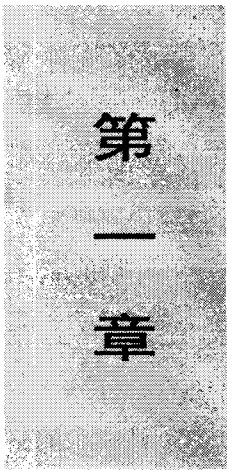


ActiveX与Visual J++概述

随着 Internet 和 WWW（World Wide Web）的迅猛发展，Web 网页由静态转向动态。用于开发动态网页的软件工具，继 Sun 公司 Java 语言和 Netscape 公司 Navigator 2.0 之后，Microsoft 又推出了 AxtiveX。在 Internet 市场上，虽然 Microsoft 发布 AxtiveX 很晚，但正全力以赴将 AxtiveX 技术树为创建 Web 控件的标准。但是，就 Microsoft 的开发体系而言，现在可以见到的允许用户创建或使用 AxtiveX 控件的 Java 开发系统有 Visual J++。Visual J++也是 Microsoft 公司开发出的可视化的 Java 开发环境。Web 开发者可以将 AxtiveX 控件嵌入到 Web 网页之中，并使用 VBScript 来驱动它们。而 Visual J++中的 AxtiveX 运行类又可为开发者提供新的途径。

AxtiveX 技术和 Visual J++是 Microsoft 向 Internet 世界发起冲击的工具。

1.1 ActiveX简介

简单地说，AxtiveX 是一种允许程序（即 AxtiveX 控件）通过网络与其它程序交互的体系结构。它与 Java 不同。Java 是一种程序设计语言和一套虚拟机规范。AxtiveX 体系结构则使用 Microsoft 的构件对象模型（COM）和分布式 COM（DCOM）标准，使不同的应用程序在本地交互或支持网络通信。AxtiveX 的一个重要功能是能在页面中增加客户机/服务器能力。AxtiveX 控件允许 Web 站点的访问者执行复杂的操作，例如从数据库或其它服务器甚至其它 Web 站点上的应用程序接受数据。这就是 Microsoft 所宣传的 AxtiveX“激活”Web 页面。

在详述 AxtiveX 之前，我们先来简单回顾一下开发 Web 网页的发展历程。

Internet、超文本和多媒体三项技术的结合使之诞生了 Web。由于 Web 的运行机制是建立在客户/服务器体系结构之上的，因此 Web 有服务器程序和客户机程序之分。在 Internet 网上以 Web 为基本信息查询手段建立起来的环球网 WWW，其最基本的传输单位是 Web 网页。早期开发 Web 服务程序，用户只能通过超文本标识语言 HTML 实现。HTML 允许在 Web 页面上显示正文和一些类型的图形，还包含有连接各个 Web 页面的链（Link）。允许 Web 页面是位于一台计算机上或分布在世界各地。随着版本的不断演化，HTML 加进了表格、框架等许多新的特性。尽管如此，HTML 基本上还是只能处理静态 Web 页面。处理的内容主要是：格式化并显示 Web 页面内容；等待用户单击 Web 页面某一处；根据用户单击的内容取出一个新的 Web 页面显示。

虽然 HTML 提供了比较丰富的信息操作手段，但无法满足生成动态 Web 页面的要求，例如在 Web 页面上访问数据库、制作动画、安排目录项等。为了实现这些要求，需要使用 CGI（通用网关接口：Common Gateway Interface）技术。

CGI 是 Web 服务器在调用外部程序时的参数规范协议，于 1993 年发布。CGI 技术允许 Web 服务器运行外部程序，它规定了一组标准的环境变量和参数格式。CGI 所提供的机制扩充了 HTML 的功能。利用 CGI 机制，Web 页面设计者可以编写一些驻留在 Web 服务器端的自定义程序，通过这些自定义程序与 Web 页面的交互来完成各种复杂的工作。

CGI 的重要意义在于它在外部程序和 Web 页面之间提供了一种连接，可以实现动态创建 Web 页面，处理 HTML 表单输入，例如用户可在页面上键入数据，然后将这些数据发送到程序。因而，通过 CGI 可以实现：

- 建立 Web 服务器与数据库之间的接口，访问数据库或向数据库中增加信息。
- 联机目录排放。
- 跟踪在 Web 页面上发现的搜索引擎。例如 Yahoo 就是使用了 CGI。
- 生成映射图像。映射图像是一种可被单击的图像，其图像的不同区域对应不同的 HTML 链。
- 向浏览器发送一系列图像，通过图像替换形成简单动画。
- 根据命令动态生成自定义 HTML 页面，例如网络页面访问者在许多页面上所见到的“你是第 2515 个访问者”信息。

CGI 脚本是按 CGI 规范编写的在服务器端执行的程序，它负责处理来自于服务器请求的一个动态响应所必需的所有任务。由于 Internet 最初是从 Unix 系统上发展起来的，很多推进因素来自 Unix，CGI 也一样。因此，大多数 CGI 程序是用一种基于 Unix 的 Perl 语言编写的。今天，大多数编程语言均可支持 CGI 规范，使 CGI 脚本的编程工具更加丰富，常见的如 Visual Basic、VBScript、Visual C++、Borland C++、Visual J++、JavaScript、ActiveX、Foxpro 等都可在以 Web 为基础的综合环境中开发。

CGI 应用程序为静态内容增加了一些动态性。这种动态性是依赖服务器端的计算能力，CGI 仅在用户请求网页时生成特定的网页。一旦网页被装入浏览器，其网页内容本身仍然遵循 HTML 2.0 的静态内容模型，既不变化也没有动感，是静态的。

但是，由于 CGI 可以支持各种不同的应用，例如发送电子邮件、查询数据库等等。因此 CGI 曾广泛流行。

CGI 也具有下述一些缺点：

● 运行效率较低。CGI 的操作过程是将用户端的动作传送到服务器端进行解释处理，之后再将处理的结果传回到用户端。客户机和服务器之间的这种数据传输方式，消耗时间，降低 Web 页面的交互响应的效率。

● 负载过重。由于 CGI 程序仅运行在服务器端，它要完全依赖于 Web 服务器，这样程序运行的速度和能力即受到服务器能力的限制。随着 Internet 的迅速发展，大多数服务器难以适应这种膨胀的需求。

● 安全性问题。在 CGI 出现的前几年间，可以通过比较简单的技术手段进入 Web 服务器，执行命令、寻找敏感数据，它意味着 CGI 在安全性方面仍具有一定风险。

