XML
COMPLETE

XML

使用详解

(美) Steven Holzner 著
潇湘工作室 译

- XML语言规范
- XML链接规范
- XML样式表规范

机械工业出版社

McGraw Hill

CMP

本书详细介绍了 XML 的三个规范：XML 语言规范、XML 链接规范和 XML 样式表规范。其中列举了很多简短而有说服力的实例，其源代码包括在本书的配套光盘中。阅读本书，用户可以学会 XML 编程的细节，定义每个 Web 站点的标记，并指定标记的处理方式，掌握视频集成，并创建分析 XML 的 Java 程序。读者将能够创建有效而且格式良好的文档，设计实际的 Web 页，并提高自己的工作效率。

本书版权登记号：图字：01-98-1912

图书在版编目（CIP）数据

XML 使用详解/（美）霍尔兹纳（Holzner, S.）著；潇湘工作室译，－北京：机械工业出版社，1999.1

书名原文：XML Complete

ISBN 7-111-06950-1

Ⅰ.X… Ⅱ.①霍… ②潇… Ⅲ.计算机网络-可扩充语言，XML Ⅳ.TP393

中国版本图书馆 CIP 数据核字（98）第 34375 号

出 版 人：马九荣（北京市百万庄大街 22 号 邮政编码 10037）

责任编辑：温莉芳 于 静

北京昌平第二印刷厂印刷·新华书店北京发行所发行

1999 年 1 月第 1 版第 1 次印刷

787mm×1092mm 1/16·30 印张

印数：0001-5000 册

定价：62.00 元（附光盘）

前　言

不必说，World Wide Web 已经成为一种编程现象。Web 已经脱离出来，而且越来越通用。然而，当 Web 用于更多的领域时（人们试图把它用于设计其他的 Web 协议），生活会变得更加复杂。Extended Markup Language（可扩展的标记语言，简称 XML）的创建，使 Web 的使用变得更为简化。

当前，最流行的 Web 标记语言是 HTML。这种语言是开发 Web 的强有力的工具，但是，它缺乏专业化的性能，这就限制了它的快速发展。所有的 HTML 都是已定义好了的，如果有一个专业化的任务，例如，以容易阅读的方式，在 Web 网上为一个医院存储和传送记录，那么 HTML 就不是最好的选择。HTML 规定好如何呈现一个 Web 页数据的格式，而不是使用那些数据所代表的意义。

而 XML 就更灵活一些，这是因为在 XML 中，你可以定义自己的标记元素。这就允许你按自己的需要剪裁一个文档，存储和组织那个文档中自己喜爱的数据，这远远超出了 HTML＜p＞和＜DIV＞标记所能为你做的。例如，如果你想存储医院记录，那么可以创建一个＜patient＞元素，而且它也可以包含很多其他元素，如＜medication＞、＜room_number＞、＜admission_data＞等等。通过定义自己的元素，XML 允许你按自己的需要构造一个文档，而不必重新构造数据，以符合已定义的标记语言。

在本书中，我们介绍了 XML 的三个规范：XML_Lang（XML 语言规范）、XML_Link（XML 链接规范）、XML-Style（XML 样式表规范）。这三部分就组成了 XML，对于这三种规范，本书将逐一进行讨论。

使用 XML，我们会明白如何定制 Web 上使用的文档，以及如何创建健壮的应用程序，而这些应用程序会使用和解释这些文档。这样，我们可以采取继 HTML 之后的下一步，以“聪明”的方式存储文档，这就会对将要创建的应用程序解释数据存储技巧。我们就可以专业化自己的文档，这远远超出了 HTML 的性能，在 Web 成为数据传送的重要媒体方面，向前迈进了一大步。

