最新Internet技术基础与应用系列丛书

CMP

(美) Michael Morrison 著
周苏明 常 征 等译

sams net

Presenting JavaBeans

JavaBeans 使用手册

机 械 工 业 出 版 社

西蒙与舒斯特国际出版公司

最新 Internet 技术基础与应用系列丛书

JavaBeans 使用手册

(美) Michael Morrison 著

周苏明 常 征 等译

冯志强 审校

机 械 工 业 出 版 社

西蒙与舒斯特国际出版公司

本书介绍了 JavaSoft 最新推出的基于 Java 的软件组件技术 JavaBeans。

全书共分五部分。第 1 部分概述了软件组件技术和 JavaBeans。第 2 部分逐个讨论了 JavaBeans 内部的属性操纵、内省、事件处理、持续性以及定制等功能。第 3 部分利用前面介绍的理论创建了四个示例 Bean。第 4 部分讨论了一些 JavaBeans 的高级议题并展望了 JavaBeans 的未来。第 5 部分的附录详细列出了 JavaBeans 的联机资源及 JavaBeans API 快速参考。

本书介绍的是最前沿的软件开发技术,内容详实,示例丰富,是广大程序员和计算机用户的不可多得的参考书。

本书版权登记号:图字:01－97－1082

图书在版编目(CIP)数据

JavaBeans 使用手册/(美) Michael Morrison 著;周苏明等译 .—北京:机械工业出版社,1997.9

(最新 Internet 技术基础与应用系列丛书)

书名原文:Presenting JavaBeans

ISBN 7-111-05972-7

Ⅰ. P… Ⅱ.① … ②周… Ⅲ. Internet 技术基础,JavaBeans 使用手册 Ⅳ. TP311.13

中国版本图书馆 CIP 数据核字(97)第 19675 号

出 版 人:马九荣(北京市百万庄南街 1 号　邮政编码 100037)

责任编辑:李成刚

北京昌平第二印刷厂印刷·新华书店北京发行所发行

1997 年 10 月第 1 版第 1 次印刷

787mm×1092mm 1/16·12.75 印张·296 千字

0 001－6 000 册

定价:23.00 元

译者的话

这是一本介绍最新 JavaSoft 的软件组件技术——JavaBeans 的书。JavaBeans 是目前最热门的网络编程语言 Java 的扩充。相信本书的面市会给广大程序员和计算机用户带来帮助。

本书是多人共同努力的结晶，除了封面上署名的译者外，参加本书翻译的还有欧梅、王春、吴华、刘芝泉、王强、李丽、贾志明、蔡汇锦以及王睿等人。

在翻译过程中，我们认真查阅了各种相关资料，对各种疑难问题进行了深入讨论，互相统一了词汇。但是由于时间仓促，加上本书技术含量较高，挂一漏万之处在所难免，欢迎读者批评指正。

1997 年 5 月于北京

前　言

当 Java 以它特有的方式成为 Internet 中标准的程序设计语言和首选运行环境时，许多人已经开始在思考 Java 的未来了。JavaSoft 作为 Java 的开发者，在它取得明显的成功之后，显然也有了类似的观念。JavaSoft 认识到了在 Internet 中 Java 明显潜伏着的许多问题，但它也开始发现其优势已经远远超过网络上的其他联机应用程序。JavaSoft 没有坐等他人对 Java 采取行动，而是抓住了纠正 Java 缺点的机会，对它运用了新型技术，使之发展成为一个功能丰富的软件技术。与 Java 相关的新技术之一就是 JavaBeans，这是 Java 对组件软件的回答。

也许读者对组件软件不太熟悉，它是一种高度围绕代码的可重用与划分而设计的软件。组件软件是一种十分流行的强劲的观念，在整个软件工业界中得到迅速应用，提高了开发的效率。软件组件的设计与开发使它可以在不同的开发与运行方案中得到访问与利用。JavaBeans 组件软件技术是基于 Java 的，提供了创建与使用作为软件组件的 Java 类的方法。因为许多人认为缺少组件软件技术是 Java 的一大缺点，所以 JavaBeans 对于 Java 的未来是相当有意义的。

