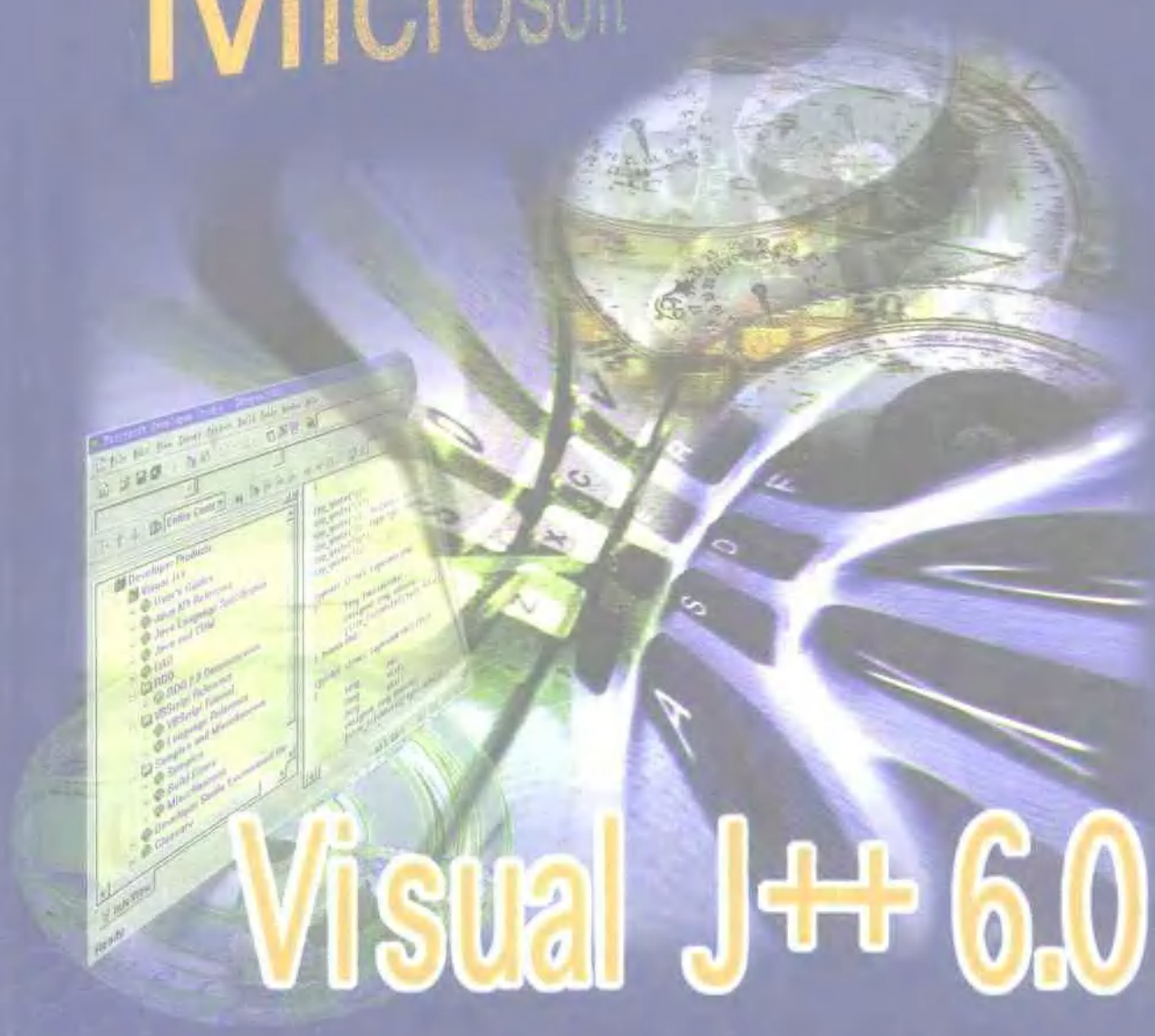
Microsoft Visual Studio 98 系列丛书

曹 伟等 编著

Microsoft Visual J++ 6.0 编程基础

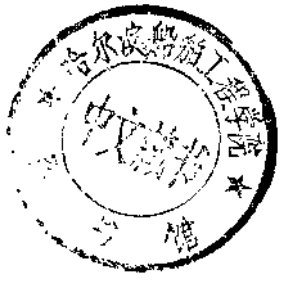


北京航空航天大学出版社

Microsoft Visual Studio 98 系列丛书

Microsoft Visual J++ 6.0 编程基础

曹 伟等 编著



北京航空航天大学出版社

内容简介

微软公司的 Visual J++ 6.0 是最新的 Java 实用程序开发环境，也是最好的 Java 实用程序开发环境之一。本书详细介绍了 Visual J++ 6.0 编程环境和一些编程技巧。全书分为三部分：第一部分是 Java 基础，介绍有关 Java 的概念和 Java 语言语法基础；第二部分介绍 Visual J++ 6.0 环境，内容包括 Visual J++ 6.0 环境、文本域和按钮、Java 布局、文本区域和面板、核选框和单选按钮、滚动条、下拉列表框和滚动表、窗口和菜单以及对话框等；第三部分介绍一些高级程序设计内容，包括图形程序设计、图像处理及动画技术。

本书适合广大初中级 Java 读者参考使用，也可作为各类培训班的教材。本书的高级部分对有一定经验的程序员也有参考价值。

图书在版编目(CIP)数据

Microsoft Visual J++6.0 编程基础/曹伟编著. —北京：北京航空航天大学出版社，1998.11

ISBN 7-81012-832-9

Ⅰ.M… Ⅱ.曹… Ⅲ.Java 语言-程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(98)第 30341 号

Microsoft Visual J++ 6.0 编程基础

编　　著　曹伟等

责任编辑　赵延永

责任校对　陈　坤

北京航空航天大学出版社出版发行

(北京市学院路 37 号(100083)，发行部电话 010-82317024)

http://www.buaapress.cn.net

E-mail:pressell@publica.bj.cninfo.net

朝阳升普印刷厂印装　各地书店经销

开本：787×1092　1/16　印张：18.25　字数：464 千字

1998 年 12 月第 1 版　1998 年 12 月第 1 次印刷　印数：5 200 册