本书内容

本书介绍了 XML 的三个规范：XML 语言规范、XML 链接规范和 XML 样式表规范。

为检查 XML 规范，我们列举了很多简短而有说服力的实例，因为看一个项目运行（从开始到结束）是学习 XML 的最好方式。列举的实例都很简明扼要，不必弄懂很多无关的细节。我们将创建 XML 应用程序，如浏览器、数据库处理程序、文档解码器等等，把这些实例与现实使用结合起来。

下面以标题顺序概述了本书所讲述的内容：

- 使用 Java
- 创建窗口化的 Java 应用程序
- 使用 Java 控件

- 使用 Microsoft XML 分析器
- 阅读 XML 文档
- 显示 XML 文档的文本
- 检索 XML 文档的标题
- 采用递归方式打开 XML 文档
- 创建自己的 XML 标记
- 声明 XML 标记
- 声明空的 XML 标记
- 解释 XML 标记
- 创建“有效的”和“格式良好的”XML 文档
- 使用外部 DTD
- 使用文本缩进的 XML 浏览器
- 获得 XML 文档的名字
- 可扩展的 Backus－Naur 符号系统
- 访问 XML 子元素
- 在 XML 文档中，用代码构造新元素
- 创建并使用 XML 标记属性
- 处理 XML 属性列表
- 创建 XML 浏览器
- 在 XML 浏览器中，解码嵌套的 XML 标记
- 创建 XML 数据库
- 搜索 XML 数据库
- 给 XML 数据库添加记录
- 将 XML 数据库写到磁盘上
- 在 XML 数据库中，处理变长记录
- 使用 XML 链接
- 使用 XML 定位器
- 使用 XML 扩展的链接
- 创建 XML 扩展的链接组
- 在 XML 浏览器中使用文本
- 在 XML 中使用图形
- 创建一个画圆圈的 XML 浏览器
- 在 XML 文档中交叉引用
- 使用完全图形化的 XML 浏览器
- 用 XML 处理图像
- 创建 XML 图像浏览器
- 在 XML 浏览器中，指定字体大小和字号
- 创建 XML 图像映像
- 使用 XML 样式表

- 使用 DSSSL、文档样式语义学和规范语言
- 定义 DSSSL 术语
- 创建 SGML 文档
- 区别 SGML 与 XML 之间的不同点
- 使用 James Clark 的 Jade DSSSL 处理器
- 从 XML 中格式化多信息文本文档
- 在 XML 浏览器中使用菜单
- 在 XML 浏览器中，使用 Java 单选按钮和复选框
- 其他内容

当我们学完这本书之后，将会看到实例中都列举了这些标题。并打算让 XML 为我们服务。

读者的需要

当前，很多 XML 软件都使用 Java 语言，而且，要想更好地利用本书，你应该在计算机上安装 Java；本书使用 Java 最通用的版本：Java 1.0.2。从 Sun Microsystem 站点（http：//www.javasoft.com）处，你可以获得一份 Java Development Kit（Java 开发工具包，简称 JDK）。你也应该拥有一个字处理器或编辑器，用以编写 Java 代码文件，在第 1 章，我们就会看到。

第 1 章之后，除第 9 章以外的所有其他章节，都将使用 Microsoft XML 分析器。在 Microsoft 站点 http：//www.microsoft.com/standards/xml/xmlparse.htm 处，你可以得到 Microsoft XML 分析器的一个备份。你可以安装 Microsoft XML 分析器，它由输入 Java 程序的 Java 类组成，而且 Microsoft 站点处的指令用起来也很容易。

最后，在第 9 章中，我们将讨论 XML 的第三种规范：XML－Style（简称 XS）。XML 支持使用 Document Style Semantics and Specification Language（文档样式语义学和规范语言，简称 DSSSL）的子集，而且我们将用 DSSSL 格式化 XML 文档的外观用于显示。在第 9 章中，通过 XS，我们使用 James Clark 的 Jade DSSSL 处理器，来创建格式化的多信息文本文件（RTF 格式）。在本书随带的 CD 盘上或在 James Clark 的 Jade 站点（http：//www.jclark.com/jade）处，你可以得到一份 Jade 处理器。