JavaSoft 也意识到了这种需求，并且立即在工作日程中给予 JavaBeans 以很高的优先权。当开始评估 JavaBeans 的最初目标时，JavaSoft 的设计者们试图使用一种非常简单的任务说明来陈述 JavaBeans 将达到的目标，这个任务说明如下：

“一次性编写，在任何地方运行，在任何地方可重用”。

任务说明使用一组非常简单、明了、优雅的需求表示了 JavaBeans 的目标。这里的第一个需求就是“一次性编写”，指出了 JavaBeans 需求的代码一旦被编写，当增加或完善功能时，就不需要再次编写代码。第二个需求是“在任何地方运行”，指 JavaBeans 组件能够运行于广泛的操作系统平台的需要。最后一个需求“在任何地方可重用”，指 JavaBeans 组件在各种应用程序和不同类型的开发环境中应用的要求。

尽管 JavaBeans 任务说明的需求被公认为有点儿含糊不清，但它无疑描绘出了该技术将达到的蓝图，本书致力于探求 JavaBeans 技术，并且解释这些任务说明是如何与 JavaBeans 的各部分相吻合的。通过讲述该技术的各种基础知识，在本书中读者可以从概念上了解 JavaBeans 的全貌。读者同样可以从非常实际的角度了解许多 JavaBeans 的知识，其方法是设计自己的 JavaBeans 组件，它可以在用户自己的 Java applet 或应用程序中得到再次利用。

尽管本书的主要前提是向读者介绍 JavaBeans 技术，笔者认为读者还是会很高兴地为发现 JavaBeans 内容的深度而感到惊奇。即便如此，笔者将努力使用实用性概念去讲述技术细节，以便读者跟上进度。当读者阅读完全书并实际使用之后，将会同意这样一个观点：JavaBeans 是自 Jave 以来最有可能让人感到兴奋的技术。在写作本书的过程中，笔者编写了许多有趣的 JavaBeans，希望将它们应用到笔者自己的项目中去。

本书的读者范围

这本书从各种不同角度涵盖了 JavaBeans 技术。正因为如此，本书适合于具有不同技术背景与知识的各种读者。从基本概念的角度来说，本书需要一点对 Java 程序设计语言与运行系

统的基本知识。然而,本书的第3部分探讨了JavaBeans组件的创建并且需要对Java程序设计语言本身有一定的了解,如果读者本人是一名Java程序员,你将发现自己正适合于本书,尤其是第3部分。在另一方面,如果读者仅仅对了解JavaBeans技术的概念方面感兴趣,仍将发现本书有许多有用的真知灼见。

即便不考虑读者的技术背景或希望学习JavaBeans的原因,但要记住,为了充分地欣赏JavaBeans全貌,对于Java总体上的认识是需要的。这归因于这样一个事实:JavaBeans是Java技术本身的扩展。如果读者对Java没有预先了解的话,笔者鼓励你参考一本或多本的讲述Java程序语言和运行系统的书。

本书是如何组织的

本书分成4个部分,还有三篇附录。每一部分都以不同的方法来讲述JavaBeans技术。尽管在各部分之间存在着一定程度的重复,但各部分的目标都是为了从不同的角度来讲述JavaBeans。尽管这些部分不完全是前后连贯的,但依次阅读是很有好处的。

在第1部分中,读者可以了解软件组件的基本概况,以及它们对未来软件的发展为什么如此重要,然后读者可以了解JavaBeans以及JavaBeans API的基础知识。

在第2部分中,读者将接触到JavaBeans API的特性。本书该部分每章的重点在于JavaBeans API的基础知识,这些基础API对应于JavaBeans的主要功能领域,包括属性、内省、事件、持续性与定制等内容。

在第3部分中,读者通过了解如何创建自己的Bean而从概念阶段进入了实用阶段。通过认识常用Bean结构的基础,开始这一工作。从这里出发,读者可利用剩余的章节来开发自己的Bean,这些Bean包括一个趣味按钮Bean、一个计量条Bean、一个LED显示Bean,以及一个声音播放器Bean。