ISBN 7-81012-832-9/TP·819　　定价：24.00 元

前　言

今天，计算技术的发展已经进入了网络时代。随着 Internet 的发展和万维网(World Wide Web)的普及，人们的生活和社会的各种活动将越来越离不开信息网络的支持。Java 是当前最适合于 Internet 应用开发的编程语言，它较好地解决了 Internet 上的异质性、代码交换和网络程序安全性等问题。同时，它也给万维网带来了新的生机和活力，使用 Java 后，原来只有静态图像和文本的网页就可以变成栩栩如生的动画。不仅如此，Java 的出现，对于计算机和信息技术应用方式和应用范围的影响也是广泛而深远的。可以说，Java 正是适应网络时代的需要而产生的。

微软公司的 Visual J++ 6.0 是最新的 Java 实用程序开发环境，也是最好的 Java 实用程序开发环境之一。本书的重点是展示 Visual J++ 6.0 是如何工作的。如果我们要想对某个程序或者功能有所了解，没有比分析它的工作代码更有效的方法了。因此，本书从程序开发者的角度出发，从介绍最简单的网页小程序入手，通过各种实例循序渐进地介绍 Visual J++ 6.0 环境的各种功能和 Java 实用程序的开发技术。本书使用了大量的完整的实例程序，涉及到了非常广泛的应用开发内容。读者可以发现，使用 Visual J++ 6.0 的各种工具，可以直接实现许多(几乎是全部)常规的程序功能，编程人员只需要编写极少的代码，就可以完成一个相对复杂的实用程序的编制工作。由于 Java 是一种面向对象的程序设计语言，因此，本书自始至终贯穿面向对象程序设计的思想和方法。相信本书对从事 Java 程序设计的读者会很有帮助。

很高兴本书能够成书并出版，在此感谢参与本书编制工作的各位同仁，特别要感谢徐晓斌、冯志峰、王路、谢云玲、王江海、黄昱和周定邦等七位同仁的支持和帮助。本书创意由王蕃、石枫负责，曹彬统稿，李海宁审校。全书插图由吴彬彬、李静制作，文字录入由张海燕负责。

由于时间仓促，且我们的水平有限，书中一定有许多不足之处，敬请广大读者不吝指正。

编　者

1998 年 12 月

目　录

第一部分　Java 基础

第二部分 Visual J++ 6.0 环境

第三部分 高级内容

第一部分　Java 基础

第一章　Java 基础

在介绍 Visual J++之前，我们必须先了解一些关于 Java 的知识。本章和下一章就是供不熟悉 Java 的读者参考的，有 Java 编程经验的读者可以略过这两章，直接进入 Visual J++世界。