● CGI 对声音、图像、动画等多媒体的支持不很完善。

● 编程复杂，只有专业编程人员才能使用 CGI 进行程序设计。

CGI 无法支持 Web 上的所有应用需求，又过于复杂，因此 Java 应运而生。

其实，开发 Java 的初衷并不是用于 Web 的。1991 年 4 月，Sun 公司的一个研究开发小组，针对消费类电子产品市场，着手开发一种与硬件平台无关、建立在标准基础之上的编程语言。开始将其命名为 Oak，后来改为 Java。由于商业因素和市场竞争，虽然 Java 在消费类电子产品应用中没有获得成功，但却歪打正着。当图形用户界面的 WWW 在 1994 年如火如荼大发展时，Sun 发现 Internet 和 WWW 所蕴藏的巨大商机，将 Java 定位于 Web 浏览器的应用之上，于 1995 年 5 月正式发布了 Java 开发环境。与此同时，最流行的 WWW 浏览器生产者 Netscape 公司宣布在其产品中支持 Java。随即，IBM、Novell、Oracle、SGI、Borland 等许多著名公司相继购买 Java 使用许可证，将 Java 推到了开发 Web 应用的至尊地位。

Java 是一种编程语言，可为 Web 页面设计提供更为强大和灵活的方法。Java 采用虚拟机技术，很容易地实现可移植性和硬件平台无关性。另外，它所具有的面向对象、安全性、健壮性、动态性、易用性、高性能、多线程等特点，从根本上改变了 Internet 和 WWW 的性质。显而易见，Java 不仅克服了 HTML 和 CGI 的许多局限，而且还能提供 HTML 和 CGI 所不具备的功能。Java 是一种完善的编程语言，运用 Java，程序设计员可以实现：

● 访问数据库或其它信息源。

● 产生动态逼真的动画。

● 实时显示、接受用户输入表格，弹出对话框和提示框。

虽然 Java 功能强大，但它主要还是为了完成复杂任务而设计的。同时，它仍然是一种编程语言，与其它程序设计语言一样，需要有经验的开发者才能使用。例如，Java 主要由应用程序 Application 和小应用程序片段 Applet 构成。运行之前用户必须对 Applet 进行编译。每次修改程序后都需要重新执行编译，每次编译都要花费时间。另外，现在也没有一个好的 Java 编辑/开发包为用户提供有效简化的 Java 程序创建工作。

在此之前，Microsoft 在 Internet 和 Web 网页开发方面几乎无所作为。但是当他发现 Internet 的重要性和 Web 的适应性后，便迅速掉转船头，开足马力冲向 Web。

1995 年 12 月 7 日，Microsoft 宣布从 Sun 获得 Java 使用许可，很快便发布了用于 Web 的客户端浏览器 Explores。

随即，Microsoft 将 Internet 的一些特征增加到它以前开发的面向对象应用的成果 OLE（对象连接与嵌入标准）之中，并且将其命名为 ActiveX。这是一个绝妙的捷径！这种做法

虽然使 Microsoft 背离了 Sun 和 Netscape 已经开辟的 Java 道路，但实际上却未多做任何艰苦复杂的开发工作就达到了工业界的领先地位。

ActiveX 是一种体系结构，实际上，ActiveX 是从 OLE 演变而来，ActiveX 对象就是带有可在 WWW 上运行的附加功能的 OLE 对象。ActiveX 与其祖先 OLE 共享许多特色，允许程序进行交互，将其嵌入到文挡之中。ActiveX 除了增强 OLE 使其可用于 Web 之上外，最重要的增强是 ActiveX 控件。

1.2 ActiveX控件

由于 OLE 投放市场时间较长，开发者们已经编写了大量 OLE 对象，这些对象现在可以用作 ActiveX 控件。如此说来，任何第三方 OLE 对象生产厂商也是 ActiveX 控件的厂商，据 Microsoft 声称这样的生产厂商已经超过 1000 家。许多软件公司大量销售 OLE 对象库，而这些对象是可用于构造程序的预制部件。现有的大量 ActiveX 控件可以使开发者以最少的时间轻松地构造程序，这正是 ActiveX 控件的最大优势。ActiveX 设计的初衷是支持 ActiveX 控件在 Web 上运行，但是，目前 Microsoft 正全力以赴地将 ActiveX 控件树为创建 Web 控件的标准。在 Microsoft 与 Sun 的这场竞争中更重要的是已存在的市场，流行的 Java 虽然功能强大而且正在增长，但现在还没有这个市场拥有量。

ActiveX 也对 OLE 对象进行了一些修改，ActiveX 控件是运行于 Web 之中的内置程序。开发者根据 ActiveX 规范使用程序设计语言创建的 ActiveX 控件，是程序的自包含片段或独立构件。开发者可以在其它程序，甚至其它语言编写的程序中重用这些 ActiveX 控件。例如，可将一个 Visual C++编写的控件插入到 Visual Basic 编写的程序中。这种特性与 Java Applet 类似，它能够制作光彩夺目的 Web 动画、实时显示常用表格、容易访问数据库等等。相比之下，ActiveX 控件（如图 1.1 所示）比 Java 控件的功能更强，它能够在客户机上存取文件，运行速度更快。

虽然从 OLE 对象转到 ActiveX 控件仅有少量改变，但它却是 Microsoft 为了控制 Internet 战略的最重要的部分。Microsoft 试图以 ActiveX 控件替代 Java 控件，因此 Microsoft 尽力赋予 ActiveX 比 Java 更多的优点，借助 OLE 优势，Microsoft 确实具有这种潜力。例如，能够创建 OLE 对象的任何工具也能够创建 ActiveX 控件，开发者不必学习任何新的编程技巧，可以使用 Borland Delphi、Visual C++、Visual Basic 等工具。

Microsoft 的另一个优势是 ActiveX 控件与桌面软件的兼容性，任何可以使用 OLE 的程序，如字处理、电子表格、数据库等等，ActiveX 控件也能使用。比较而言，Java Applet 只能在浏览器中或作为独立应用程序运行。