在第4部分里,读者将了解一些高级的和未来的JavaBeans议题。先学习Bean在手工编码的应用程序中是如何应用的,从这里开始,读者将遇到各种高级JavaBeans问题,然后是对JavaBeans未来的展望。

附录中提供了JavaBeans参考信息,包括一份联机的JavaBeans资源清单、JavaBeans API快速参考、本书所附光盘的描述,以及词汇表等。

本书使用的约定

本书使用边条来指明一些重要信息。这些边条以及它们各自的含义如下:

新术语: 新术语边条用来指明在讨论中引入的新术语。
注释: 注释边条用于指明与讨论有关的令人感兴趣的信息。
警告: 注意边条用于指明与讨论有关的潜在问题。

目　录

第 1 部分　JavaBeans 概述

第 2 部分　JavaBeans API 内核

第3部分 创建 Bean

第 5 部分 附 录

第1部分　JavaBeans概述

第1章　软件组件的基础

如果一本有关JavaBeans的书开始不解释软件组件的概念基础(正是它形成了JavaBeans技术的基础),那么该书就不会有太多的用处。通过提供软件组件技术的特定方法,JavaBeans能够使软件开发者设计并且创建出可重用的软件部分,它们可以容易地与软件的其他部分、应用程序,甚至与开发工具彼此结合。如果JavaBeans的这一描述对读者有用,那太好了;否则,读者也不要感到气馁,因为笔者有意提前用这种方式对JavaBeans进行描述。即使它存在于JavaBeans的核心,有关软件组件的概念也不是希望读者在本书掌握的。实际上,本章的目的在于通过研究软件组件以及为什么它们对于未来的软件如此重要,从而为本书开个好头。

说明软件组件魅力的最好的办法是寻找发明它们的原因。换句话说,也就是软件组件设法解决的问题是什么?对这些问题的最简单、最直接的回答就是软件的可重用,它是尽可能多地在每个新开发的项目中应用过去工作的挑战。即使已经形成了各种不同的软件组件方法,但直到目前为止,还没有充分地提供创建真正可重用软件的方法。在本章中,读者可了解到许多有关这个问题的内容以及用来解决该问题的基础概念:软件组件。读者将了解软件组件是如何以各种途径来改善软件开发的,这将使开发者在新代码场合下,花费更多的精力来处理已有的代码,而不是分离或抛弃它。更重要地是,本章说明了JavaBeans的背景,它可能是最令人兴奋和最有前途的软件组件技术。

在本章中,读者可以了解到以下内容:

- 面向软件业的问题
- 软件组件基础
- 可视化与非可视化组件
- 组件模型

1.1　软件组件的需求

包含在软件开发革新领域的变化可能比其他专业领域的更多。软件进入最终的商业领域只须一段很短的时期,随之而来需要完善与提高的步骤是立即规划并且开始。正因为这个原因,没有任何其他领域能够夸耀自己的产品的"新颖和完善"能超过软件业。但是,所有这些革新都归功于价格。价格是试图在短时间内创建奇迹的压力。软件开发者经常处于很大的压力下,他们必须快速地工作并且推出更好的成果。毫无疑问,这种压力也存在于其他行业中,但是软件开发界以它能够满足看起来难以完成的期限而感到自豪。

由于软件开发者经常需要在短期内发布具有包装功能的应用程序,因此,他们经常必须把精力集中于如何在开发应用程序时减少困境。这种困境通常导致代码严重地依赖于特定的应用程序,而对于其他特定的项目来说,该代码则很少有用处。尽管在许多情况下这是可以接受的,但

最终的结果是花费在项目中的努力对于未来项目的开发很少提供帮助。换句话说,它倒退至为每个新项目画框图或者键盘的地步。如果读者在开发某个应用程序时,能再利用需要同样功能的另一应用程序中的类似功能,岂不是更好吗?当然是这样,但笔者这里超前了一点。

不管开发者在项目开发过程为了减少困境需要怎样的细致,仅仅是项目的规模就会导致许多问题。当项目超过了可以管理的规模时,就连最细致、最深思熟虑的设计也可能导致严重的混乱。