Java 是一种新的计算机语言，它最显著的特点就是能够用于编写可在 Internet 网上广泛发行的独立于平台的应用程序。实际上，Java 不仅仅是一种计算机语言，而且还是一个由许多相关技术组成的系统，这个系统包括 Java 语言本身、安全模型、Java 实时系统以及其他的一些元素。在本章，我们将简要介绍 Java 系统的各个元素。

本章主要内容如下：

- 怎样用 Java 开发万维网(World Wide Web)应用程序；
- Java 小程序(Applet)和 Java 独立程序(Application)有何区别；
- Java 虚拟机(Java Virtual Machine)和即时编译器(just－in－time compiler)的概念；
- 怎样将一个 Java 小程序加入到网页中；
- Java 安全模型和 COM 对象的关系；
- 与 Visual Basic 和 Visual C++相比，Visual J++在开发网络应用程序方面的优点。

1.1　Java 简介

前面我们提到，Java 远不仅是一种计算机语言。从根本上说，Java 是一个可进行分布计算(distributed computing)的平台和一个支持万维网的实时环境。下面列出 Java 语言的主要特点：

- Java 是面向对象(object oriented)的语言。
- Java 程序是独立于平台(platform independent)的语言。同样的一个 Java 程序既可以运行于 PC 机，也可以运行于 Macintosh 机、UNIX 机以及其他所有能够提供 Java 翻译器(Java interpreter)的操作平台上。
- Java 具有内置安全性。这种安全性防止了网际应用程序对用户计算机的恶意侵犯，如读和写用户机上的文件。这就使网络用户可以放心地从网络上下载代码。

正是以上的特点使得 Java 成为目前最受欢迎的网络语言。下面我们简要介绍 Java 的这些特点。

1.1.1 Java 的起源和历史

Java 起始于 1991 年，同年万维网也出现了。有趣的是，Java 的设计者 James Gosling 的最初目的并不是为了开发网络程序。他最初不过是想为一些消费用的电子器件(如交互电视的顶端盒)设计一个通用环境。当时，他的设计目标是发明一种可以方便地对这些消费器件进行编程的新语言，而这种新语言能够方便地从一个器件移植到另一个器件上。他的这种想法可以说是市场驱动的，因为消费者都希望这些电子器件能够比传统的计算机更好用且更简单。

Java 开发小组最初用 C++作为他们的模型。但很快他们就意识到他们需要一种比 C++安全性和可移植性更好的语言。于是，他们先开发出一种称为 OAK 的语言，后来这种语言被称为 Java。这种语言的语法和功能同 C++差不多，但相比之下更为简单和平台中性化(Platform-neutral)。

到 1992 年秋天，开发小组开发出一种具有小型可视界面的手持遥控设备，这个设备被称为 *7。*7 中有一个叫 Duke 的动画人物，这个人物引导人们操作图形界面，后来 Duke 成为 Java 的吉祥物。

1994 年中期，万维网的成功引起了 Sun 公司的注意。OAK 小组注意到了网页的一个根本问题：它们都是单调乏味的。无论一个网页上有多少嵌入的图形、声音和图像，网络浏览者无法和这个网页交互。虽然开发者可以通过 HTML 将数据从网页上发送到一个基于服务器的程序，但这也是他所能做的全部。网页需要交互性。正是这个灵感使 OAK 小组致力于使 Java 语言向 Internet 和万维网方向发展。

1994 年秋天，Sun 公司发行了 WebRunner 和 HotJava，它们都是用 Java 开发的网页浏览器。网络先锋 Netscape 公司注意到了这种新技术，他们宣布他们流行的浏览器 Netscape Navigator 将支持 Java。

从今天的标准来看，最早的 Java 开发工具是非常原始的，比如 Sun 公司的 Java Developers Kit(JDK)1.0。这类开发环境不仅没有交互功能，而且调试功能也很有限。至 1996 年初，几个公司都发行了交互开发环境。1996 年中期，微软公司发行了它的 Java 交互开发环境——Microsoft Visual J++ 1.0。