拥有大量可重用对象使 ActiveX 可能成为开发通用客户机/服务器程序的有力工具。它不仅提供了现成的控件库，也允许开发者重用自己的控件。

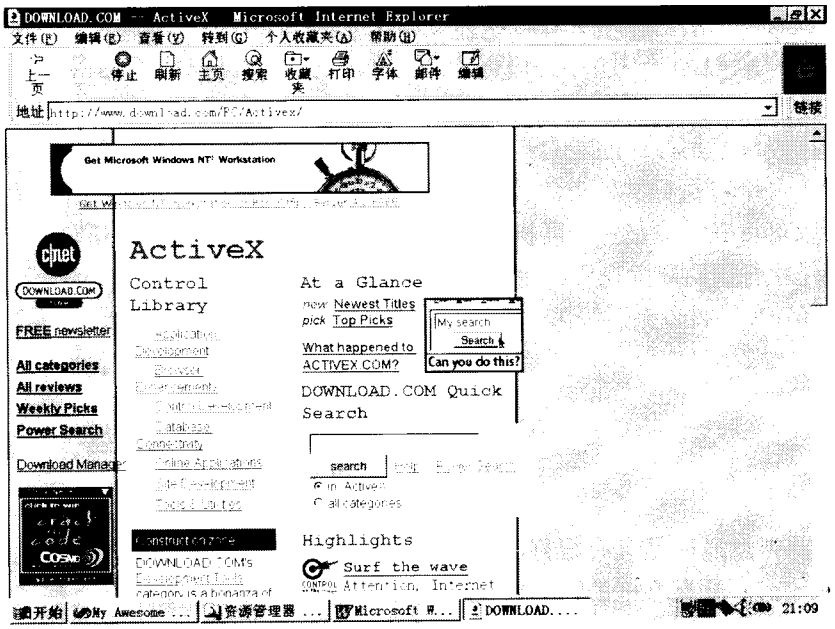


图1.1 在Internet上的ActiveX控件

ActiveX 也存在下述一些问题：

● 跨平台问题

对于 ActiveX 控件来说，主要的缺点是与 Windows 平台的紧耦合性。开发者选择 ActiveX，则将被约束在 Windows 之上。如果在一个完全由 Windows 构成的 Internet 上运行或访问 Web，ActiveX 控件会很好的支持。但事实上 Internet Web 网页的访问者可能是来自多种不同的操作系统。开发者不能期望只编写一个控件供所有的 Web 站点访问者使用，必须对在不同平台上运行的 ActiveX 控件重新进行编译。目前为止，ActiveX 只能在 Windows 系列平台（Windows 95、Windows NT、Windows 3.X）上运行。Microsoft 已经着手改变这种状况，准备向其他公司提供，以便用于 Unix 和 Apple 的 Macintosh 系统。

比较而言，Java 不受硬件限制，具有更广泛的跨平台能力。由于 Java Applet 是运行在 Java 虚拟机上，而 Java 虚拟机运行于操作系统顶层，所以 Java Applet 与操作系统完全无关，只需要在自己的 Web 页面编写一个 Java Applet 就可以在各种平台上运行。

但是，也有另外一种提法，认为不必过多考虑运行平台问题。因为，不仅 Windows 已经成为全球性的桌面标准，而且，随着企业内部网 Intranet 将连接到 Internet 网上的发展趋势，跨平台要求可能逐渐成为了非关键因素。企业内部常常是将应用统一到单一平台之上，也就是说，在企业内部网 Intranet 上，跨平台问题仍在讨论之中。

● 安全性问题

ActiveX 与 Java 的安全性形式不同。

Java 的安全模型称为“沙盒”（sandboxing），它是将所有可执行代码限制在一个受限区域内。Java 取消了内存指针，程序无法访问不该访问的地方。Java 编译器也不处理内存布局，程序员无法从类的定义中推断出运行时刻的实际内存布局。实际上，Java 已经抛弃了开发者直接进行内存分配及布局的能力。Java 的另一个安全性措施是在运行系统中增加了字节代码验证器，它在运行之前对字节代码进行验证，以确保代码被有意无意改变。Java

的安全机制可以有效地防止网络病毒、黑客侵入及下载程序对本地文件系统的破坏。

ActiveX 的安全模型采用的是身份验证技术，它由实现代码签名的 Microsoft 程序 Authenticode 提供。这种安全性的功效是依赖于核实 ActiveX 控件开发者身份的能力，它只能赋予用户根据授权鉴别控件的能力和责任，而不能保护控件客户不受恶意及不当控件的侵犯。例如，对于一个可执行文件类的控件，它的运行与.EXE 文件相似，由于是已编译的二进制代码，所以可将其看成是一个联机命令系统。这样，开发者就能够开发出一个 ActiveX 控件，其功能是执行硬盘删除操作！因此，开发者必须确保 ActiveX 控件的来源，不能使用来路不明的 ActiveX 控件。

1.3 ActiveX与Java的集成

Java 语言的得意之处是具有很好的可移植性。其实，跨平台能力本身就提供了一种良好的可移植性。Java 具有严格的语言定义，例如，它所提供的基本数据类型其长度及表示方法都是确定的，这样就不必依赖某台机器及它所使用的编译器。为了使 Java 的应用程序不依赖于具体系统，Java 提供了一个用于访问底层操作系统功能的可扩展类库，程序使用这些库可以保证在支持 Java 的各种平台上运行。Java 还具有内部的异常管理，可以存取类库的源码。

开发者使用 Java 语言编程时，不必关心释放内存问题，因为 Java 具有内存碎片收集功能。

ActiveX 与 Java 的关键区别是，ActiveX 是一种体系结构，Java 是一种语言。Java 帮助程序员编写软件构件，ActiveX 允许这些构件相互交互，可以集成两项技术协同工作，而不是相互角逐。

ActiveX 和 Java 的共生关系所解决的问题是任何独立的单项技术所不能确定的，例如：

- 容易学习 —— 开发者能够使用C++、Visual Basic、甚至Java等任何语言编写ActiveX构件。如果开发者既想创建 ActiveX 控件，又不愿放弃 Java 的豪华，可以二者兼顾。开发者不用从头学习任何新的编程语言就能够编写 ActiveX 控件。
- 减少开发的重复过程 —— Java 类库虽然功能强大，但不能随意管理，而许多场合用户要求使用先前编写或购买的源码。Java 虽然提供一个本地码接口，它是在 Java 程序中使用C语言，但文档缺少，使用困难，且破坏了Java的可移植性。好的解决办法是在Java Applet之中封装 ActiveX 控件。利用 ActiveX 功能，Java 开发者可从现有的大量商业化 ActiveX 控件中寻找、选择所需要功能特点的控件，也能够将以 C、C++、Visual Basic 等语言编写的现有的代码转换成配套的 ActiveX 控件，然后将控件插入到 Applet 中去。使用 ActiveX，大部分控件代码不必从零开始编写。
- 与其它程序通信 —— 因为 Java 的设计不是用于 OLE 的，所以它不能与当前流行的一些软件交互通信，例如 Microsoft EXCEL、Borland Paradox 等等。但是，如果借助 ActiveX，Java 就能够调用 OLE 接口，从而实现与支持 OLE 的软件通信，使 Applet 不再孤立，也具有了存取第三方厂家应用的宽广能力。