在这种情况下,庞大的应用程序的混乱和复杂已经达到不可管理的程度,要想弄明白其意义是极端困难的。不少软件开发小组在认识到他们正在使用复杂的、令人困惑的代码基础走向死胡同时,纷纷放下手边的工作而转向了新的项目。即使代码编写的很好,仍会有一些限制达到在代码层让组织结构帮助弄清事物含意的程度。

恐怕读者会认为笔者落伍了,或者仅仅由于过份的消极而忽视了在软件开发技术上最新的进展。依笔者看,为了改善这种方案,面向对象的设计方法与编程语言已经走过很长的一段路。即使如此,没有任何一种语言能够真正满足充分地可重用软件标准的需求。与C++很相似,面向对象的语言类虽然好,但它们仍受到如下固有的限制:执行的地址空间、通信需要的特定协议、编译它们的平台等。换句话来说,程序员可以可重用C++,它是面向对象的语言类,但仅仅是在他们开发的应用程序的特定的语境中,这有时是有用的,而其他情况下则受到限制。同样的问题也存在于 Java 中,它在功能上类似于C++类,尽管它在大多数情况下通过交互式平台确实增加了许多优点。

新术语: 平台是一种特定操作系统与运行环境,如 Windows 95 或 Solaris 等 。

新术语: 交互式平台是指不需要任何特殊修改就可以在不同平台上执行的软件。

笔者已经描绘出一幅关于软件世界的相当丑陋的图画,可能让读者感到失望 与气馁,对吗?可能不完全如此,笔者还是充满信心地认为软件的开发方式中存在一些重要问题和一些必须去着手改善的事情。面向对象的编程是一个在正确的方向上的很大的进步,因为它们能够使程序员处理对象,而不必为过程和数据担心,这与人类的思考方式非常类似,因为我们生活的世界充满了对象。

面向对象程序设计语言的问题是仅在特定的程序模型范围,诸如C++与 Java 内强调对象范例。当用户在特定目标平台上使用相同的编程语言时,它当然会使用户较容易地再利用代码,但它对更宽视野的软件却无能为力,后者包括了许多不同的编程语言和平台。问题是在代码层之外,面向对象的思想不能够得到充分的实现,这是令人感到羞耻的。

注释: 支持多平台的问题是值得关注的,它已随着 Web 的流行变得更为重要,它把使用各种类型的计算机硬件与操作系统的用户集合起来。

软件界已经朝着大范围可重用代码的观念发展了一段时间。但是还没有出现一种技术来对许多固有的软件可重用问题提供解答。原因在于真正的解决方案不仅能使开发人员在一个特定的应用程序中容易地可重用代码,而且还要跨越多个不同的平台,甚至是在分布式网络环境,诸如 Internet 中再利用代码。最终,面向未来的可行的软件技术必须能够容易地集成到客户/服务器模型中,它已成为大多数现代计算机系统的标准。

除了这些需要之外,代码可重用问题的长期解决方案必须针对软件的多种版本的存在提

供完善的方案。正如前文所讲,软件处于连续的变化状态之中,这推动了处理多个版本的需求。指出软件成长与扩展的道路是极为重要的,因为它能够使开发者无止境地进行革新来完善刚刚产生的思想。

1.2 软件组件的开始

软件开发界对于探索软件的可重用问题已经有一段时间了。读者可能已经听说过可重用软件的更流行的名称"软件组件"。为了使读者不致于困惑,这里做一下解释,组件是指那些能够容易地组装起来,以更高的开发效率创建应用程序的可重用软件部分。为了不使读者认为这个观念听起来有些不切实际,读者只需要大致地回顾一下近一个世纪以来这一相同观念的不同类型的应用。笔者是指在工业革命时期,流水线生产手段得以发展,并且引入了装配机械。这个观念如果应用到软件中,就是一次性建立小型的、可重用的组件,并且尽可能多地可重用,从而使整个开发过程成为一个整体。