Microsoft Visual J++完全符合 Sun 公司的 Java 语言规范。它是一个开发 Java 小程序和独立程序的编程平台，包括 Java 编译器、编辑器、调试器和大量在线文件。

近年来，随着网络的蓬勃发展，Java 语言也受到越来越多的网络程序设计人员的青睐。人们称其为 21 世纪的语言，这恐怕是 Java 的开发者们所始料不及的。

虽然 Java 是以 C++为模型生成的，但它的开发者们赋予它许多新特点：更简单的语法；更好的可靠性和安全性；在不同平台间的可移植性更强。因此，尽管 Java 保留了许多 C++的风格，但它实际上是一种与 C++很不一样的语言。Java 和 C++之间的主要不同点如下：

- 由于 Java 是为开发网络应用程序而设计的，它能够支持高效的通讯和发布；而 C++不具备这些功能。
- Java 编译器生成的是独立于平台的虚拟代码(bytecode)文件，这种文件的执行需要 Java 虚拟机；而 C++编译器生成的是可执行机器代码文件。
- Java 既可以生成与传统可执行文件差不多的独立应用程序，也可以生成在网上实时运行的小程序；而 C++只能生成独立应用程序或组件。

· 在运行时，Java 程序动态地链接所有需要的类；而 C++程序只能在编译时静态地链接所需代码。

· Java 只支持对内存的引用；而 C++不仅支持引用，还支持指针和直接内存访问(DMA)。

· Java 支持自动收集垃圾的动态内存分配；C++也支持动态内存分配，但开发者必须自己为每一个类分配内存。

由于有以上这些不同，在创建高级应用程序时，Java 可以部分代替 C++。但是，C++的灵活性、高效率和成熟使其仍然是一种常用的程序开发语言。

1.1.2 小程序和独立程序

Java 程序可以是小程序也可以是独立的应用程序，两类 Java 程序都具有以下相同特点：

· 两类程序都由一个或多个以.CLASS 为后缀的文件组成。

· 两类程序都需要用户系统安装 Java 虚拟机(JVM)。Java 虚拟机能够载入并翻译 Java 程序，并且可以提供 Java 内核包的实现。

当然，这两类程序也有不同点，它们的主要不同在于：

· Java 小程序可以被嵌入 HTML 网页内，从而可以在网络上发布，当网页被浏览时它们可以在浏览器中运行；而 Java 应用程序却不支持网页嵌入和下载。

· Java 小程序只能在与 Java 兼容的容器中运行，例如现代的网页浏览器；而 Java 应用程序却没有这个限制。

· Java 小程序的运行受到严格的安全限制，例如它不能访问用户计算机上的文件和系统服务；而 Java 应用程序没有固有的安全限制。关于安全性问题，稍后我们会详细讨论。

实际上，我们很容易生成既可以是 Java 小程序又可以是 Java 独立应用程序的“混血 Java 程序”。我们在下一章介绍 Visual J++的 Applet Wizard 时再讨论具体的实现方法。下面，我们将详细说明 Java 小程序和 Java 独立应用程序之间的异同点。

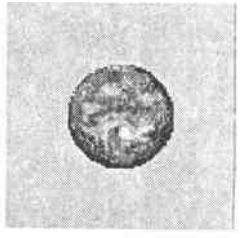


图 1-1 小程序的运行结果

从用户的角度来说，Java 小程序和 Java 独立程序在运行时有很明显的差别。图 1-1 是在网页浏览器上下载一个 Java 小程序运行的情况，而图 1-2 是在本地机上运行 Java 独立程序时的情况。它们之间的差异是一目了然的。

Java 小程序和 Java 独立程序在代码的编写上也有很大差别。我们在前面提到过：Java 是面向对象的程序设计语言。这意味着它的主要组成模块是类或者说是用户自定义的数据类型。在 Java 代码中实现这些类的方式就决定了开发者设计的是 Java 小程序还是 Java 独立程序。对源代码来说，Java 小程序和独立程序之间的基本区别如下：

· 一个 Java 小程序必须定义一个 Applet 类的子类，一个独立程序也可以定义一个 Applet 类的子类，但这不是必需的。下面的代码演示了如何扩展 Applet 类：

```
//从 java.applet 类库中引入类
import java.applet.*;
//从 Applet 类中扩展一个名为 FirstApplet 的子类
public class FirstApplet extends Applet {
```

```
    //Java 源程序
}
```