1.4 Visual J++简介

Visual J++是由 Microsoft 公司开发的第一个商业化、可视化的 Java 开发环境，其用户界面采用了 Microsoft 统一的 Developer Studio，与 Visual C++十分相似。目前，Visual J++是在 ActiveX 和 Java 二者之间提供紧密集成的唯一工具。

Visual J++是专门用于创建 Java Applet 和应用的开发环境，是 Microsoft Visual C++的同胞兄弟。Visual J++运行于 Windows 95 或 Windows NT 4.0 平台之上，它由以下几个部分组成：

- 文本编辑器；
- 资源编辑器；
- 帮助用户创建 Applet 和 ActiveX 控件的向导（Wizard）；
- 方便对 Java 对象理解的类浏览器；
- 可视的调试器；
- 高速源代码编译器；
- 可提高 Applet 执行速度的高级 JIT 编译器；
- Java 类层次视图；
- 工程文件管理；
- 联机帮助和演示。

Visual J++和 Java 完全兼容，在保持 Java 原有优点的基础上，又进行了下述几方面的改善和优化：

- Visual J++的源代码编译器非常快，每秒钟可编译 10000 行以上代码。同时，Visual J++在函数调用中优化字节顺序，既加快了应用程序的执行速度，又不影响跨平台的可移植性。
- Visual J++是一个基于窗口界面的开发工具，这是 Sun 公司基于字符界面的 Java Development Kit（JDK）所无法比拟的。Visual J++提高了所有的 Java 应用系统开发所必须的功能。例如，Visual J++ ClassView 以图标和颜色等可视化方式显示出用户应用项目中所有类的分层体系及类之间的逻辑关系和抽象连接，从而替代早期开发工具中常用的基于文件的视图。帮助开发者掌握类、方法、属性之间的联系；可以使用资源编辑器设计表格和选单，或从 Visual C++等其它 Microsoft 产品中引入已经开发好的表格；可以使用 RDO 或 DAO 集成的数据库访问；可以免费提供 Internet Explorer 3.0，其内部配备了高效的 JIT（Just-In-Time）Applet 编译器，提高终端用户在网络上的导航速度。
- Visual J++支持 ActiveX 控件，可以保护用户以前的投资，这也是 Visual J++与其它开发环境不同的一个地方。在 Visual J++中，用户能够将 ActiveX 控件集成到 Java Applet 中，例如，开发者可以直接使用已有的 ActiveX 控件来构造一个 Java Applet，而无须自己重新开发。开发者甚至可以使用 Microsoft Excel 类现有的应用程序中的某些功能，通过 COM 将这些功能提供出来。这意味着用户可以将 Excel 电子表格放在 Web 页面上，并通过 Applet 实施控制。Visual J++提供的向导可以帮助开发者构造 ActiveX 控件。也可以使用 OLE 与其它应用系统连通。
- Visual J++提供一个图形化的调试器，可以加快对应用程序中错误的定位。Visual J++

是调试 Java 代码的一个强有力的工具，具有标准调试功能，如断点观察。集成的调试器能够浏览调用堆栈，监视线程，甚至到字节码级，用于查找那些特殊的危险的故障。

● 容易学习。Java 是 C++的一个子集，是需要专业编程人员深入学习才能够掌握的一种程序设计语言。Visual J++则不然，它可以在以下几个方面帮助用户快速掌握 Java：

Visual J++的向导可以指导、简化用户构造 Java Applet 过程；

交互式帮助和在线演示；

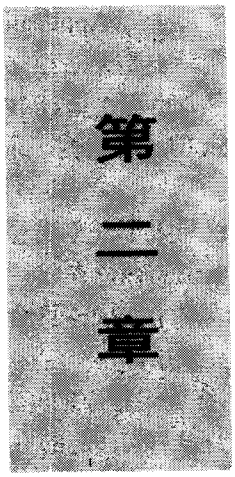
完整的在线文档；

由于 Visual J++是 Microsoft Developer Studio 的一个组成部分，与 Visual C++等其它开发环境的界面完全一致。因此，熟悉 Microsoft 开发环境的开发者可立即使用 Visual J++进行应用开发。

由于 Visual J++所具有的上述特点，使其市场前景比 Sun 公司的 Java 开发工具更看好。

Visual J++ 2.0 之前的版本省略了类向导（ClassWizard），而在 Visual C++中，类向导是非常有用的一个概念。

实际上，Visual J++与其名字并不完全相符，因为它并不是完全可视化的，例如，它不能拖拉用户接口组件，将其全部连接到以 Borland Delphi、Visual Basic 或 Visual J++编写的程序码上。另外，Visual J++当代码发生变化后，在运行前需要关闭浏览器，否则运行的将是改变之前的代码。资源编辑器不允许访问所有的 Java AWT 控件，代码产生是一个单向的过程，不能将自己的类增加到调色板中。这些都是将 Visual J++用作快速应用开发工具的不足。Visual J++更适合完成诸如创建菜单、对话框等基本组件，并将其输入到 Java 应用程序之中去之类的任务。



关于Java

2.1 Java概述

Java 语言是 Sun Microsystem 公司开发的新一代程序设计语言，其目标是为满足在异种机器、不同操作系统平台的网络环境中开发应用软件。Java 语言集面向对象、平台无关、分布式、健壮安全、可移植和多线程于一身，是一种简单、高性能、面向对象的程序设计语言。随着 Internet 和 WWW 的迅猛发展，Java 语言越来越引起人们的重视，正逐渐成为 Internet 上的重要开发语言。

Java 是以 C++为基础设计的，为了使其更加简单，Java 保留了 C++中的大部分功能，舍弃了其中诸如多重继承之类的一部分内容。Java 语言与 C++很相似，因此熟悉 C++的程序员很快便能掌握 Java。但与 C++相比，Java 具有自己的一些特点。