另外，在下面这些统一资源定位器处，你可以找到 XML 的三种规范：第Ⅰ部分：http：//www.w3.org/TR/WD－xml；第Ⅱ部分：http ：//www.w3.org/TR/WD－xml－link；第Ⅲ部分：http：//sunsite.unc.edu/pub/sun－info/standards/dsssl/xs/xs970522.rtf.zip（注意，xs970522 文档只是一个临时版本，它将成为 XML 规范的第 III 部分；关于第Ⅲ部分 XML 规范的最新版本，请查阅站点 http：//www.w3.org/XML/Activity）。这三类文档组成了正式的 XML 规范，而且对于本书未提供的所有答案，你应该参考它们。

就到这儿吧。这就是我们所需要的。我们准备出发吧。由于很多 XML 软件（例如，将要使用的 Microsoft XML 分析器）都是用 Java 语言编写的，所以本书从头到尾都使用 Java 语言。在第 1 章中，在讨论 XML 之前，我们会花一点儿时间了解 Java（如果你很熟悉 Java 语言，那么只须快速浏览一下第 1 章，然后跳转到第 2 章）。

目　录

第1章 准备出发

欢迎学习 XML 编程！本书展示 XML 提供的所有功能。我们将了解全部内容：从 XML 的基本知识到高级主题；从创建自己的 XML 文档到使用 XML 样式表，从创建 XML 链接到创建 XML 浏览器。

1.1 什么是 XML

确切地讲，XML 到底是什么呢？其完整名称是 Extensible Markup Language（可扩展的标记语言），缩写为 XML。使用它，就可以以容易而一致的方式格式化和传送数据（通常是在 World Wide Web 网上）。实际上，将 XML 设计成 Standard Generalized Markup Language（标准通用标记语言，简称 SGML）的一个子集，它是简化的，而且目标是面向 Web。HTML 是 SGML 的另外一个子集。事实上，如果很熟悉 Hypertext Markup Language（超文本标记语言，简称 HTML），那么再来学习 XML 就会很省劲。例如，看看下面的 HTML Web 页：

```
<HTML>

<HEAD>
<TITLE>The Best Web Page Ever! </TITLE>
</HEAD>

<BODY>
<CENTER>
<H1>
<B>This is the Best Web Page Ever! </B>
</H1>
</CENTER>

Welcome to the best Web Page ever!

</BODY>

</HTML>
```

这里，我们使用 HTML 标记来指定文本在 Web 页上的显示方式：

标识标记

HTML 和 XML 语言中的标记都是用尖括号＜和＞括起来的文本字符串，而且对读入 HTML 或 XML 文档的应用程序来说，它们是伪指令。例如，在上述 Web 页中的第一个标

记是<HTML>：

```
<HTML>
    .
    .
    .
```

这就会通知 Web 浏览器：文档是用 HTML 编写的，所以它是固有的 Web 页。下一步，使用<HEAD>和<TITLE>标记设置 Web 页标题：

```
<HTML>

<HEAD>
<TITLE>The Best Web Page Ever! </TITLE>
</HEAD>
    .
    .
    .
```

这就设置了 Web 页的标题，正如浏览器中所显示的（通常处于顶部浏览器的标题栏中）；在本例中，Web 页标题是“The Best Web Page Ever!”（在某种程度上有点儿夸张）。

对每个起始标记，如本例中的<HEAD>或<TITLE>，都有一个结束标记，如</HEAD>或</TITLE>。结束标记与起始标记使用同一名字，只是多了一条斜线；例如，起始标记为<HEAD>时，其结束标记为</HEAD>。在这种情况下，起始标记和结束标记都成对出现，把 Web 页文本括了起来，并指定以何种方式显示文本。

