高等学校计算机及其应用系列

XML技术及应用

主　编/刘晓义
副主编/刘亚辉　赵　妍　朱自民

重点规划教材

哈尔滨工程大学出版社
Harbin Engineering University Press

高等学校计算机及其应用系列

XML技术及应用

主　编/刘晓义
副主编/刘亚辉　赵　妍　朱自民

重点规划教材

HEUP 哈尔滨工程大学出版社
Harbin Engineering University Press

内容简介

XML是一个开放的,以文字为基础的卷标语言,它可以提供结构化的以及与语意有关的信息数据。这些关于数据的数据或中继数据提供附加的意义和目录给使用那些数据的应用程序,而且也将以网络为基础的信息管理和操作提升到一个新的水平。

本书共分为十六章,对XML的基本概念、语法规则和程序设计方法进行了详细的介绍。内容包括XPath教程、XSLT教程、XSL-FO教程、XQuery教程、XLink和XPointer教程及DTD教程等。章后附精选习题,可供上机练习或备考使用。

本书可作为大专院校XML教材,也可供自学使用。

图书在版编目(CIP)数据

XML技术及应用/刘晓义主编. —哈尔滨:哈尔滨工程大学出版社,2011.6
ISBN 978-7-81133-960-4

Ⅰ.①X… Ⅱ.①刘… Ⅲ.①可扩充语言,XML-程序设计 Ⅳ.①TP312

中国版本图书馆CIP数据核字(2011)第104453号

出版发行 哈尔滨工程大学出版社
社　　址 哈尔滨市南岗区东大直街124号
邮政编码 150001
发行电话 0451-82519328
传　　真 0451-82519699
经　　销 新华书店
印　　刷 黑龙江省教育厅印刷厂
开　　本 787mm×1 092mm 1/16
印　　张 13.5
字　　数 325千字
版　　次 2011年6月第1版
印　　次 2011年6月第1次印刷
定　　价 27.00元

http://press.hrbeu.edu.cn
E-mail:heupress@hrbeu.edu.cn

前言

当 XML(扩展标记语言)于 1998 年 2 月被引入软件行业时,它给整个行业带来了一场风暴。有史以来第一次,这个世界拥有了一种用来结构化文档和数据的通用且适应性强的格式,它不仅仅可以用于 WEB,而且可以被用于任何地方。

XML 是一个开放的,以文字为基础的卷标语言,它可以提供结构的以及与语意有关的信息给数据。这些关于数据的数据或中继数据(metadata)提供附加的意义和目录给使用那些数据的应用程序,而且也将以网络为基础的信息管理和操作提升到一个新的水平。

既然 XML 是以文字为基础的,因此可以很容易让非技术性人员理解,它所提供的组织、描述和结构化数据的能力,使它也可被技术性的应用程序所使用。因此,XML 所建立的结构化数据可同时供数据处理和显示之用。

目前关于 XML 的教材版本较多,因此特色和实用性显得尤为重要。本书在保证阐述问题清楚的前提下力求内容简练,在保留经典内容的同时摒弃了陈旧过时的知识。本书具有以下特点:

1. 学习本书无需有 XML 编程基础,可以从零学起;

2. 书中内容循序渐进、难点分散、各个击破,使读者不会被大量的新知识、新名词吓倒,从而渐入佳境;

3. 本书介绍的是 XML 基础知识,实际上 XML 内容十分丰富。在学习完本书后,编写更加复杂的 XML 程序时,应进一步学习 XML 的深入、提高部分,可参考其他相关书籍;

4. 本书介绍的 XML 知识和 XML 程序符合 SGML 标准,因此具有通用性。所涉及的程序可全部在任意浏览器中运行通过;

5. 本书所精选的习题具有典型性,同时涵盖了 XML 语言基础内容。因此,本书既可以作为大专院校的教材,也可以作为 XML 爱好者自学的参考资料。

书共分为 16 章,其中第 1 章、第 9 章、第 11 章、第 12 章由刘晓义编写,第 2 章、第 4 章、第 5 章、第 10 章、第 14 章及附录四、附录五由刘亚辉编写,第 3 章、第 6 章、第 13 章、第 15 章及附录一、附录二、附录六由赵妍编写,第 7 章、第 8 章、第 16 章及附录三由朱自民编写。