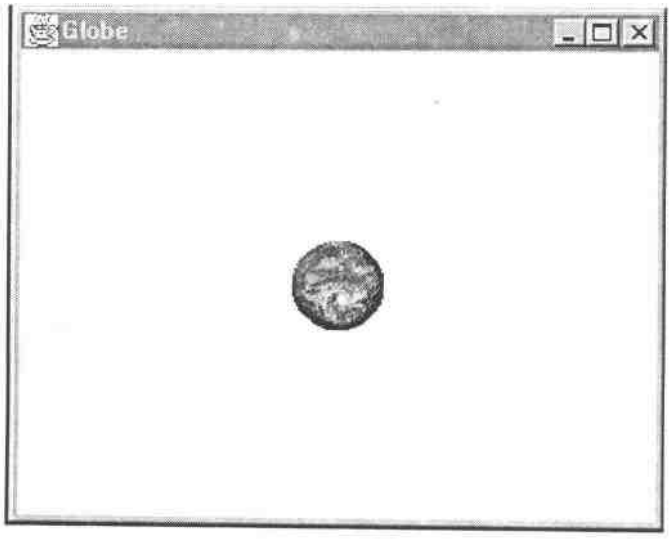


图 1-2 独立程序的运行结果

• 一个 Java 独立程序必须在一个类中定义一个 main 方法，该方法代表了该独立程序的入口点；而一个 Java 小程序并不定义 main 方法，它的执行是由 Applet 类定义的多个方法控制的。下面的代码演示了怎样在一个 Java 独立程序中定义 main 方法：

```
//定义一个叫作 FirstApplication 的类
public class FirstApplication {
//定义 main 方法
public static void main(String Args[]) {
    // Java 源程序
    }
}
```

1.1.3 开发和发行 Java 程序

Java 开发平台及相关支持技术的更新速度是非常快的。商业 Java 程序在 Java 语言中的开发和部署很好地说明了 Java 的快速成熟。这主要是由以下的因素造成的：

• 降低了开发时间和花费。Java 语言一开始就是面向对象的。这样，开发者一旦掌握了它面向对象的特点，开发工作就能够比常规的循序渐进方法进行得更有效。而且，Java 语言在快速生成高级和强壮的软件方面比 C++有更大的优越性。

• 商业类库。Java 语言有大量的内置类。这些类是开发者编写 Java 程序的基础。渐渐地，商业类库的出现使开发者的工作有一个很高的起点。例如，如果开发者能够使用提供用户界面功能(如各种窗口控制)的商业类库，他编写 Java 小程序的工作将简单得多。

• 基于组件的应用程序。很多软件商认为 Java 是新一代可繁殖程序的基础，这些可繁殖程序是基于对象或组件编程模型的。在这个模型中，应用程序是由一些可下载的组件组成的，而不是一个单一的闭门造车的产物。

在使用一个基于组件的应用程序时，用户并不需要将整个程序安装在他的本地系统上。相反，他可以只从浏览器上下载他所需要的那部分功能。例如，在创建一个字处理程序时，用户从第一个小程序中得到主要的字处理功能，而图形功能可以在需要时才从网上的第二个小程序中下载下来。

这种程序开发的组件模型和传统的软件开发方式有很大的不同，现在还只处于初级阶段。如果程序开发员采用这种开发模型，他将面对很大的挑战。

Java 语言和 Java 实时系统允许开发者在万维网上发行 Java 小程序。小程序一般都很小，在浏览器提出请求时可被下载并执行。Java 小程序的加入给万维网增色不少。例如，Java 小程序可以提供高级控制并对用户产生的事件作出反应。Java 语言提供丰富的事件模型，它比 HTML 具有更强大的功能。

Java 还允许开发者编写可以脱离网页浏览器运行的有特色的 Java 独立程序。在这方面，Java 有可能代替 C 语言或其他的语言，这是因为 Java 具有以下特点：

- 使用 Java 强大的网络功能可以改善许多本地基于网络的客户/服务器程序的性能。
- Java 独立程序可以在一个组织的本地网上开发、安装和运行，而 Java 小程序有那么多的安全限制。另外，由于 Java 有许多内置的安全特性，它可以为用户系统提供额外的保护。