关于 Java 语言的概念和使用，已有许多专著，本书不再赘述。只是根据介绍 ActiveX 和 Visual J++的需要涉及其中的部分内容。

2.1.1 Java的特点与不足

Java 语言具有下述一些重要特点：

（1）平台无关与可移植性：Java 的应用程序运行是以字节代码形式运行于 Java 虚拟机之上的，因此，支持 Java 虚拟机的任何平台都可运行 Java 程序。Java 程序的运行与平台无关，它是独立于操作系统的，不限制在 Windows、Unix、Macintosh 甚至将来其它的平台。Java 为了保证应用程序运行不依赖于平台，还提供了

一个用于访问操作系统底层功能的可扩展类库供用户调用。平台无关性的本身就提供了一种良好的可移植性。由于 Java 的运行系统是用 ANSI C 编写的，编译器是用 Java 编写的，因此对于操作系统和硬件平台而言，Java 的环境本身也是可移植的。

（2）内置网络性：Sun 公司开发的 Java 语言是专门针对客户/服务器环境的。Java 程序可分为 Java Application（Java 应用程序）和 Java Applet（Java 小应用程序）。Java Application 类似其它高级语言编写的程序。而 Applet 是一种特殊的 Java 程序，它实际上是无法独立运行的一些程序片段，通常要通过支持 Java 的网页浏览器下载后执行（当然，也可以通过 JDK 之类的 Java 程序开发工具中的浏览工具观看 Applet 的效果）。由于 Applet 的这种特性，Java 十分适用于世界上最大的客户/服务器网络 Internet。

（3）增强 Web 功能：Java 语言大大增强了 Web 页面功能，它不仅局限于浏览观看，更允许用户与 Web 进行交互作用。例如，一个含有电子元器件图形的电子邮件，在以往的 Web 页面上，只能显示出固定的一些说明性文字信息和元器件图形。而如果在 Web 页面上含有 Java Applet，即可允许用户在电路板上移动电子元器件，实时观察其效果。

（4）坚固性：坚固性包括健壮和安全两个方面。由于 Java 是用于网络应用程序的开发，为了使用户从网络下载程序执行时，保护本地系统不受损害并防止病毒程序，Java 的安全机制是将操作性限制在 Applet 上。Java 编译器不处理内存布局，程序员既不能控制对内存的分配，也无法从类的定义中推断出运行时的实际内存布局。同时，Java 的运行系统不信任任何引入的代码，在运行系统中增加了字节代码验证器，对字节代码进行验证。Java 代码即使通过了 Java 编译器，如果从网络上下载后，只要不符合 Java 语言的限制，字节代码验证器也会发现并予以拒绝。Java 的健壮性主要表现在它取消了内存指针，杜绝了对内存的非法访问。

（5）程序设计简单：Java 是面向对象的程序设计语言，使用的概念不多。虽与 C++相似，但删除了 C++中很少使用或容易出错的许多语言功能，所以比较容易学习。Java 语言中的一个更重要特色是提供一个自动无用单元收集器，它将开发者从复杂的内存管理、非法指针引用等困扰中解脱出来。另外，Java 中的代码重用、继承、可移植性等特性，也使开发者不用花费更多时间，即可将编码迁移到其他平台。

（6）分布式：Java 是一种分布式程序设计语言，支持网络上的应用程序，用户无需很多计算机操作经验就可运行 Java Applet。使用浏览器，用户端自动安装 Applet，系统管理员能够容易地管理网络系统。Java 打开远程文件和打开本地文件几乎一样简单。开发者能够快速容易地将以 Java 开发的应用系统放置到 Web 服务器上，并分发到网络客户端，包括远程用户。如果需要修改软件，可只在服务器端修改 Java 代码。所以，Java 语言十分适合建造企业内部网 Intranet。

（7）多线程：多线程技术可提高用户界面的交互性能。Java 语言中提供了内置的多线程控制，可简化多线程应用程序的开发。而 C++需要由开发者负责线程的锁定与释放，极易产生死锁，往往令人头痛。

与所有程序设计语言一样，Java 也并非完美无暇，也有如下的不足之处。

（1）运行效率较低：Java 的程序编码需要执行两次编译，一次在运行时，一次在设计时，其结果是 Java 程序运行时要比其它语言编写的程序慢。即使采用现代优化技术，Java Applet 的运行速度仍然不快。将来，如果在计算机硬件中加入 Java 芯片，也许可以解决运行效率问题。

（2）学习问题：Java 语言是基于 C++设计的，它所带来的接口和异常管理方面的问题，除非是熟练的 C++程序设计员，否则，较难掌握。

（3）安全问题：程序设计中有一个不成文规则是，未经用户允许，一个应用不允许引入其它应用。但是，Java 的默认设置打破了这个规则。虽然 Java 采取了预防恶意码措施，但有些用户仍然担心外部程序未经证实即可进入自己的系统。Java 对 Applet 进行了一些限制，下面是 Applet 不能执行的任务：

— 在客户机端不能读或写文件；
— 不允许直接存取内存；
— 不允许直接访问其它硬件；
— 不能启动程序；
— 不能产生系统调用；
— 不能找出用户名、电子邮件或 IP 地址。

利用 ActiveX，既可以避免上述限制，也不必担心破坏了 Java 的安全方式，因为 ActiveX 控件有其自己独立的一套安全性措施（详见第七章 7.1.3 节安全性问题）。

2.1.2　Java虚拟机

计算机编程语言基本可分成解释型和编译型两种形式。

解释型语言，是对源码文件直接解释，转换成计算机可读的二进制码，这种语言易于开发，但执行速度较慢，因为编码错误是当程序实际运行时才会发生，因此不易调试。

编译型语言，是在程序运行之前对源码进行编译转换成二进制码，这种语言开发过程较慢，但执行速度快。由于大多数程序错误可在编译过程中发现，因此易于调试。

Java 不属于上述任何一种语言类型，而是二种类型的混合物。简单地说，Java 的应用程序运行是先编译再解释，这个处理过程是由虚拟机完成的。

虚拟机，顾名思义它不是实际存在的计算机。Java 的虚拟机由 Java 解释器和运行系统构成。Java 应用程序的运行过程是，首先要对程序源码进行编译，编译结果不是生成某种可执行的 CPU 指令码，而是生成 Java 特有的字节码（Bytecode）。Java 的字节码是一种与平台无关的对象文件格式，它要运行在 Java 虚拟机之上。当应用程序实际运行时，Java 虚拟机中的解释器再将字节码翻译、解释成可执行的机器代码格式。每一种操作系统的 Java 解释器是不同，但实现的 Java 虚拟机是相同的。如此看来，当用户在 Visual J++下创建 Java Applet 时，其程序代码的目标不是面向 Windows，而是面向虚拟机的。