注意 并不是所有的起始标记都需要结束标记，但大多数都需要。

例如，在 Web 页主体内（用<BODY>标记指定），我们用<CENTER>标记使文本居中：

```
<HTML>

<HEAD>
<TITLE>The Best Web Page Ever! </TITLE>
</HEAD>

<BODY>
<CENTER>
<H1>
<B>This is the Best Web Page Ever! </B>
<H1>
</CENTER>
    .
    .
    .
```

另外，我们使用<H1>标记来以大字体显示页的标题（HTML 标题可以从最大字体

<H1>到最小字体<H6>中进行选择），而且用<B>标记进行加粗显示：

```
<HTML>

<HEAD>
<TITLE>The Best Web Page Ever! </TITLE>
</HEAD>
<BODY>
<CENTER>
<H1>
<B>This is the Best Web Page Ever! </B>
</H1>
</CENTER>
      .
      .
      .
```

这样，我们使用 HTML 指定如何显示 Web 页中的文本和图形。

XML 与 HTML 有所区别，但也有共同点。HTML 描述了如何显示 Web 页中的数据，而 XML 描述数据本身。

换句话说，作为一种数据描述语言的 XML，使用的频率会更高，如果选择了它，就允许我们将数据组织成数据结构，甚至是很复杂的数据结构。你可以按照自己的需要剪裁数据；XML 最有吸引力的特征是你可以创建自己的标记。这就允许你在 XML 文档中以自己的方式组织数据。下面让我们看一个 XML 实例，以使这个概念更清楚。

1.2 一个 XML 实例

这个例子名为 grocery.xml，我们记录杂货店中的每位顾客所购买的商品及购买时间，设置一个 XML 文档记录这些数据。现在，让我们看看如何完成这项工作。

我们以一条处理指令开始 grocery.xml 程序，这条指令显示出此文档是 XML 文档，而且是使用 1.0 版本的 XML：

```
<? XML version = "1.0" ? >
    .
    .
    .
```

处理指令以<？开始，以？>结束，而且它们直接传给 XML 处理器，读入并解释 XML 的应用程序。在本例中，我们将 XML 版本通知给 XML 处理器。

接下来的内容是 grocery.xml 的 XML 标记。以<DOCUMENT>标记开始（“DOCUMENT”是我们自己选择的标记，并没有已定义的 XML DOCUMENT 标记），来显示我们要启动 XML 文档了：

```
<? XML version = "1.0" ? >
```

```
<DOCUMENT>
    .
    .
    .
```

此标记启动一个新 XML 文档。我们可以一个顾客接一个顾客地构造 XML 文档 grocery.txt，在<CUSTOMER>和</CUSTOMER>标记之间列出每个顾客的货物：

```
<? XML version = "1.0" ? >
<DOCUMENT>
<CUSTOMER>
    .
    .
    .
</CUSTOMER>
```

接下来，存储这个顾客的名字。我们可以在 XML 中嵌套任意深度的标记；在本例中，就意味着，通过创建和使用一个<NAME>标记，我们可以为每个顾客设置一个名字部分，如下所示：

```
<? XML version = "1.0" ? >
<DOCUMENT>
<CUSTOMER>
     <NAME>
        .
        .
        .
    </NAME>
</CUSTOMER>
```

我们可以调用<FIRSTNAME>和<LASTNAME>标记，进一步存储顾客的第一个及最后一个名字：

```
<? XML version = "1.0" ? >
<DOCUMENT>
<CUSTOMER>
      <NAME>
          <LASTNAME>Edwards</LASTNAME>
          <FIRSTNAME>Britta</FIRSTNAME>
      </NAME>
      .
      .
      .
</CUSTOMER>
```

现在，通过设置一个<DATE>标记，我们可以存储顾客购买货物的日期：

```
<? XML version = "1.0" ? >
<DOCUMENT>
<CUSTOMER>
    <NAME>
        <LASTNAME>Edwards</LASTNAME>
        <FIRSTNAME>Britta</FIRSTNAME>
    </NAME>
    <DATE>April 17, 1998</DATE>
        .
        .
        .
</CUSTOMER>
```