1.1.4 Java 虚拟机

我们说过，要运行 Java 程序必须在用户系统上安装 Java 虚拟机，而什么是 Java 虚拟机呢？Java 虚拟机实际上是一个应用程序，这个应用程序包括一个 Java 虚拟码的翻译器，这个翻译器可以执行虚拟码指令。Java 虚拟机把虚拟码翻译成本地代码（也叫机器码）。可以看出，Java 程序的执行速度肯定没有本地应用程序快。这也是目前 Java 面临的问题之一。可以说，这是为实现系统可移植性而付出的代价。我们可以打个比方来说明 Java 虚拟机和虚拟码之间的关系：如果说虚拟码是汇编代码的话，那么 Java 虚拟机就相当于中央处理器（CPU）。

还有一个问题就是：如何安装 Java 虚拟机呢？事实上，大部分开发者不需要考虑这个问题，因为和 Java 兼容的网络浏览器都内置 Java 虚拟机，只要安装了一个和 Java 兼容的浏览器就自动安装了 Java 虚拟机。例如，只要我们安装了微软的 Internet Explorer 就自动安装了 Java 虚拟机。我们可以从下面的网址免费得到微软的 Internet Explorer：

http://www.microsoft.com/ie

从长远的角度看，最新推出的操作系统中都会把 Java 虚拟机集成到系统中去，所以我们根本不用为安装 Java 虚拟机担心。

1.1.5 Java 小程序与 HTML

当我们在网上运行 Java 小程序时，我们实际上是在网页的 HTML 代码中得到 Java 小程序的。HTML 允许向 Java 小程序传送参数，例如，我们可以通过设定参数来规定一个动画的移动速度。

在浏览器中调用 Java 小程序的传统方法是使用 HTML 的 APPLET 标志。下面的例子演示了如何使用该标志调用 Java 小程序：

```
<APPLET
    CODE=MyApplet.class
```

```
    ID=FirstApplication
    WIDTH=320
    HEIGHT=240 >
</APPLET>
```

其中参数的意义如下：

- CODE 指定要下载的 Java 虚拟码文件(.class)；
- CODEBASE 用 URL 指定要下载的 Java 文件所在的目录；
- ID 指定小程序的名字，在 HTML 文档中该参数可以指代 Java 小程序；
- WIDTH 指定小程序窗口的宽度；
- HEIGHT 指定小程序窗口的高度。

第二个参数 CODEBASE 是可以缺省的，如果 Java 类文件和 HTML 文件没有放在同一个目录下，我们就必须用该参数指定相应的目录。下面的例子演示了如何用 APPLET 标志调用一个不在同一目录下的 Java 小程序：

```
<APPLET
    CODE=MyApplet.class
    CODEBASE="http://www.acme.com/JavaClasses"
    ID=FirstApplication
    WIDTH=320
    HEIGHT=240 >
</APPLET>
```

前面曾提到，我们可以在 HTML 中将参数传递给 Java 小程序。我们可以用两种方法实现这项功能：一种是用 PARAM 标志静态地传递这些参数。在下面的例子中，我们传递的参数是 Message，它的值被设定为"Hello!"。

```
<APPLET
    CODE=FirstApplet.class
    ID=FirstApplet
    WIDTH=320
    HEIGHT=240 >
<PARAM name=Message value="Hello!">
</APPLET>
```

另一种方法是用脚本文件调用 Java 小程序，第八章中我们会详细地介绍这种方法。

1.2 Java 体系结构

我们已经知道，Java 语言及其实时环境是为方便网络计算而设计的。最初的 Java 开发小组致力于使 Java 程序具有很大的灵活性和很高的可靠性，从而可以通过网络发行并在任一个操作系统上运行。这样，Java 的体系结构必须有自己的特点。本节的内容就是介绍 Java 语言的体系结构和 Java 程序在网络上运行的方式。

1.2.1 Java 体系结构简介

我们知道，Java 语言的设计目的就是要使其程序能在各种各样的计算机上运行，因此，

Java 的体系结构和大部分其他的现代语言不同。

Java 实时系统先将源文件以 ASCII 码文件格式存储，然后这些源文件被编译成 Java 虚拟码(bytecode)文件。Java 虚拟码是一种标准化的、与机器无关的低级语言。最后，用户系统上的 Java 虚拟机将这些虚拟码文件翻译成在本地机上可执行的机器码。举一个例子，当我们从网络上下载一个 Java 小程序时，网页浏览器中的 Java 虚拟机就将 Java 虚拟码翻译成可执行的机器码。由此可见，在网上发行的 Java 程序是和机器无关的虚拟码，这就是 Java 实现独立于平台的方法。我们已经说过，这样做的代价是 Java 程序的执行速度变慢了，因为比其他现代语言程序的执行过程多了一个翻译虚拟码的中间过程，而且翻译器的翻译速度比本地机器码的执行速度要慢。