本书在编写过程中,参考或引用了国内一些专家的论著,在此表示感谢。

由于编者水平有限,书中错误在所难免,恳请读者批评指正,以便将来再版修改。

编　者

2011 年 4 月

目 录

第 1 章 XML 概述 …… 1
1.1 XML 简介 …… 1
1.2 XML 树结构 …… 3
1.3 XML 语法规则 …… 5
1.4 XML 元素 …… 7
1.5 XML 属性 …… 8
1.6 XML 验证 …… 11
1.7 一个简单 XML 案例 …… 12
1.8 XML 命名空间(XML Namespaces) …… 13
1.9 转义字符 …… 16
1.10 CDATA …… 16
1.11 XML 编码 …… 17
1.12 XML 编辑器 …… 18
第 2 章 XPath 教程 …… 19
2.1 XPath 简介 …… 19
2.2 XPath 节点 …… 19
2.3 XPath 语法 …… 21
2.4 XPath Axes(坐标轴) …… 23
2.5 XPath 运算符 …… 25
第 3 章 XSLT 教程 …… 27
3.1 XSLT 简介 …… 27
3.2 XSLT 转换 …… 27
3.3 XSLT 语法 …… 30
第 4 章 XSL-FO 教程 …… 43
4.1 XSL-FO 简介 …… 43
4.2 XSL-FO 区域 …… 44
4.3 XSL-FO 输出 …… 45
4.4 XSL-FO 实例 …… 45
4.5 XSL-FO 流 …… 46
4.6 XSL-FO 页 …… 47
4.7 XSL-FO 块 …… 49
4.8 XSL-FO 列表 …… 52
4.9 XSL-FO 表格 …… 53
第 5 章 XQuery 教程 …… 55
5.1 XQuery 简介 …… 55

5.2 XQuery 实例 …… 55
5.3 XQuery FLWOR …… 57
5.4 XQuery FLWOR + HTML …… 57
5.5 XQuery 语法 …… 58
5.6 XQuery 添加元素和属性 …… 59
5.7 XQuery 选择和过滤 …… 61
5.8 XQuery 函数 …… 62
第 6 章 XLink 和 XPointer 教程 …… 65
6.1 XLink 和 XPointer 简介 …… 65
6.2 XLink 和 XPointer 语法 …… 65
6.3 XLink 和 XPointer 实例 …… 66
第 7 章 DTD 教程 …… 69
7.1 DTD 简介 …… 69
7.2 DTD 元素 …… 70
7.3 DTD 属性 …… 72
7.4 DTD 验证 …… 74
7.5 DTD 实例 …… 75
第 8 章 XML Schema 教程 …… 78
8.1 XML Schema 简介 …… 78
8.2 XML Schema 的优点 …… 78
8.3 如何使用 XSD …… 79
8.4 XML Schema 规范 …… 80
8.5 一个 XSD 实例 …… 101
8.6 XSD 数据类型 …… 106
第 9 章 XML 数据接口 DOM …… 114
9.1 XML DOM 简介 …… 114
9.2 XML 节点和节点树 …… 114
9.3 解析 XML DOM …… 117
9.4 XML DOM 加载函数 …… 118
9.5 XML DOM 的属性和方法 …… 119
9.6 XML DOM 访问节点 …… 121
9.7 获取 XML DOM 节点信息 …… 123
9.8 定位 DOM 节点 …… 124
9.9 XML DOM 节点操作 …… 126
第 10 章 XForms 概述 …… 133
10.1 XForms 简介 …… 133
10.2 XForms 的特点 …… 133