新术语: 软件组件是指分离成独立的、易于可重用结构的软件部分。

尽管组件软件具有它本身的优点,但由于各种原因,充分可重用的软件还要真正地建立其本身。可考虑最起码的事实:软件业与其他在工业革命中产生的行业相比,仍然是十分年轻的行业。它仅仅说明了在整个软件生产过程中花费时间来消除困难的原因。如果读者像笔者一样,将会拥抱发生在软件界的快速变化,并且品味出这样一个事实:你是这场信息革命中的一分子。

也许软件组件不得不面临的最困难的问题是:当今所用的各种各样的微处理器和操作系统是不同的。在软件组件中有各种可以理解的尝试,但它们始终局限于特定的操作系统之中。Microsoft 的 VBX 和 OCX 组件结构在 PC 世界中是相当成功的,但是它们对于缩小与其他类型的操作系统之间的差距则无能为力。考虑到在众多操作系统中运行的固有的依赖于平台的组件技术需求的工作量,这仅仅说明 Microsoft 只关注于 PC 机市场。

注释: 实际上,Microsoft 新的 ActiveX 技术是 OCX 技术的修定版本,用来提供与各种平台相兼容的通用组件技术。然而,考虑到 ActiveX 对 32 位 Windows 代码的依赖性,已经可以看出 Microsoft 打算如何解决平台依赖性这一问题。读者将在第 15 章了解到 JavaBeans 与 AetiveX 之间具体的相同与差别。

在学习 Internet 之前,平台的依赖性问题并不是这个重要观念的全部内容,PC 开发者不必过多地耽心他们的产品不能在 Solaris 系统上运行。一些 PC 开发者放弃了他们的赌注,并且将应用程序移植到 Macintosh 平台上,但是大多数都需要非常冗长和资源密集型的开发工作。整个方案随着与 Internet 产生的端口相融合的操作系统而变化。结果导致对于开发这样的软件产生了新的兴趣:不考虑其运行的操作系统,每一个人都可以使用。Java 已经成为使真正意义上的平台独立性软件开发成为现实的一个重要因素。然而,直到目前,Java 还没有对软件组件提供答案——读者等一会儿就会知道,实际上,这贯穿了本书的其余部分。

如果平台依赖性问题不充分,对一些现存的组件技术同样会产生影响,因为它们必须使用特定的程序设计语言或具体的开发环境。就像平台的依赖性在运行时削弱组件一样,把组件的开发限制在特定的编程语言或开发环境中也会在开发端削弱组件。软件开发者需要能够自己决定对一个特定的任务什么语言是最合适的。同样,开发人员希望能选择最适合他们自己的开发环境,而不是被迫使用基于组件技术局限的开发环境。

因此，任何实际的长期组件技术都必须处理平台依赖性与语言依赖性这两个问题。它们带领读者进入 JavaBeans：JavaSoft 的 JavaBeans 技术是一种直接回答这两个问题的组件技术。为了创建和使用动态 Java 软件组件，JavaBeans 被实现为一种独立于结构与独立于平台的应用程序编程接口（API）。JavaBeans 吸收了其他组件技术忽略的技术，使用可移植的 Java 平台作为提供完整的组件软件解决方案的基础，它可以方便地在网络世界得到应用。在笔者退出叙述之前，请让笔者停下来提醒读者在本书的其余部分中考虑 JavaBeans 的细节问题。现在继续讨论通常意义上的软件组件。

1.3 可视化软件组件

如果读者对迄今为止有关软件组件的讨论感到有些混乱的话，希望本节能澄清这些问题。到目前为止，读者对软件组件的了解都比较抽象，有时领会起来是困难的。理解软件组件的一种较好的方式就是观看组件的特定子集：可视化组件。可视化组件是在父应用程序的显示表面上需要物理空间以进行可视化表示的软件组件。父应用程序有时更多地是指容器。读者在本章后面的 1.6 节中将会了解到更多的有关容器的知识。

新术语： 可视化组件是一种在父应用程序的显示表面上需要物理空间以进行可视化表示的软件组件。

可视化组件最简单的示例可能就是按钮。按钮是一种图形元素，它可以与包含它的应用程序完全区别开来。许多可视化设计工具，或应用程序建立器工具，都对以图形方式操作按钮提供支持。这说明按钮是一个独立的实体，它能以独立于任何父应用程序的方式被安排并交互作用。按照这种方式，按钮功能作为一种分离的自我包含的单元，而成为软件组件的关键特色之一。尽管按钮是一个独立的单元，它所提供的真正作用是能够容易地集成到应用程序中。使用应用程序建立器工具，可以在单击或拖动鼠标时轻松地增加按钮。图 1-1 显示了在 Visual

