 DTD 的 XML 文档中元素名、属性名冲突的基本方法。Canonical 描述了一种对输入的 XML 文档生成范式的方法。

图 11-2 中的右侧四个标准 XML 文档的处理提供有效的方法与规则。Schema 是对 DTD 的补充，提供了一种更为严格的描述 XML 文档的结构、属性、数据类型等的方法，以便可对 XML 文档进行更加严格的自动化处理。DOM 定义了一组与平台和语言无关的接口，以便程序和脚本能够动态访问和修改 XML 文档内容、结构及样式。XQuery 的目的是为从 WEB 中的实际的或虚拟的文档中提取数据，提供一种灵活的查询机制。XForms 的关键思想是将用户界面和表现与数据模型和逻辑分开，以便同一个表单可被广泛地应用于手持设备、桌面设备或基于语音的浏览器等各种情况。

大家可能对图 11-2 中的 XSL、XLink 一组标准很熟悉。因为这组标准在 HTML 标准中已有其雏形：显示与链接。而且这是 HTML 中最为重要与常用的内容。同时，这组标准的内容充分继承了 SGML 标准中 DSSSL 与 HyTime 相关内容。在这组标准中 XPath 描述如何识别、选择、匹配 XML 文件中的各个构成元件，包括元素、属性、文字内容等。XPointer 和 XLink 标准，继承了 HyTime 标准中有关定位、链接方面的内容，链接采用单独的元素形式，并在

标准中定义了"元元素"，以便作为模板或父元素类型，链接可以有多种形式等。CSS 已经是很熟悉的内容了，它开始制定时是用来进行 HTML 文档的显示，随着 XML 的出现与发展，它也被用来作为 XML 文档显示的样式标准，并将继续使用下去。但是，XML 出现的目的并不在于它的显示，而主要在于对数据内容本身的描述，其中数据转换是其重要内容之一。为此，很快出现了制定 XML 转换标准的需求，XSL 孕育而生。XSL 的制定借鉴了 DSSSL 和 CSS 的内容与经验，如 XSL 标准采用对 XML 文档形成树状结构，采用元素节点匹配的方式进行转换、采用格式化对象的方法进行显示格式的定义。

在图 11-2 中，还有一部分可能读者接触较少。RDF、PICS、P3P、XML-Signature。RDF (Resource Description Format)是采用 XML 语法格式处理元数据的应用，为描述图像、文档和它们之间的相互关系定义的一个简单数据模型。其他几个标准一般的 Internet 使用者很少直接使用这些标准。但它们是采用 XML 定义的几个 Internet 应用的基础标准。

15　XML 应用前景及实用工具

作为互联网的新技术，XML 的应用非常广泛，可以说 XML 已经渗透到了互联网的各个角落。本讲主要对现有的 XML 应用作一个简要介绍。

XML 应用分类

虽然人们对 XML 的某些技术标准尚有争议（也许这就是许多标准迟迟不能推出的原因），但是人们已经普遍认识到 XML 的作用和巨大潜力，并将 XML 应用到互联网的各个方面。考察现在的 XML 应用，可以大致将它们分为以下几类：

- 设计置标语言
- 数据交换
    - 替代传统的 EDI
    - 智能代理和精确搜索
- Web 应用
    - 集成不同数据源
    - 本地计算
    - 数据的多种显示和网络出版
    - 支持 Web 应用的互操作和集成
- 文件保值

身边的 XML

XML 自从出现以来，它已经逐渐来到我们身边，只是由于 XML 作为底层的实现技术，不一定能引起人们的注意。

在 Netscape Navigator 4.06 以后的版本中，其中有一项功能称为"What's Related"，用来指示与当前浏览的网页相关的站点或网页。当用户选择该菜单时，从服务器传输回的数据格式就是资源描述框架 RDF（Resource Description Framework）格式。RDF 是用于处理元数据的 XML 应用，所谓元数据，就是"描述数据的数据"或者"描述信息的信息"。RDF 规范并

没有定义描述资源所用的词汇表，而是定义了一些规则，这些规则是各领域和应用定义用于描述资源的词汇表时必须遵循的。当然，RDF 也提供了描述资源时具有基础性的词汇表。

在 Internet Explorer 4.0 及其以后的版本中，Microsoft 推出了"频道"（Channel）的概念，用户可以通过订阅频道实现站点更新内容的自动获取，频道定义使用的格式 CDF（Channel Definition Fomat），也是一个 XML 应用。CDF 文件是一个定义了读者和站点内容的连接参数的 XML 文件，它与站点上的 HTML 文件分开，但链接到某个或某几个 HTML 文件。Internet Explorer 是唯一支持 CDF 的浏览器。