10.3 XForms 框架 …… 134
10.4 XForms 命名空间 …… 136
10.5 一个 XForms 实例 …… 137
10.6 XForms 绑定 …… 137
10.7 XForms 输入控件 …… 139
10.8 XForms 选择控件 …… 140
10.9 XForms 数据类型 …… 142
10.10 XForms 属性 …… 142
10.11 XForms 行为 …… 143
第 11 章 简单对象访问协议 SOAP …… 144
11.1 SOAP 简介 …… 144
11.2 SOAP 语法 …… 144
11.3 SOAP HTTP Binding …… 149
11.4 一个 SOAP 实例 …… 150
第 12 章 网络服务描述语言 WSDL …… 152
12.1 WSDL 简介 …… 152
12.2 WSDL 文档 …… 152
12.3 WSDL 端口 …… 153
12.4 WSDL 绑定 …… 155
12.5 WSDL 和 UDDI …… 156
12.6 WSDL 语法 …… 156
第 13 章 RDF 概述 …… 159
13.1 RDF 简介 …… 159
13.2 RDF 规则 …… 159
13.3 RDF 实例 …… 160
13.4 RDF 主要元素 …… 161
13.5 RDF 容器元素 …… 163
13.6 RDF 集合 …… 164
13.7 RDF Schema (RDFS) …… 165
13.8 RDF 都柏林核心元数据倡议 …… 166
第 14 章 RSS 概述 …… 168
14.1 RSS 简介 …… 168
14.2 RSS 语法 …… 168
14.3 RSS <channel> 元素 …… 169
14.4 RSS <item> 元素 …… 170
14.5 RSS 发布 feed …… 171
14.6 RSS 阅读器 …… 172

第 15 章　WAP 概述 …… 173
15.1　WAP 简介 …… 173
15.2　WMLScript …… 173
15.3　WAP 基础 …… 177
15.4　WML 格式化 …… 178
15.5　WML 链接和图像 …… 181
15.6　WML 输入 …… 182
15.7　WML 任务 …… 185
15.8　WML 计时器 …… 186
15.9　WML 变量 …… 187
15.10　WML 实例 …… 187
第 16 章　Web Services 概述 …… 190
16.1　Web Services 简介 …… 190
16.2　Web Services 两种类型的应用 …… 191
16.3　一个实例:ASP. NET Web Service …… 191
16.4　ASP. NET 的自动化处理 …… 192
16.5　使用表单访问 Web Service …… 192
附录一　XSLT 元素 …… 194
附录二　XLink 属性 …… 196
附录三　XSD 参考手册 …… 197
附录四　XForms 数据类型 …… 200
附录五　RSS 参考手册 …… 203
附录六　WML 参考手册 …… 205
参考文献 …… 208

第1章 XML 概 述

XML(Extensible Markup Language)即可扩展标记语言,它与 HTML 一样,都是 SGML(Standard Generalized Markup Language,标准通用标记语言)。XML 是 Internet 环境中跨平台的,依赖于内容的技术,是一种简单的数据存储语言,同时也是当前处理结构化文档信息的有力工具。本章将向读者展示 XML 的特点及它的主要内容,引导读者对 XML 有一个入门性的了解。

1.1 XML 简 介

(1) XML 指可扩展标记语言(Extensible Markup Language);
(2) XML 是一种标记语言,类似 HTML;
(3) XML 的设计宗旨是传输数据,而非显示数据;
(4) XML 标签没有被预定义,您需要自行定义标签;
(5) XML 被设计为具有自我描述性;
(6) XML 是 W3C 的推荐标准。

1.1.1 XML 与 HTML 的主要差异

XML 不是 HTML 的替代品,XML 和 HTML 是为不同的目的而设计的。HTML 被设计用来显示数据,其焦点是数据的外观,XML 被设计用来传输和存储数据,其焦点是数据的内容。HTML 旨在显示信息,而 XML 旨在传输信息。

XML 是没有任何行为的,也许这有点难以理解,但是 XML 不会做任何事情。XML 被设计用来结构化数据、存储以及传输信息。

下面是 John 写给 George 的便签,存储为 XML:

```
<note>
    <to>George</to>
    <from>John</from>
    <heading>Reminder</heading>
    <body>Don't forget the meeting!  </body>
</note>
```