为了提高 Java 程序特别是大型 Java 程序的执行速度而不影响它的可移植性，许多 Java 零售商在他们的浏览器中或编译器中安装了即时编译器(just-in-time compiler)。这个可选器件可加入 Java 体系结构中起代替或补充 Java 翻译器的作用。我们知道，Java 翻译器是 Java 虚拟机的一部分，于是 Java 即时翻译器也可以看做是 Java 虚拟机的一部分。Java 即时编译器的工作方式可以从它的名字中推断出来：它不是一行一行地翻译虚拟码指令，而是一次将整个 Java 方法翻译成相应的本地码；另外，这种翻译工作不是在编译时完成的而是边运行边进行的，这一点与 C 和 C++语言差不多。当然，使用这种方法的代价是本地代码的执行时间加长了，因此，对一些 Java 小程序来说，使用 Java 即时编译器反而会降低整体性能。一般来说，如果 Java 程序中的重复操作很多或计算比重很大，这时我们使用 Java 即时编译器能够大幅度地提高 Java 程序的执行速度。

1.2.2 Java 实时系统

通过上面的介绍，我们知道了 Java 实时系统主要是由以下几个组件构成的：

· 将 ASCII 码文件转换为 Java 虚拟码的 Java 源代码编译器

我们知道，如果要编写一个 Java 小程序或 Java 独立程序，就必须先编写 Java 源程序。Java 源程序通常都是以.java为后缀的 ASCII 文件。一般地，我们都要使用 Visual J++类库或其他商业类库来完成和 Java 源代码的编写工作。一个或多个和 Java 源代码就可以组成一个和 Java 项目。

当我们完成了一个 Java 项目后，我们必须运行一个 Java 源代码编译器，例如 Visual J++。Java 源代码编译器为每一个和 Java 源文件生成一个 Java 虚拟码文件，这种文件都是以.class为后缀的。在 Java 虚拟码文件中就含有可以被下载并在本地机上执行的指令。

在完成虚拟码文件的转换过程之后，我们既可以将它们放在一个网络服务器的目录中，又可以把它们当作独立程序在自己的系统上运行。但是，无论我们把它们作为网上的 Java 小程序还是 Java 独立程序来运行，它们都是被下载到一个 Java 虚拟机中并翻译执行的。

· 将 Java 虚拟码转换为本地机器码的 Java 虚拟机

关于 Java 虚拟机我们已作了很多的介绍，现在只是简单介绍一下其具体的工作情况。在 Java 虚拟机转换 Java 虚拟码时，它一般要做以下一些工作：

1. 检查 Java 虚拟码文件的合法性，包括检查它们的格式和安全性。这项功能也被称作虚拟码识别器(bytecode verifier)；

2. 为 Java 虚拟码文件分配内存。这项功能也被称作类载入器(class loader)；

3. 翻译 Java 虚拟码指令；

4. 完成有关标准类库的调用，这些类库经常是作为 Java 虚拟机的一部分安装的。

· Java 即时编译器

我们只能在支持 Java 即时编译器的浏览器(或者 Java 虚拟机)中运行 Java 即时编译器，比如微软公司的 Internet Explorer 就是这样的一个网络浏览器。我们可以按以下的步骤使 Internet Explorer 支持 Java 即时编译器：

1. 在 view 菜单中单击 Options 菜单项；

2. 在 Options 对话框中单击 Advanced 按钮；

3. 选择 Enable Java JIT Compiler。

1.2.3 网络浏览器

当我们在网络浏览器中运行 Java 小程序时，主要的工作是由 Java 虚拟机完成的，而网络浏览器不过是为 Java 小程序的输出提供一个矩形边框。这意味着 Java 小程序可以在任何支持 HTML 的 APPLET 标志和安装了 Java 虚拟机的网络浏览器中运行，而且运行结果是完全一样的。如果用户的浏览器不支持 HTML 的 APPLET 标志，浏览器只是忽略过这个标志而对整个网页的载入没有影响。由于小程序窗口的高度和宽度参数也连带着被忽略了，浏览器相当于是压缩了网页，小程序就像完全不存在一样。