这时，我们准备存储顾客购买货物的顺序。可以创建一个新<ORDERS>标记，来完成这个任务：

```
<? XML version = "1.0? ">
<DOCUMENT>
<CUSTOMER>
    <NAME>
        <LASTNAME>Edwards</LASTNAME>
        <FIRSTNAME>Britta</FIRSTNAME>
    </NAME>
    <DATE>April 17, 1998</DATE>
    <ORDERS>
        .
        .
        .
    </ORDERS>
</CUSTOMER>
```

我们可以用<ITEM>标记存储顾客顺序购买的每种商品：

```
<? XML version = "1.0" ? >
<DOCUMENT>
<CUSTOMER>
    <NAME>
        <LASTNAME>Edwards</LASTNAME>
        <FIRSTNAME>Britta</FIRSTNAME>
    </NAME>
    <DATE>April 17, 1998</DATE>
    <ORDERS>
        <ITEM>
            .
```

```
        .
        .
    </ITEM>
  </ORDERS>
</CUSTOMER>
```

假定顾客购买了五个黄瓜，那么可以使用三个新标记<PRODUCT>、<NUMBER>和<PRICE>存储购买数量和价钱：

```
<? XML version = "1.0" ? >
<DOCUMENT>
<CUSTOMER>
    <NAME>
        <LASTNAME>Edwards</LASTNAME>
        <FIRSTNAME>Britta</FIRSTNAME>
  </NAME>
  <DATE>April 17, 1998</DATE>
  <ORDERS>
        <ITEM>
            <PRODUCT>Cucumber</PRODUCT>
            <NUMBER>5</NUMBER>
            <PRICE> $ 1.25</PRICE>
      </ITEM>
  </ORDERS>
</CUSTOMER>
```

除了黄瓜以外，假定这个顾客还购买了莴苣。那么我们可以以同样的方式存储这种新商品的数据如下：

```
<? XML version = "1.0" ? >
<DOCUMENT>
<CUSTOMER>
    <NAME>
    <LASTNAME>Edwards</LASTNAME>
    <FIRSTNAME>Britta</FIRSTNAME>
  </NAME>
  <DATE>April 17, 1998</DATE>
  <ORDERS>
      <ITEM>
          <PRODUCT>Cucumber</PRODUCT>
          <NUMBER>5</NUMBER>
          <PRICE> $ 1.25</PRICE>
  </ITEM>
  <ITEM>
      <PRODUCT>Lettuce</PRODUCT>
```

```
        <NUMBER>2</NUMBER>
        <PRICE>$ .98</PRICE>
      </ITEM>
    </ORDERS>
</CUSTOMER>
```

在同一 XML 文档中，我们甚至可以为其他顾客存储数据如下：

```
<? XML version = "1.0" ? >
<DOCUMENT>
<CUSTOMER>
     <NAME>
          <LASTNAME>Edwards</LASTNAME>
          <FIRSTNAME>Britta</FIRSTNAME>
    </NAME>
    <DATE>April 17, 1998</DATE>
    <ORDERS>
          <ITEM>
              <PRODUCT>Cucumber</PRODUCT>
              <NUMBER>5</NUMBER>
              <PRICE>$ 1.25</PRICE>
        </ITEM>
        <ITEM>
            <PRODUCT>Lettuce</PRODUCT>
            <NUMBER>2</NUMBER>
            <PRICE>$ .98</PRICE>
        </ITEM>
    </ORDERS>
</CUSTOMER>
<CUSTOMER>
        <NAME>
             <LASTNAME>Thompson</LASTNAME>
             <FIRSTNAME>Phoebe</FIRSTNAME>
    </NAME>
    <DATE>May 27, 1998</DATE>
    <ORDERS>
         <ITEM>
             <PRODUCT>Banana</PRODUCT>
             <NUMBER>12</NUMBER>
             <PRICE>$ 2.95</PRICE>
        </ITEM>
        <ITEM>
            <PRODUCT>Apple</PRODUCT>
            <NUMBER>6</NUMBER>
            <PRICE>$ 1.50</PRICE>
        </ITEM>
```

```
    </ORDERS>
</CUSTOMER>
</DOCUMENT>
```