上面的这条便签具有自我描述性。它拥有标题以及留言,同时包含了发送者和接收者的信息。

但是,这个 XML 文档仍然没有做任何事情,它仅仅是包装在 XML 标签中的纯粹的信息。需要编写软件或者程序,才能传送、接收和显示这个文档。

特别提醒一下,XML 仅仅是纯文本而已,有能力处理纯文本的软件都可以处理 XML。能够读懂 XML 的应用程序可以有针对性地处理 XML 的标签。标签的功能性意义依赖于

应用程序的特性。在这里读者通过 XML 可以定义并使用自己的标签。

上例中的标签没有在任何 XML 标准中定义过,比如 <to> 和 <from>,这些标签是由文档的创作者定义的。这是因为 XML 没有预定义的标签,而在 HTML 中使用的标签(包括 HTML 的结构)是预定义的。HTML 文档只使用在 HTML 标准中定义过的标签,比如 <p>、<h1>等。XML 允许创作者定义自己的标签和自己的文档结构。XML 不是对 HTML 的替代,而是对 HTML 的补充。

XML 不会替代 HTML,理解这一点很重要。在大多数 Web 应用程序中,XML 用于传输数据,而 HTML 用于格式化并显示数据。

对 XML 最好的描述是:XML 是独立于软件和硬件的信息传输工具。XML 于 1998 年 2 月 10 日成为 W3C 的推荐标准。

1.1.2 XML 的用途

目前,XML 在 Web 中起到的作用不亚于一直作为 Web 基石的 HTML。XML 是各种应用程序之间进行数据传输的最常用的工具,并且在信息存储和描述领域变得越来越流行。XML 应用于 Web 开发的许多方面,常用于简化数据的存储和共享。

如果需要在 HTML 文档中显示动态数据,那么每当数据改变时将花费大量的时间来编辑 HTML。通过 XML,数据能够存储在独立的 XML 文件中。这样就可以专注于使用 HTML 进行布局和显示,并确保修改底层数据时不需要对 HTML 进行任何的改变。

通过使用几行 JavaScript,就可以读取一个外部 XML 文件,然后更新 HTML 中的数据内容。

使用 XML 可以获得如下的好处。

1. XML 简化了数据共享

在真实的世界中,计算机系统和数据使用不兼容的格式来存储数据。XML 数据以纯文本格式进行存储,因此提供了一种独立于软件和硬件的数据存储方法,这让不同应用程序共享数据变得更加容易。

2. XML 简化了数据传输

通过 XML,可以在不兼容的系统之间轻松地交换数据。对开发人员来说,在网络上的不兼容系统之间交换数据,一直是一项最费时的挑战。由于可以通过各种不兼容的应用程序来读取相同的 XML 数据,因此以 XML 交换数据降低了这种复杂性。

3. XML 简化了平台的变更

升级到新的系统(硬件或软件平台)是非常费时的。因为必须转换大量的数据,所以不兼容的数据经常会丢失。XML 数据以文本格式存储,这使得 XML 在不损失数据的情况下,更容易扩展或升级到新的操作系统、新应用程序或新的浏览器。

4. XML 使数据更有用

由于 XML 独立于硬件、软件以及应用程序,因此可以使数据更可用,也更有用。不同的应用程序都能够访问这些数据,不仅仅在 HTML 页中,甚至也可以从 XML 数据源中进行访问。通过 XML,这些数据可供各种阅读设备使用(如手持的计算机、语音设备、新闻阅读器等),甚至还可以供盲人或其他残障人士使用。

5. XML 用于创建新的 Internet 语言

很多新的 Internet 语言是通过 XML 创建的,其中的例子包括:

(1) XHTML,最新的 HTML 版本;

(2) WSDL,用于描述可用的 Web Service 的语言;

(3) WAP 和 WML,用于手持设备的标记语言;

(4) RSS,用于 RSS feed 的语言;

(5) RDF 和 OWL,用于描述资源和本体的语言;

(6) SMIL,用于描述针对 Web 的多媒体的语言。

未来也许会出现某种字处理软件、电子表格程序以及数据库,它们可以使用纯文本格式读取彼此的数据,而不需要使用任何的转换程序。

1.2 XML 树结构

XML 文档采用的是一种树结构,它从"根部"开始,然后扩展到"枝叶"。

XML 使用简单的具有自我描述性的语法,一个 XML 文档实例如下:

```
<? xml version = "1.0" encoding = "ISO-8859-1"? >
<note >
    <to > George </to >
    <from > John </from >
    <heading > Reminder </heading >
    <body > Don't forget the meeting! </body >
</note >
```