Java 虚拟机的逻辑成分，包含一组寄存器、一组指令集、一个栈、一个无用单元收集堆、一个方法区域等五个部分。这五个部分不依赖于任何实现技术，但其功能必须在真实的机器上以确定的方式实现。

虚拟机为用户提供了一个非常好的可移植性，Java Applet 可运行于任何具有虚拟机的平台之上，如图 2.1 所示，它不仅包含 Windows、Unix、Macintosh 系统，也可用于用户设备，例如有线电视盒和 Internet 应用等等。

由于面向虚拟机，程序设计者将不再关心操作系统和指令集，不必关心是否习惯或指定运行平台。所有的程序代码都是运行在 Java 虚拟机一种机器之上，用户所关心的将只是

应用的内容。

由于在支持 Java Applet 的 Web 浏览器中也嵌入了 Java 虚拟机，所以任何人在任何支持虚拟机的环境（设备）上开发的 Java 代码都可以通过 Internet 传送到用户的 Web 浏览器之上。

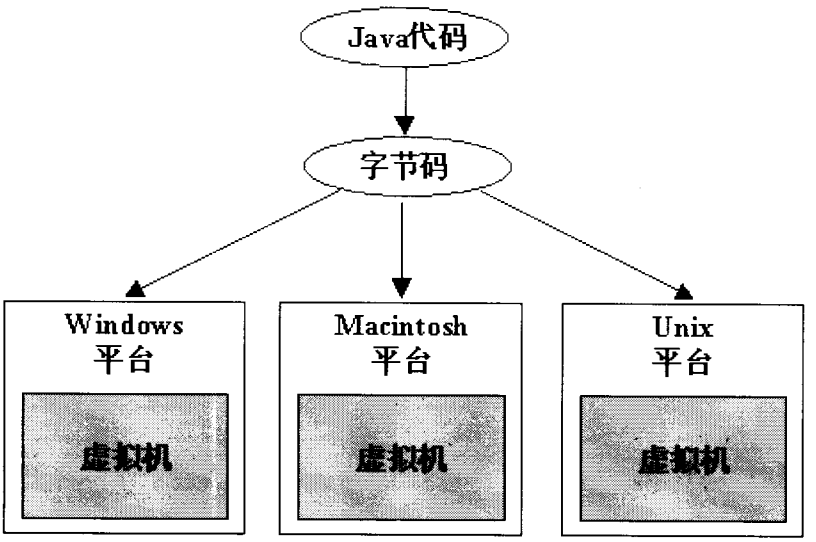


图2.1 编译成字节码的Java代码可以运行于任何具有虚拟机的平台

在Internet上，有些游荡的程序具有潜在的危险性。为了保证安全，嵌入在Web浏览器中的虚拟机不允许在客户机上执行文件存取，不能以任何方式访问客户机，Applet的作用仅局限于屏幕显示。有些Web浏览器的虚拟机还含有特征检查，以发现不需要的软件，制止其屏幕显示。

2.1.3 字节码（Bytecode）解释器

Java 虚拟机中的字节码解释器是一个特殊部件，它装载字节码，将其翻译成计算机可识别的机器指令。由于每一种计算机平台有自己专用的指令集，所以各平台上的字节码解释器基本不同。

字节码与汇编语言相似，是一种以符号描述的低级计算机指令，例如，下面是一组字节码：

```
iinc    y,1        // 整型变量 y 加 1
iload   nMyVar     // 将局部变量 nMyVar 推入堆栈
i2f                // 整型数弹出堆栈，将其转换成浮点数，然后送回到堆栈
```

上述字节码读起来比一般的 Java 代码困难，它更接近于计算机可识别的机器语言格式。字节码的重要性在于便于移植。

在 Java 虚拟机的字节码翻译器中还含有一个字节代码验证器，用以在字节码执行之前对其进行完整性、安全性校验，如图 2.2 所示。这样，可以确保不会发生堆栈溢出、不存在