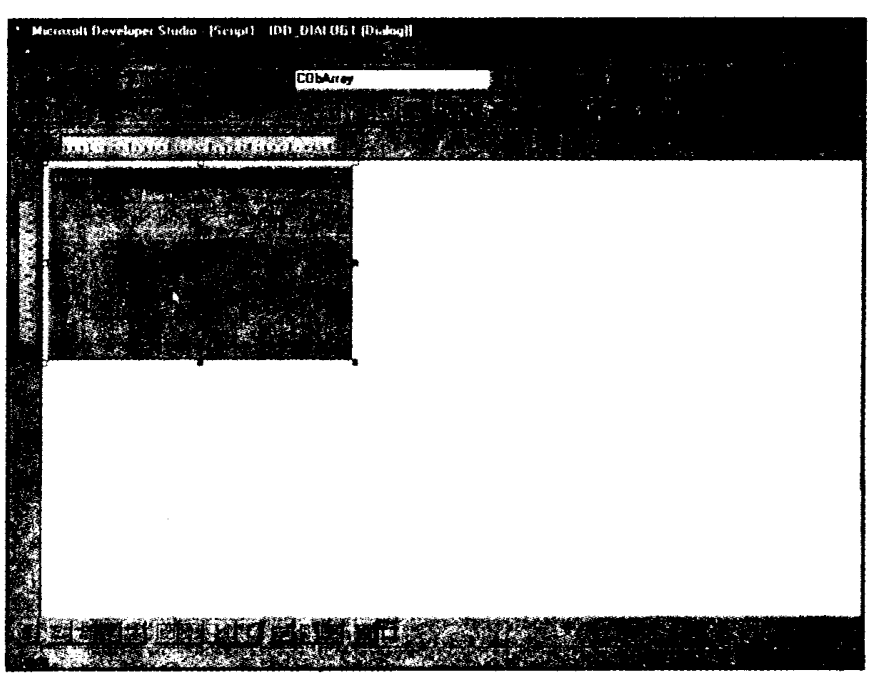

图 1-1 增加到 Visual J++ 对话框中的按钮

J++中添加到对话框中的按钮，这是一种流行的 Java 开发工具。

图 1-1 所示的按钮是一种可视化的软件组件，正如图中所表现的那样。除了它的可视化表示，按钮组件同样能以编程方式相互作用。例如，可以指定一段代码当按钮被按下时执行。按下按钮的动作称为用户的输入事件，如果应用程序对了解事件感兴趣的话，可视化组件通常为父应用程序生成输入事件。在本书的第 6 章中将会了解到更多的关于事件以及它与 JavaBeans 相关联的知识。

许多其他类型的可视化组件也在可视化开发环境，诸如 Visual J++中得到支持，这些组件包括复选框、列表框、文本编辑框等。记住在 Visual J++中使用的可视化组件的局限性在于它们实际上是基于标准 Java 类的。这种情况的一种例外是 ActiveX 控件的使用，它提供了一些真正的软件组件所需要的高级功能。当然，JavaBeans 组件也是一个例外，但是笔者并不希望在这里过多地超前叙述。

1.4 非可视化软件组件

如果可视化软件组件听起来像软件组件技术的理想化应用的话，理解非可视化组件在某些情况下也是十分有益的。有一个十分流行的 Visual Basic 控件就是 Timer 控件，它在运行环境中是完全不可见的。在固定的时间间隔内（例如每隔一秒钟），Visual Basic 的 Timer 控件能够用来触发事件。Timer 控件对于产生计时循环是非常有用的。因为 Timer 控件完全使用在程序设计层，所以在运行时没有必要提供组件的图形化视图。利用 Timer 控件的好处是使用控件的应用程序根本不必考虑控件的内在实现方式。应用程序使控件处理自己的事务，这就是软件组件的优点！