第一行是 XML 声明,它定义 XML 的版本 (1.0) 和所使用的编码 (ISO-8859-1 = Latin-1/西欧字符集)。

<note > 描述文档的根元素,接下来 4 行描述根的 4 个子元素(to, from, heading 以及 body):

```
<to > George </to >
<from > John </from >
<heading > Reminder </heading >
<body > Don't forget the meeting! </body >
```

最后一行定义根元素的结尾:

```
</note >
```

从本例可以得知,该 XML 文档包含了 John 给 George 的一张便签。

XML 文档必须包含根元素,该元素是所有其他元素的父元素。每个 XML 文档有且只能有一个根元素。XML 文档中的元素形成了一棵文档树。这棵树从根部开始,并扩展到树的最底端。所有元素均可拥有子元素,示例如下:

```
<root >
  <child >
    <subchild > ... </subchild >
  </child >
</root >
```

父、子以及同胞等术语用于描述元素之间的关系。父元素拥有子元素。相同层级上的

子元素称为同胞(兄弟或姐妹)。所有元素均可拥有文本内容和属性(类似 HTML)。

XML 示例如下:

```
<bookstore>
    <book category="COOKING">
      <title lang="en">Everyday Italian</title>
      <author>Giada De Laurentiis</author>
      <year>2005</year>
      <price>30.00</price>
    </book>
    <book category="CHILDREN">
      <title lang="en">Harry Potter</title>
      <author>J K. Rowling</author>
      <year>2005</year>
      <price>29.99</price>
    </book>
    <book category="WEB">
      <title lang="en">Learning XML</title>
      <author>Erik T. Ray</author>
      <year>2003</year>
      <price>39.95</price>
    </book>
</bookstore>
```

上面示例可以表示成图 1.1 的树形结构。

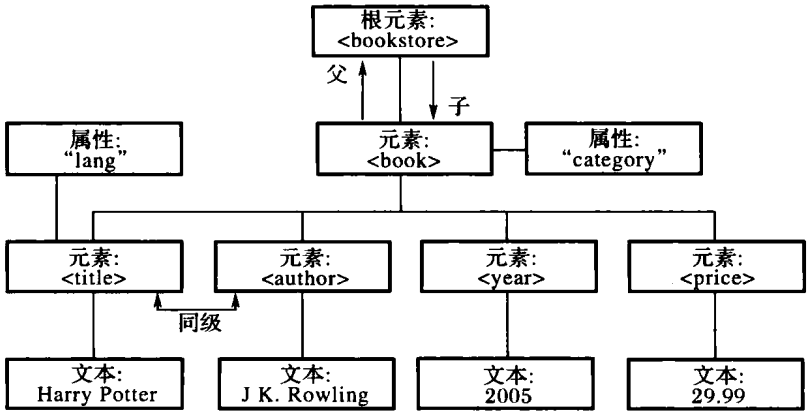

图 1.1 XML 的树形结构

例子中的根元素是 <bookstore>,文档中的所有 <book> 元素都被包含在 <bookstore> 中。<book> 元素有 4 个子元素:<title>、<author>、<year>、<price>。

1.3 XML 语法规则

XML 的语法规则很简单,且很有逻辑。这些规则很容易学习,也很容易使用。

1. 所有 XML 元素都需有关闭标签

在 HTML,经常会看到没有关闭标签的元素,如:

```
<p>This is a paragraph
<p>This is another paragraph
```

在 XML 中,省略关闭标签是非法的,所有元素都必须有关闭标签,如:

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

注释:这里读者也许已经注意到 XML 声明没有关闭标签,这不是错误。声明不属于 XML 本身的组成部分,它不是 XML 元素,也不需要关闭标签。

2. XML 标签对大小写敏感

XML 元素使用 XML 标签进行定义。XML 标签对大小写敏感,在 XML 中,标签 <Letter> 与标签 <letter> 是不同的。必须使用相同的大小写来编写打开标签和关闭标签,如:

```
<Message>这是错误的。</message>
<message>这是正确的。</message>
```