伪造的指针、没有违反访问权限、严格遵守访问对象的规范、使用有效的参数调用等。

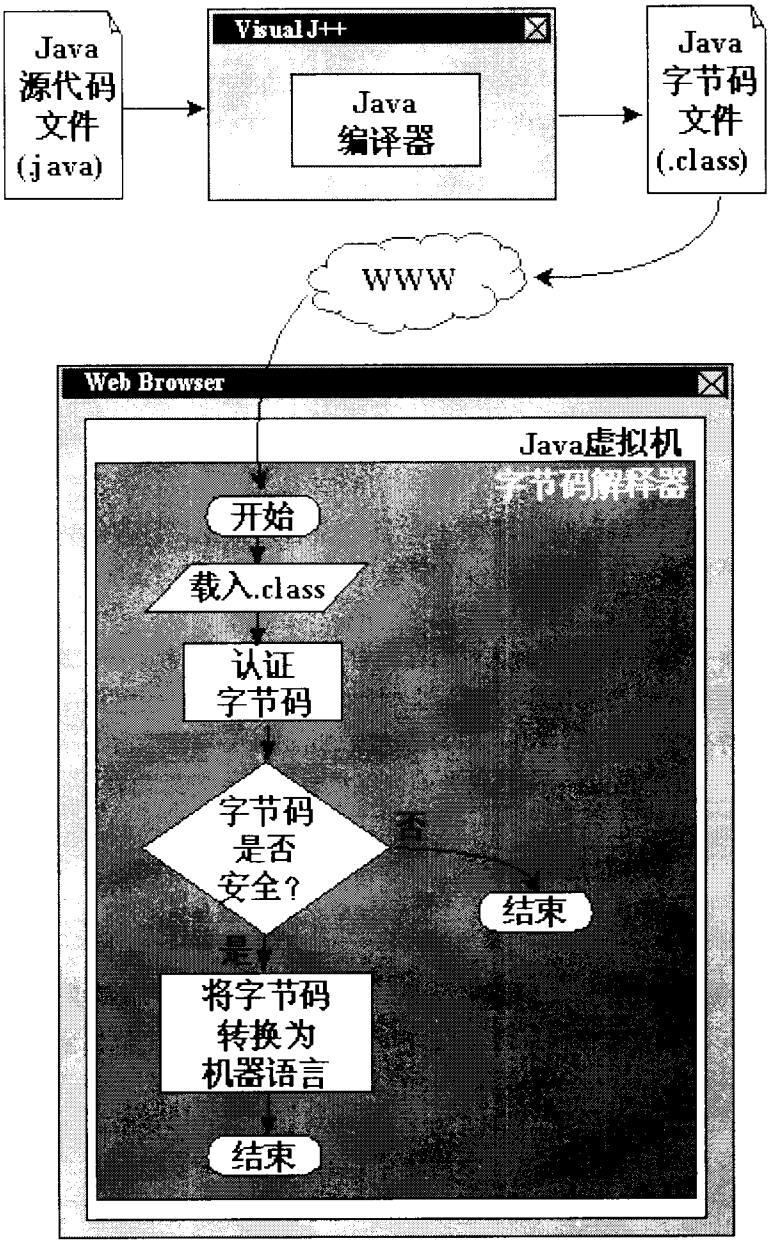


图2.2　Java从源码、字节码到机器语言的处理过程

2.1.4　Just-In-Time编译器

字节码解释器的解释过程和对字节码校验的处理过程比较费时，其结果将使得 Java 程

序的运行速度较慢。为了解决这个问题，Java 工具厂商采用了一个称做 JIT 的处理方案。

JIT（Just—In—Time）的本意是一种使库存减至最低并改进制造过程的管理方法，要求元件和原材料仅在需要时才小批量提供，并且在产品交付前短时间内生产出来。Java 编译器引用这个名词的含义不言而喻。

采用 JIT 技术，当 Java 程序运行时，编译生成的字节码加载到虚拟机后，由 JIT 替代字节码解释器将字节码翻译转换成本地的机器指令，JIT 就是即时（Just—In—Time）编译字节码。

当 Java 应用程序调用函数时，字节码解释器通常必须重复对其进行编译。而 JIT 编译器则是在内存中保留编译过的函数和对代码的链接。如果一个函数被再次调用，则激活链接，使已编译过的代码可立即执行。这个处理过程既避免了字节码解释器重复编译的低效率，也使 Java 代码的执行速度能够接近于 C++代码。

2.1.5 Java的内存管理

Java 虚拟机除了解释并执行指令外，还负责管理数据、寄存器和内存。Java 虚拟机的内存区域不依赖实际内存的位置，也不要求连续，但要求逻辑长度固定。Java 的堆（heap）是运行时刻为对象动态分配的存储区域。

C++和 Java 语言的最大区别是对内存的管理。C++开发者必须时刻牢记分配内存，如果忘记释放内存则很容易产生程序崩溃，这类危险错误是经常发生的。读者不禁要问，为什么长时间以来不解决这个问题。实际上，Lisp 或 Smalitalk 等程序语言已经使用了无用内存单元收集技术（Garbage Collection）来处理内存释放，自动管理内存，使开发者不必再担心如何释放内存。

Java 具有自动无用内存单元收集技术，程序设计不必显式地释放内存空间，这也是它流行的原因之一。虽然大多数程序设计员一致承认自动收集无用内存单元的价值，但也认识到它的一些缺陷。

自动无用内存单元收集的问题之一是，所有一切处理过程都是发生在幕后，用户无法通过内存进行控制。当用户程序运行时，Java 虚拟机启动一个低优先级的后台线程，连续不断的查找无用的内存块。当然，这个线程也要占用计算机的处理资源，但其数量仅占大约百分之三，几乎可以忽略不计。

自动无用内存单元收集的另一个问题是，自动处理的效率不如采用人工技巧处理的效率高。因为程序设计员最熟悉自己的编码，当重新分配内存时，能够作出最好的决定。而 Java 的自动内存收集只能依靠推测与计算，其结果可能是在内存堆（Heap）中产生大量碎片。

为了修复内存碎片，Java 无用内存单元收集器对内存累积采用了下面两种算法：

• 标志—清扫（Mark and sweep）算法：诸如链接表类的环型内存能够欺骗 Java 的无用内存单元收集器，使之将实际不用的内存误认为正在使用中。为了记录这些隐藏的无用内存，收集器通过跟踪每个内存引用并采用“已使用”标记标识出所找到的内存块，到达最后引用之后，收集器再快速的预排内存堆，扫除所查到的内存碎片，如图 2.3 所示。

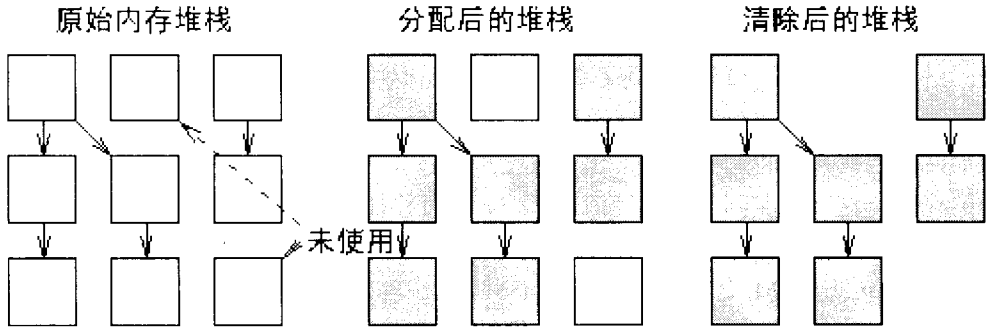


图2.3　标志 — 清扫算法找出隐藏的内存。

• Stop－And－Copy 算法：即使使用“标志－清扫”算法之后，内存堆中仍会含有部分碎片。Java 虚拟机能够以类似于磁盘优化的方式，停止无用内存单元收集，将内存堆复制到一个新的位置，在执行复制操作的同时，将碎片堆整理成为连续的存储块，如图 2.4 所示。