这样，通过创建和使用自己的 XML 标记，我们已经存储并构造了需要记录顾客的有关数据。使用一套已定义的标记，就免于受 HTML 的限制。

然而，正如你所知道的，XML 是一种自由形式的语言。它取决于我们是否弄清了所使用标记的意义，而不是一些预写的浏览器应用程序。学完本书，我们就会清楚地知道如何解释 XML 了。

在 XML 中，我们编制自己的标记，来描述和构造欲存储和使用的数据。在较大的文档中，这会导致很多混乱现象：怎样保证已正确设置了 XML 文档呢？也就是说，在上述列表中，如果我们省略了结尾标记</CUSTOMER>会怎样呢？当我们只想存储一个顾客（而不是两个）的数据，XML 处理器可能会产生错误。如果我们省略了顾客的最后名字及相应标记<LASTNAME>和</LASTNAME>，那又会怎么样呢？如果我们把标记放错了位置，会怎么样呢？我们怎样知道呢？

1.2.1 文档类型声明

你可以定义在 XML 文档中使用的标记、它们的排列顺序、其他标记可以包含的标记；在文档的 Document Type Declaration（文档类型声明）或 DTD 中，你可以设置其他规则。在很多 XML 文档中，DTD 并不是严格要求的内容，但是要确保文档有效（这是 XML 的专业术语），你应该引入一个 DTD，于是 XML 处理器就进行检查，以确保 XML 文档遵从已设置的规则。在下一章，我们将更详细地讨论 DTD，下面的实例就是一个为文档创建的 DTD，它指定哪个标记可以包含其他标记、这些标记的排列顺序，以及可包含的数据标记类型（PC－DATA 代表已分析的字符数据或文本，而且＊号表示它所指的项目可以重复）：

```
<? XML version = "1.0" ? >
<! DOCTYPE DOCUMENT [
<! ELEMENT DOCUMENT (CUSTOMER) * >
<! ELEMENT CUSTOMER (NAME, DATE, ORDERS) >
<! ELEMENT NAME (LASTNAME, FIRSTNAME) >
<! ELEMENT LASTNAME (#PCDATA) >
<! ELEMENT FIRSTNAME (#PCDATA) >
<! ELEMENT DATE (#PCDATA) >
<! ELEMENT ORDERS (ITEM) * >
<! ELEMENT ITEM (PRODUCT, NUMBER, PRICE) >
<! ELEMENT PRODUCT (#PCDATA) >
<! ELEMENT NUMBER (#PCDATA) >
<! ELEMENT PRICE (#PCDATA) >
] >
<DOCUMENT>
<CUSTOMER>
        <NAME>
```

```
            <LASTNAME>Edwards</LASTNAME>
            <FIRSTNAME>Britta</FIRSTNAME>
        </NAME>
        <DATE>April 17, 1998</DATE>
        <ORDERS>
            <ITEM>
                <PRODUCT>Cucumber</PRODUCT>
                <NUMBER>5</NUMBER>
                <PRICE>$1.25</PRICE>
            </ITEM>
            <ITEM>
                <PRODUCT>Lettuce</PRODUCT>
                <NUMBER>2</NUMBER>
                <PRICE>$.98</PRICE>
            </ITEM>
        </ORDERS>
        </CUSTOMER>
        <CUSTOMER>
            <NAME>
                <LASTNAME>Thompson</LASTNAME>
                <FIRSTNAME>Phoebe</FIRSTNAME>
        </NAME>
        <DATE>May 27, 1998</DATE>
        <ORDERS>
            <ITEM>
                <PRODUCT>Banana</PRODUCT>
                <NUMBER>12</NUMBER>
                <PRICE>$2.95</PRICE>
            </ITEM>
            <ITEM>
                <PRODUCT>Apple</PRODUCT>
                <NUMBER>6</NUMBER>
                <PRICE>$1.50</PRICE>
            </ITEM>
        </ORDERS>
    </CUSTOMER>
    </DOCUMENT>
```