电子商务

电子商务就是利用电子手段尤其是互联网进行商务活动。从技术上说，电子商务是通过互联网传输和交换商务数据，并能根据商务数据进行人工或自动处理。XML 的可扩展性和自相容性等特点，使它成为数据交换的有力工具。

电子商务首先出现的类型是企业-消费者，该类型电子商务的信息是直接呈现在浏览器中，供人们阅读，因此侧重表现的 HTML 在其中起到了巨大作用。随着比较购物和个性化要求以及企业-企业类型电子商务的出现， 人们要求计算机能够理解数据的语义，而且能够将数据和表现的分离开来，这时 HTML 就显得力不从心。XML 弥补了 HTML 的巨大缺陷，成为电子商务中的核心技术。

随着 XML 标准体系的成熟和技术的发展，已经出现了相当多的客户化工具，尤其是可视化工具的出现，使得人们可以无须了解 XML 的细节就能够编写出需要的 XML 文档，使得 XML 应用在电子商务中成为可能。而浏览器对 XML 越来越强的支持能力，对 XML 应用起到了巨大的促进作用。

当前已经出现了很多基于 XML 的针对企业-企业电子商务的标准或旨在形成相应标准的计划，包括 Microsoft 的 BizTalk、UN/CEFACT 小组和 OASIS 共同发起的 ebXML 计划、CommerceNet 发起的 eCo 计划、RosettaNet 的 PIP（Partner Interface Process）和 RosettaNet 应用网络标准、XML-EDI、CommerceOne 的 xCBL 标准、Ariba 的 cXML 等。

网络出版

随着互联网的飞速发展，互联网已经成为继报刊、电台、电视台之后的一种新型媒体。在 1998 年 5 月举行的联合国新闻委员会年会上，互联网这一新型媒体被正式冠以"第四媒体"的称号。网络出版自从出现以来，用于信息发布的主要是 HTML 技术，但是这种方式在跨媒体出版时遇到了极大的困难，人们需要为不同媒体制作不同版本。XML 的内容与显示分离的特点，人们可以一次性制作内容，配以不同的样式单，实现一次制作多次出版。

为了满足不同领域和显示设备的需要，人们利用 XML 定义了多个面向显示的语言，包括 XHTML（eXtensible Hyper Text Markup Language，用 XML 重新定义的 HTML）、面向 WEB 图形的 VML（Vector Markup Language）、PGML（Precision Graphics Markup Language）和 SVG（Scalable Vector Graphic）、面向多媒体的 SMIL（Synchronized Multimedia Integration Language）、面向电子书和电子报纸的 OEB（Open eBook Structure Specification）、面向手持设备的 WML（Wireless Markup Language）和 HDML（Handheld Device Markup Language）等。可以说 XML 已经成为网络出版的重要工具，并将发挥日益重要的作用。

移动通信

为了满足人们随时随地与互联网连接的需要，Phone.com 联合了 Nokia、Ericsson、Motorola 在 1997 年 6 月建立了 WAP 论坛，旨在利用已有的互联网技术和标准，为移动设备连接互联网建立全球性的统一规范。在 1998 年 5 月，推出了 WAP 规范 1.0 版。并于 1999 年 11 月发布最新的 1.2 版。WAP 规范包括 WAP 编程模型、无线置标语言 WML、微浏览器规范、轻量级协议栈、无线电话应用（WTA）框架、WAP 网关几个组件。其中 WML 是利用 XML 定义的专为手持设备的置标语言。另外 W3C 也定义了一个基于 XML 的手持设备置标语言 HDML，WML 和 HDML 非常类似，因为 WML 脱胎于 HDML，可以说根在 HDML，而花开 WML。需要指出的是，虽然人们在提到 WAP 时首先想到的是手机上网，但掌上电脑等手持设备的上网也可以使用 WAP。