如果网页中的小程序提供很重要的功能，忽略它也许会造成严重的后果。这时，我们有两种方式可以解决这个问题：

1. 最简单的解决办法就是显示一个警告信息，说明网页中嵌入了 Java 小程序而用户浏览器无法执行这个小程序，其他的事情由用户自己去解决。

2. 第二种方法就是提供替代的 HTML 语句，这些语句完成跟 Java 小程序一样的工作。支持 Java 的浏览器会忽略这些 HTML 语句，不支持 Java 的浏览器会忽略 APPLET 标志而执行这些语句。例如，在下面的代码中，替代的 HTML 语句在显示了一个警告信息之后完成了和 Java 小程序一样的工作。

```
<APPLET
  CODE=outline.class
  HEIGHT=150
  WIDTH=200>
You're missing a Java outline applet.
<UL>
<LI><A HREF="default.htm">Internal Training Home Page</A></LI>
<LI><A HREF="States.htm">Location Courses are Offered</A></LI>
</UL>
</APPLET>
```

1.3 Java 安全性

除了可移植性之外，Java 语言和 Java 实时系统的另一个主要目标就是保证用户系统的安全性。如果 Java 程序的安全性得不到保障，用户是不敢从网上下载 Java 程序的，而 Java 作为

网络语言也就失去了生命力。实际上，Java 安全性和可移植性是矛盾的。我们很容易理解这个结论，因为如果用户能够从网上透明并且自由地下载 Java 程序，这本身就不可避免的会带来很多安全性问题，而如果要保护用户的本地系统，最好是不让用户透明地下载 Java 程序。在 Java 语言的实际设计中，这两方面的要求都得到了最大程度的满足，这也是 Java 成功的原因之一。

1.3.1 安全级别

前面我们介绍了 Java 的体系结构，要保证 Java 程序的安全性。从 Java 源程序的编写到它的最终运行，我们都要考虑它的安全性问题。根据这个过程，我们可以分四个级别考虑其安全性，分别介绍如下：

1. 语言和编译器

Java 语言和编译器是 Java 安全性的第一级别。虽然在很多特点上 Java 和 C++一致，但 Java 语言没有 C++那么多的安全问题。例如，Java 不支持指针操作，虽然这会使其缺少一定的灵活性，但可以防止某些程序设计者利用指针绕过系统的安全屏障。

2. 虚拟码识别器

虚拟码识别器是 Java 虚拟机的一个功能，它可以检查虚拟码文件是否侵犯了系统的安全性，其具体的功能如下：

- 确保虚拟码文件没有伪造指针；
- 确保虚拟码文件没有侵犯访问限制；
- 确保对象是被正确使用（比如确保 InputStream 对象是被用作输入流而不是其他的任何东西）；
- 确保方法是用正确类型的参数调用的；
- 确保虚拟码文件不会造成堆栈溢出；
- 确保虚拟码文件没有不安全的指令。

3. 类载入器

类载入器的功能是为每一个类分配内存，以及确保代码没有试图取代一个内置类。也就是说，Java 不允许程序员编写自己的类（如 String 类）来代替内置类执行。

4. 文件系统保护和网络访问

如果 Java 代码通过了以上三个级别的检查，那么现在必须考虑它对用户文件系统的安全性的影响。实际上，Java 根本就不允许 Java 小程序对本地的资源进行访问，而文件作为本地资源的一种也是不能被 Java 小程序访问的。人们把这称为 Java 安全模型的沙盒元素（sandbox element）。虽然沙盒使编写 Java 小程序的工作难而且笨拙了一些，但它保证了用户机上没有文件会被创建、删除或者崩溃。

然而，我们并不是说用户不能存储文件，实际上用户可以存储文件，只不过这些文件被存在原来的服务器上而不是用户的计算机上。例如，一个用 Java 编写的字处理程序不能将文件存在用户的硬盘上，但可以存在服务器的硬盘上。

虽然前三个安全检查阶段对用户和 Java 程序开发者都是透明的，第四个阶段（沙盒模型）却不是这样，用户和 Java 程序开发者都必须清楚沙盒模型禁止 Java 小程序做什么。在网络浏览器和其他容器中，Java 小程序不能做以下的事情：