到现在为止，我们已看到了一个完整的 XML 文档。注意，XML 是自由形式的，这就取决于我们如何解释它。你想像过，一个主要的 Internet 浏览器（如 Internet Explorer）试图阅读上述文档吗？那个浏览器不能识别标记（如<COSTUMER>或<ITEM>）。它应如何处理它们呢？它应该以特殊格式显示那个数据吗？它应该把数据存入一个文件吗？这就取决于我们如何解释自己的文档（正如我们所看到的，已经定义了 XML 的各部分，而且主要浏览器计划要支持它们）。

1.2.2 分析和浏览 XML

要想解释 XML 文档，我们应该对它们进行分析（也就是说，把它们分成逻辑结构）。我们可以使用很多 XML 分析器。分析一个 XML 文档就是把它分成很多部件元素。之后，就取决于我们对结果理解的程度，这是因为只有自己才清楚＜COSTUMER＞或＜ORDERS＞标记中数据的含义。

最近一段时间，几乎所有的 XML 分析器都是用 Java 语言编写的，这就意味着，本书会使用很多 Java 程序。XML 分析器随着已编写的 Java 代码产生，我们把正确的代码放入应用程序之中。我们将会利用程序中的代码阅读 XML 文档，并确定出它们的结构。在下面几节，我们将加速讨论 Java，以熟悉使 XML 分析器运行的代码——如果你已经很了解 Java 语言，那么就复习一下。然后，我们可以用 Java 编写程序，来装入、分析和处理（即根据文档内容所显示的，进行显示、解释、分析）XML 文档。

注意 如果你已非常熟悉 Java 语言及本章的标题，那么可以直接跳转到第 2 章。

现在，让我们开始学习 Java 语言，本章的下面几节都将讨论 Java 语言。下一章，我们就开始分析和利用实际的 XML 文档。这里，我们使用 Java 的 1.02 版本，目前最流行这种版本（实际上，你也可以使用 Java 的其他版本，但是这里我们只使用 1.02 版本）。要想利用本书中的代码（实际上是每个 XML 分析器），应确保计算机上已安装了 Java。你可以从 Java Web 站点 http: //www.javasoft.com 处下载 Java Development Kit（Java 开发工具包，简称 JDK）。欲安装 Java，只需跟随 JDK 的指令提示即可。

1.3 第一个 Java 应用程序：helloapp

我们的第一个 Java 应用程序非常简单，只是打印问候语“Hello”。我们这就开始进军 Java，创建本书所使用的 Java 应用程序的类型。现在我们就看看它是如何运行的。下一章，我们将讨论如何把 XML 分析器和 Java 应用程序连接起来。

目前，学习 Java 是使用 XML 的基础，这是因为几乎所有的 XML 软件都使用 Java，而且你需要用 Java 使用那个软件。XML 是为 Web 设计的，而 Java 是 Web 上最通用的编程语言。本书的所有章节都使用 Java（第 9 章除外）。

我们介绍的第一个 Java 应用程序只打印出问候语“Hello”，所以给它命名为 helloapp。使用一个文本编辑器或字处理器（如果你使用字处理器，应确保以普通文本格式保存 Java 程序，不应包含任何特殊的格式字符），创建文件 helloapp.java。注意 Java 文件的扩展名必须为 .java；如果你的编辑器不能采用此扩展名创建文档，那么用 .jav 扩展名保存文件，之后，再将它的扩展名改为 .java。

首先，把下面的文本放到 helloapp.java 程序中：

```
public class helloapp {
.
.
.
}
```