注释:打开标签和关闭标签通常被称为开始标签和结束标签。不论使用哪种术语,它们的概念都是相同的。

3. XML 必须正确嵌套

在 HTML 中,常会看到没有正确嵌套的元素,如:

```
<b><i>This text is bold and italic</b></i>
```

在 XML 中,所有元素都必须彼此正确嵌套,如:

```
<b><i>This text is bold and italic</i></b>
```

在上例中,正确嵌套的意思是:由于 <i> 元素是在 <b> 元素内打开的,那么它必须在 <b> 元素内关闭。

4. XML 文档必须有根元素

XML 文档必须有一个元素是所有其他元素的父元素,该元素称为根元素,示例如下:

```
<root>
  <child>
    <subchild>...</subchild>
  </child>
</root>
```

5. XML 的属性值需加引号

与 HTML 类似,XML 也可拥有属性。在 XML 中,XML 的属性值需加引号。请研究下面的两个 XML 文档,第一个是错误的,第二个是正确的:

(1) 错误的文档

```
<note date =08/08/2008 >
  <to > George </to >
  <from > John </from >
</note >
```

(2) 正确的文档

```
<note date ="08/08/2008" >
  <to > George </to >
  <from > John </from >
</note >
```

第一个文档的错误是 note 元素中的 date 属性没有加引号。

6. 实体引用

在 XML 中,一些字符拥有特殊的意义。如果把字符“ <”放在 XML 元素中,会发生错误,这是因为解析器会把它当作新元素的开始,这样会产生 XML 错误,示例如下:

```
<message > if salary < 1000 then </message >
```

为了避免这个错误,请用实体引用来代替“ <”字符,示例如下:

```
<message > if salary &lt; 1000 then </message >
```

在 XML 中,有 5 个预定义的实体引用:

① <: <小于。

② >: >大于。

③ &:和号。

④ ':'单引号。

⑤ ":"引号。

注释:在 XML 中,只有字符“ <”和“&”是非法的。大于号是合法的,但是用实体引用来代替它是一个好习惯。

7. XML 中的注释

在 XML 中编写注释的语法与 HTML 的语法很相似,语法如下:

```
<! -- This is a comment -- >
```

在 XML 中,空格会被保留,HTML 会把多个连续的空格字符裁减(合并)为一个:

HTML:Hello my name is David.

输出:Hello my name is David.

在 XML 中,文档中的空格不会被删减,XML 以 换行符(LF)来存储换行。

在 Windows 应用程序中,换行通常以一对字符来存储,即回车符 (CR) 和换行符 (LF),这对字符与打字机设置新行的动作有相似之处。在 Unix 应用程序中,新行以 LF 字符存储,而 Macintosh 应用程序使用 CR 来存储新行。

8. XML 中的空元素

没用内容的元素称为空元素,注意空元素也可以具有属性。

例如:

```
<banner > <banner >
<banner id ="b001" > <banner >
```

空元素也可以这样简写:

```
<banner/>
<banner id="b001"/>
```

1.4 XML 元 素

XML 元素指的是从(且包括)开始标签到(且包括)结束标签的部分。元素可包含其他元素、文本或者两者的混合物。元素也可以拥有属性。

```
<bookstore>
<book category="CHILDREN">
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
<book category="WEB">
  <title>Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>
</bookstore>
```

在上例中,<bookstore> 和 <book> 都拥有元素内容,因为它们包含了其他元素。<author> 只有文本内容,因为它仅包含文本。

在上例中,只有 <book> 元素拥有属性(category="CHILDREN")。

1. XML 命名规则

XML 元素必须遵循以下命名规则:

(1) 名称可以含字母、数字以及其他的字符;

(2) 名称不能以数字或者标点符号开始;

(3) 名称不能以字符“xml”(或者 XML,Xml)开始;