XML 前景展望

XML 自从出现以来，一直受到业界的广泛关注。自从 1998 年 2 月成为推荐标准后，许多厂商加强了对它的支持力度，包括 Microsoft、IBM、ORACLE、SUN 等，它们都推出了支持 XML 的产品或改造原有的产品支持 XML。W3C 也一直在致力于完善 XML 的标准体系。然而由于 XML 的复杂性和灵活性，加上工具的相对缺乏，增加了 XML 使用的难度。因此，XML 很难在短期内完全替代 HTML，成为互联网的主角。另外，由于 XML 是元置标语言，任何个人、公司和组织都可以利用它定义新的标准，这些标准间的通信成为了巨大的问题，因此人们在各个领域形成一些标准化组织以统一这些标准，但是这些努力并不一定能够形成理想的结果。无论如何，XML 的出现为互联网的发展提供了新的动力，终将成为互联网上全新的开发平台。它促使了新的类型的软件和硬件的形成和发展，而这些发展又将反过来促进 XML 的发展。

实用工具概览

伴随着 XML 技术本身的成熟和应用领域的不断扩大，相关的工具也如雨后春笋不断地被开发出来。下面将介绍现有的、较有代表性的 XML 的实用工具。

一． XML 浏览工具

1． Microsoft Internet Explorer IE 是 Micorsoft 公司开发的 Web 浏览器，是当今两大主流浏览器之一，最先开创了 XML+CSS、XML+XSL 的 Web 浏览方式。但 IE5 对 CSS 的支持并不完全，尚不完全支持 CSS1，对 CSS2 只提供部分支持。IE5 对 XSLT 有所支持，能实时地将一个 XML 文档根据 XSL 样式单转换为 HTML 文档来显示，但支持的是早期草案。

新发布的 IE5.5，增强了对 DHTML 和 CSS 的支持，但仍然有待改进。对 XSLT 的支持仍然基于早期的草案。

参考网页：http://www.microsoft.com/windows/ie/download/ie55.htm。

2．Mozilla 是两大主流浏览器之一，是在 Netscape Communicator 5.0 的一个较早版本的源代码的基础上开发而成的，因此，人们总是把它看做是 Netscape5.0 的未来版本。Mozilla 的新增功能中有很大的比重在 XML 方面，表现在其对 XML 的支持、对 MathML 的支持。在对 CSS 的支持方面，也以较完善的支持超过了 IE5，但它不支持 XSL。

此外， Netscape6.0 基于 Mozilla 引擎，支持 HTML 4.0、XML、CSS、DOM、命名空间，简单 XML 链接。

参考网页：http://www.mozilla.org。

http://www.netscape.com。

二． XML 编辑工具

1．XML Spy

Icon Information-Systems 公司的产品，提供集成开发环境 IDE，但不支持所见即所得。支持 Unicode、多字符集，支持格式良好的和有效的 XML 文档。可编辑 XML 文档、DTD、schema，以及 XSLT。

XML Spy 提供了四种视窗：结构视窗以树形结构编辑 XML 文档（包括 XML、XSL、DTD 文档）；增强表格视窗以表格的方式显示出文档中元素的数据库项；源代码视窗可以查看和修改文档源码；预览视窗采用内嵌 IE5 的方式在软件内对 XML 文档进行浏览，支持 CSS 和 XSL。

XML Spy 可运行在 Windows95/98/2000 和 WindowsNT 环境。

参考网页：http://new.xmlspy.com。

2．VisualXML

由 Pierre Morel 开发，以树形结构显示 XML、DTD、DOM 文档；实现同数据库的集成，并可通过 Wizard 方式进行数据库的浏览、SQL 语句和存储过程的创建和执行；以图形界面实现 XML 元素同数据库对象的绑定，同时创建 XML 文档和 DTD 文档；支持多种数据库，如 Oracle、Access、SQL Server、Informix、Sybase、DB2。

该软件的运行环境是 Java (JDK 1.1)。

参考网页：http://www.pierlou.com/visxml。

3．EditML

EditML Technologies 开发的产品，是 Windows 平台上的 XML 编辑器，可以用于创建有效的及格式良好的 XML 数据文档，Schema 文档或者 XSL 样式单。它使用 Microsoft 的 MSXML 分析器，遵守 W3C 的 XML1.0 标准。

参考网页：http://www.editml.com。

三． XSL 编辑工具

1． eXcelon Stylus

是 eXcelon 公司开发的第一个可视化 XSL 编辑软件。

eXcelon Stylus 是一个面向 XSL 编辑，集创建、管理、保存于一体的集成环境，使用它用户可以快速、简便地创建 XSL 样式单，并可以很方便地进行调试。使用 Stylue 可以进行 XML 商务开发。