非可视化软件组件的另一种较好的示例就是假定的拼写检查组件，它能够处理文本并找到拼写错误的单词。由于这种拼写检查器是一种自包含组件，它可以集成到任何应用程序中，使这些应用程序获得拼写检查的功能。例如，相同的拼写检查器组件可以应用于字处理器与电子邮件应用程序中。这将把程序开发人员从设计与实现自己的拼写检查器的紧张的编写工作中解放出来。这样，他们就可以从第三方组件供应商那里购买拼写检查器组件，或者，如果他们愿意，也可以开发自己所需的拼写检查器组件。拼写检查器的功能隔离于自包含单元中，这个组件可以在应用程序中以最小的代价来使用。

1.5 软件建立块

本次讨论的最后一个问题就是组件（包括可视与不可视）等同于软件建立块。使用组件，用户可以一次性地创建应用程序中独立的功能部分。为了更好地领会在软件设计的发展过程中组件为什么如此重要的原因，可以考虑一些现实世界中的建立块。如果读者需要把泥瓦匠作为自己的第二职业，将首先购买一些砖块、泥板与泥刀等。利用泥刀并且略晓使用泥刀的常识，无须太多困难，读者就可以自己建出一垛墙来。

在软件世界中，事件并非如此简单。首先，在建墙时，砖块和泥板都是准备好的，但开发软件时情况却不同。更常见的情况是，用户找到的软件处于明显比自己计划的低的细节层。结果，用户不得不花费很多时间来开发应用程序的低层部分。对于泥瓦匠来说，就是自己制造砖块，这显然不是一名成功的泥瓦匠所必须掌握的技巧。

事情并不止这些！回到软件设计的另一面，读者购买的许多软件都局限于特定的平台，这

意味着如果读者要在不同的平台上开发,经常会处于孤立状态。也就是说,在读者的泥瓦匠生涯中,当希望建筑一垛不同的墙时,必须知道如何制出完全不同类型的砖块。另外,读者还有可能不得不购买新的泥刀,以适应设计某种类型的墙的需要。

好了,笔者的比喻可能有一些混乱,但笔者认为读者一定了解了这一观点。正如泥瓦匠建墙时用的砖与泥板一样,软件组件是软件开发的建立块。

1.6 组件模型

现在,读者已经有了关于什么是软件组件的好的观念,下面就来更深入地了解一下组件的工作原理。软件组件技术的核心是组件模型,它定义了组件的结构以及组件是如何在外部进行操作和相互作用的。由组件模型定义的结构主要负责决定组件在动态环境下是如何相互作用的。理解组件模型及其相关的结构是了解软件组件的主要图景以及它们是如何工作的关键。

所有的组件模型都定义了两种基本元素:组件与容器。组件模型的组件部分表明了不同组件的创建与利用的基础。换句话说,组件模型提供了创建实际组件的模板。组件模型的容器部分定义了将组件集中起来成为有用结构的方法。容器提供了安排组件以及与其他组件相互作用的语言环境,例如,一个使用一组组件的应用程序可以作为组件的容器。

新术语: 容器是能够使组件组合在一起并相互作用的语言环境。

容器有时也叫做窗体、页面、框架、或外壳,可以作为应用程序的基础。有一点容易引起混乱的是,容器同样可以是组件。尽管初听起来有些奇怪,但这种功能却是重要的,因为它能使组件相互嵌套,产生复杂的可视化图形界面。图 1-2 所示就是一个在 Visual J++ 中的组框组件,它用作包含对话框中一组按钮的容器。