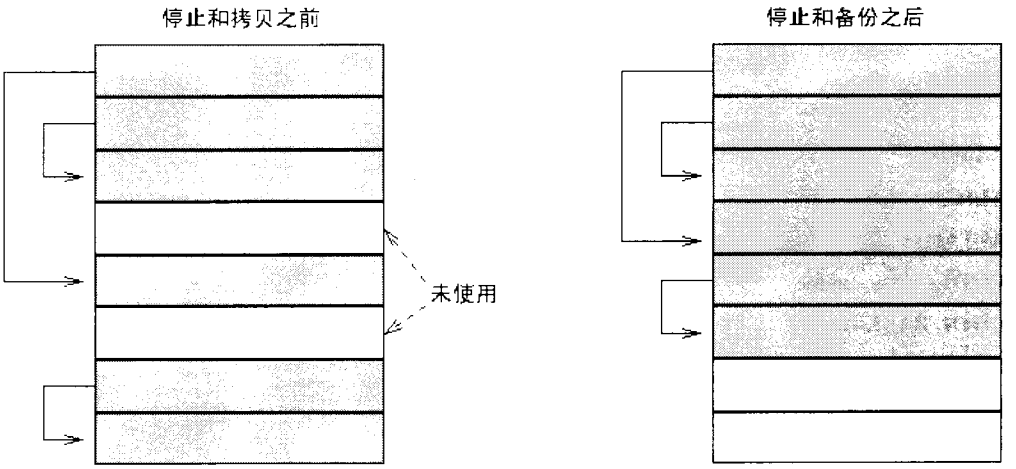


图2.4　Stop－And－Copy算法压缩存储块

Java 提供一个 System.gc（ ）系统调用，该函数的作用是强制无用内存单元收集器清扫、压缩内存堆。调用这个函数，将立即运行“标志—清扫”和“Stop－And－Copy”算法。注意，仅当所分配的环行数据量十分庞大时方可调用这个函数，而在其它所有情况下，无用单元收集器的内在智能性已经可以很好地满足程序运行对内存的要求。

2.1.6　本地码接口

Java 语言提供了丰富的类库，从简单的字符串操作到复杂的多媒体命令集，可以满足大部分应用程序的功能要求。但是，无论 Java 类库如何详尽，仍有一些程序设计中的功能

要求无法处理，例如，开发者希望编写一段将磁盘信息备份到磁带上的 Java 处理程序，或者让用户的 Java 应用程序能够利用某些图形卡的硬件技术特性。而目前，Java 类库还不提供直接访问硬件的能力。由于受到 Java 类库的限制，有些任务在 Java 中是无法实现的。用户的唯一资源是本地码接口（Native Code Interface），它允许 Java 直接调用 C 或 C++代码。为了解决直接访问硬件的问题，本地码接口还提供了一种在 Java 中使用已存在代码的方法。

然而，本地码接口也远非理想。出于安全原因，本地码接口从 Web 浏览器的运行中保护用户的代码。这样，实际上它已经破坏了 Java 可移植性这一最大特色。同时，它还强制用户对凌乱的头文件和生成的存根文件保持跟踪。一个更容易、安全和具有可移植性的方法是使用 Micorsoft 的 COM（Componet Object Mode）。与本地码接口一样，COM 允许用户直接访问硬件，可从 Java 程序中调用 C 或 C++代码，用户甚至可以从 Delphi 及其它任何与 COM 兼容的开发工具中导入代码。COM 与本地码接口的区别是，COM 与 Web 浏览器完全兼容。如何使用 COM 将在第七章介绍。

2.1.7 〈Applet〉标识

为了使 Web 页面使用 Java Applet，开发者必须首先创建一个含有〈Applet〉标识的 HTML 文件，〈Applet〉标识允许 Web 浏览器寻找、加载并显示 Applet。〈Applet〉标识的格式如下所示：

```
<Applet>
ALIGN=对齐类型
ALT=text
CODE=class - name
CODE BASE=url
HEIGHT=n
HSPACE=n
NAME=text
VSPACE=n
WIDTH=n>
<PARAM  NAME=text  VALUE=“value”
< PARAM…>
…
error-text
</Applet>
```

上述程序中每行的含义如下：

ALIGN 设置 Applet 的对齐方式，“对齐类型”可有下述值之一：

- BASELINE：Applet 底部与文本外界的基线对齐。
- CENTER：Applet 位于左、右边缘的中心，在 Applet 之后的文本开始于下一行。

• LEFT：Applet 与左边缘对齐，其后的文本环绕 Applet 的右边界。
• MIDDLE：Applet 的中央与环绕文本基线对齐。
• RIGHT：Applet 与右边缘对齐，其后的文本环绕 Applet 的左边界。
• TEXTBOTTOM：Applet 的底部与其后文本的底部对齐。
• TEXTMIDDLE：Applet 的中央与其后文本的基线和 X 高度之间的中心点对齐。
• TEXTTOP——Applet 的顶部与其后文本的顶部对齐。

一部分文本型的 Web 浏览器可以排列〈Applet〉标识，但不能显示 Applet。这种浏览器将显示跟在 Applet 之后由 ALT 参数所指定的“text”文字。

CODE 指定含有 Java Applet 主类文件的名称，例如 MyApplet.Class，要求必须具有这个参数。

CODBASE 参数指定通知浏览器寻找 Applet 的 URL 地址，例如，HTTP://MyServer.Com/MyDirectory/。无论 Applet 是否位于用户的 HTML 文件目录之中，都必须使用这个参数。

HEIGHT 指定用于 Applet 的建议高度，要求必须具有这个参数。

HSIGHT 指定水平装订线，它是在 Applet 和文本或图像至 Applet 的左或右边缘之间的一个空白区域。

NAME 建立 Applet 的名称，程序设计者可在窗体和脚本中，使用这个名称作为 Applet 的引用助记符。

VSPACE 指定垂直装订线，它是在 Applet 和文本或图像至 Applet 的上或下沿之间的一个空白区域。

WIDTH 指定用于 Applet 的建议宽度，要求必须具有这个参数。

〈PARAM〉参数允许程序设计者建立 Applet 加载后的直接属性。例如，如果用户的 Applet 显示一个 JPEG 图像的内容，则可通过〈PARAM〉将 JPEG 文件的名称传递到 Applet，其〈Applet〉标识如下：

```
<APPLET
CODE=My_JPEG_Viewer.class
WIDTH=100
HEIGHT=200>
<PARAM NAME=JPEG_File_Name VALUE="My_JPEG_File.jpeg">
</APPLET>
```

并非所有的 Web 浏览器都能显示 Java Applet，不兼容的浏览器将显示<Applet>标记的 error-text 部分，该部分提示 Web 用户无法显示整页，需要更换使用合适的 Java 浏览器。

2.2 Java在Windows中

1995 年 12 月 7 日 Microsoft 从 Sun Microsystem 获得了 Java 使用许可证，这个协议允许 Microsoft 销售 Java 开发工具（例如 Visual J++），允许在其 Web 浏览器 Explorer 中提供