参考网页：http://www.exceloncorp.com。

2．IBM XSLEditor

IBM 公司的 alphaworks 开发，是交互式 XSL 编辑软件，支持 XSLT 与 XPath。有四个主要的窗口完成 XML 编辑、XSL 编辑、结果树显示和 XPath 匹配。

该软件要在 JAVA 环境中运行，要求安装有 JDK1.1 版本。
参考网页：http://www.alphaworks.ibm.com。

四． XML 分析工具

1． IBM XML4J

完全用 JAVA 开发，是目前功能比较全面的支持有效性检查的 XML 分析器。它遵守 XML1.0 标准、最新的 DOM、SAX、命名空间的标准，提供多语种支持，支持 XML 的有效性检查，支持元素识别、DOM 创建、错误处理等。

除此以外，IBM 还用 C++编写了 XML 分析器，称为 XML4C。
参考网页：http://www.alphaworks.ibm.com。

2．Microsoft 的 MSXML

微软 XML 分析器已经内嵌入 IE4 和 IE5，支持一般的语法检查，也提供有效性检查供选择，它利用 JAVA 将一个 XML 文档中的数据组织为树型结构。
参考网页：http://www.microsoft.com/workshop/xml/parser/jparser.asp。

3．expat

expat 是 James Clark 实验室的开发项目，用 C 写成，目前正尝试用于 Netscape Nevigator 5 和 Perl 中，以便在 Netscape 的下一个版本中对 XML 提供支持。
参考网页：http://www.jclark.com/xml/expat.html。

五． 图形应用工具

1．IBM SVGViewer

IBM 公司 alphaworks 开发，该软件的主要功能有：基本图形、路径（Path）、图形勾边（Stroke）、图形填充（fill）、文字字体字号、字的轮廓填充、图象、Path 裁剪、链接、对象成组。

该软件在运行时要安装 JAVA2.0 以上的运行环境和 IBM 的 XML4J XML 分析器。
参考网页：http://www.alphaworks.ibm.com/tech/svgview。

2．CSIRO SVG Viewer

由 CSIRO 开发，对图形的支持较好，对 SVG 的支持功能有基本图形、路径（Path）、图形勾边（Stroke）、图形填充（fill）、文字的字体字号、图象、Path 裁剪、链接、对象成组、嵌入 Javascript。
参考网页：http://sis.cmis.csiro.au/svg/。

3．支持 SVG 的其他工具

- Corel 公司的 CorelDraw9.0 配备了 SVG 的过滤器，可生成含有 SVG 的网页。
- Adobe Illustrator9.0 也支持 SVG。
- Adobe 公司开发出两大主流浏览器 IE 和 Netscape 的 SVG 插件。

六． WAP 应用工具

1． UP.SDK for WML 由 Phone.com 提供的，支持 WML1.1。UP.SDK 包含有文档说明、工具和编码样例，帮助开发者在 Phone.com 的 UP.Link 平台上开发 WAP 应用程序。

在该开发包中还有一个 UP 模拟器，可以模拟手持设备上网。

该开发包的运行平台是 Windows 95/98 及 Windows NT 4.0。
参考网页：http://www.phone.com。

2．Nokia WAP 开发包

由 Nokia 提供的 WML 开发包，由两部分组成， WML 编辑器和 Nokia 手机的模拟器，可以边调试边显示。

软件的运行环境是 JAVA，对中文有所支持。
参考网页：http://www.nokia.com。

3．Wappage

由 Wapmine.com 公司开发，是一个交互式的所见即所得 WML 编辑软件。支持项目管理，并有多个视窗方便用户编辑。用户不需要知道详细的 WML 标记就可以进行操作，也允许直接修改 WML 代码。对于 Card 的管理也用树形结构表示。

该软件的运行平台是 Windows95/98/NT。
参考网页：http://www.wapmine.com。

七． 电子出版工具

1． Microsoft Reader

Microsoft 开发出的一种专门的 eBook 阅读软件 Microsoft Reader，支持 OEB1.0。

Reader 的最大特色就是首次使用了 Microsoft 对于字符显示的研究新成果 ClearType，它是一种能使文字在显示器上的显示效果象印在纸上一样平滑清晰的字符技术。
参考网页：http://www.microsoft.com/reader。

2．ReaderWorks standard

ReaderWorks.com 推出的支持 Microsoft Reader 的软件，它可以生成 Reader 可读的文件格式，可将 HTML、TXT 等格式文件转换为 Reader 格式，让用户生成自己的 eBook。
参考网页：http://www.readerworks.com/English/standard.html。