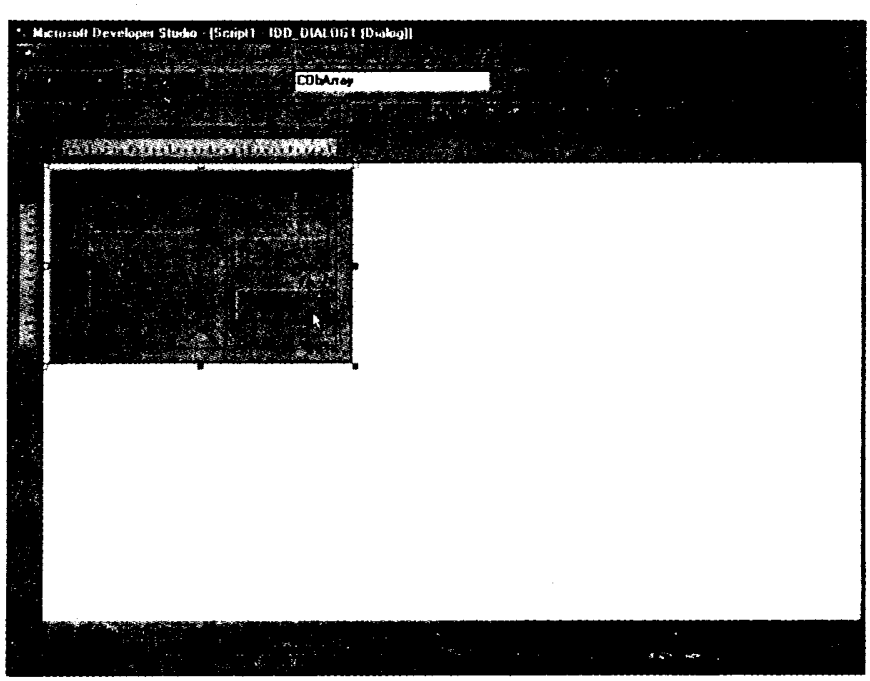

图 1-2 在 Visual J++ 中作为容器的组框组件

除了定义组件的结构与容器外,组件模型同样负责提供各种形式的服务。更特殊地是,功

能完善的组件模型可支持下列六种主要服务：

1.内省
2.事件处理
3.持续性
4.布局
5.对应用程序建立器的支持
6.对分布式计算的支持

1.6.1 内省

内省是向外部世界展示组件功能的机制。通过内省，应用程序能够对组件提出询问，设法找到它的功能并且与组件相互作用。内省是组件模型的关键特征之一，因为它负责决定组件对应用程序以及其他组件的外观。请读者回忆一下，组件的重要需求之一就是它是完全自包含的。如果组件既是自包含的，又是从外部可用的，那么它必须完全支持内省。

新术语： 内省是向外部世界展示组件功能的机制。

1.6.2 事件处理

事件处理是一种能够使组件产生事件通知的机制，这种通知与组件内部状态的变化相对应。当组件状态变化时，组件产生事件通知，告诉所有感兴趣的组件。这些感兴趣的部分要么是父应用程序，要么是其他组件。事件处理机制按照使事件能容易地被获得，并且以统一的方式做出响应的方式构成。

新术语： 事件是组件中发生的动作，应用程序或其他组件希望了解它并对它产生响应。

例如，回忆一下本章前面提到的按钮组件，当用鼠标单击它时，它将产生一个事件。在这种情况下，按钮状态的变化由单击按钮这一事实反映出来。这种状态的变化引起了一个事件的产生并且通知给任何感兴趣的事件收听者。假定一个父应用程序是感兴趣的收听者，这个父应用程序具备特定的代码来处理按动按钮这一事件，它是根据接收的事件通知来执行的。

新术语： 事件收听者是被设计成对特定组件作出响应的应用程序或组件。

按钮示例中的事件通知与对事件的响应的问题可能看起来非常简单和直接。然而要记住，把事件转发给各个收听者的整个机制是在组件模型中必须详细描述的。而且，这种机制必须与众多组件和事件类型相一致，应用程序或组件才能对任何事件作出响应。

1.6.3 持续性

持续性是指在固定的地点，诸如硬盘存储和获取组件的方式。实际上，存储和获取的有关组件的信息是组件的内部状态，连同它与容器或其他组件的关系。使用这种信息，组件可以安全地存储，并且在以后再次产生。在设计工具中，持续性是特别重要的，它能使开发者修改组件的属性以适应特定的应用程序。

新术语： 持续性是指在固定的地点存储和获取组件的方式。

1.6.4 布局

组件模型的另一个重要部分是它对组件的物理布局的支持。物理布局实际上仅仅应用于