(4) 名称不能包含空格。

在符合规则的前提下,可以使用任何的名称,并且 XML 没有保留的字词。

2. 最佳命名习惯

为使名称具有描述性,可使用带下画线的名称。

名称应当比较简短,比如:<book_title>,而 <the_title_of_the_book> 不是一个合适的名字。

(1) 避免“-”字符。如果您按照这样的方式进行命名,如“first-name”,一些软件会认为您是需要提取第一个单词。

(2) 避免“.”字符。如果您按照这样的方式进行命名,如“first.name”,一些软件会认为“name”是对象“first”的属性。

(3) 避免“:”字符。冒号会被转换为命名空间来使用(稍后介绍)。

XML 文档经常会有一个对应的数据库,其中的字段会对应 XML 文档中的元素。有一个实用的经验,即使用数据库的名称规则来命名 XML 文档中的元素。

非英语的字母比如 éòá 也是合法的 XML 元素名,不过需要留意当软件不支持这些字符时可能出现的问题。

3. XML 元素是可扩展的

XML 元素是可扩展的,以便携带更多的信息。请看下面这个 XML 例子:

```
<note>
  <to>George</to>
  <from>John</from>
  <body>Don't forget the meeting!</body>
</note>
```

考虑创建一个应用程序,可将 <to>、<from> 以及 <body> 元素提取出来,并产生以下的输出:

MESSAGE

To: George

From: John

Don't forget the meeting!

假设这个 XML 文档作者又向这个文档添加了一些额外的信息,如下:

```
<note>
  <date>2008-08-08</date>
  <to>George</to>
  <from>John</from>
  <heading>Reminder</heading>
  <body>Don't forget the meeting!</body>
</note>
```

那么这个应用程序会中断或崩溃吗? 答案是不会。这个应用程序仍然可以找到 XML 文档中的 <to>、<from> 以及 <body> 元素,并产生同样的输出。XML 的优势之一,就是可以经常在不中断应用程序的情况下进行扩展。

1.5 XML 属 性

从 HTML 中,读者也许会回忆起:<img src = "computer. gif" >。"src"属性可提供有关 <img> 元素的额外信息。

在 HTML 中(以及在 XML 中),属性可提供有关元素的额外信息,示例如下:

```
<img src = "computer. gif" >
<a href = "demo. asp" >
```

属性通常提供不属于数据组成部分的信息。在下面的例子中,文件类型与数据无关,但是对需要处理这个元素的软件来说却很重要:

```
<file type = "gif" >computer. gif</file>
```

1. XML 属性必须加引号

属性值必须被引号包围,单引号和双引号均可使用。比如一个人的性别,person 标签可以这样写:

```
<person sex = "female" >
```

或者这样也可以:

```
<person sex = 'female' >
```

注意:如果属性值本身包含双引号,那么有必要使用单引号包围它,就像这个例子:

```
<gangster name = 'George "Shotgun" Ziegler' >
```

或者可以使用实体引用:

```
<gangster name = "George &quot;Shotgun&quot; Ziegler" >
```

关于 XML 元素和属性的对比,请看这些例子:

```
<person sex = "female" >
  <firstname > Anna </firstname >
  <lastname > Smith </lastname >
</person >
<person >
  <sex > female </sex >
  <firstname > Anna </firstname >
  <lastname > Smith </lastname >
</person >
```

在第一个例子中,sex 是一个属性;在第二个例子中,sex 则是一个子元素。两个例子均可提供相同的信息。

没有什么规定可以告诉我们什么时候该使用属性,而什么时候该使用子元素。通常在 HTML 中,属性用起来很方便,但是在 XML 中,应该尽量避免使用属性。如果信息感觉起来很像数据,那么建议使用子元素。

下面的三个 XML 文档包含完全相同的信息:

第一个例子中使用了 date 属性:

```
<note date = "08/08/2008" >
  <to > George </to >
  <from > John </from >
  <heading > Reminder </heading >
  <body > Don't forget the meeting! </body >
</note >
```

第二个例子中使用了 date 元素:

```
<note >
  <date > 08/08/2008 </date >
  <to > George </to >
  <from > John </from >
  <heading > Reminder </heading >
  <body > Don't forget the meeting! </body >
```