第一部分　快速开发的基础

第1章　Delphi 5下的Windows编程

本章内容：

- Delphi产品家族
- Delphi是什么
- 历史回顾
- Delphi的IDE
- 创建一个简单的应用程序
- 事件机制的优势在哪里
- 加速原型化
- 可扩展的组件和环境
- IDE最重要的十点功能

这一章让读者对Delphi有一个总的认识，包括它的历史、功能、怎样适合 Windows开发环境以及作为一名Delphi程序员必须应该知道的一些重要信息。同时为了使读者的技术更加娴熟，本章还讨论了关于Delphi IDE的必备知识，指出了一些难以发现的功能，这些功能即便是经验丰富的 Delphi程序员也不一定知道。本章不准备教给你那些开发 Delphi软件所需的非常基本的东西。我们相信你为买这本书付出不小的一笔投资，是为了学到新的和有趣的知识 —— 而不是重读早就在Borland的文档里看到过的内容。因此我们的任务就是：向你介绍 Delphi的强大功能及最终怎样调用这些功能来开发商业性软件。但愿我们的经验能够帮助我们不断地向你提供一些有趣和有用的知识。我们相信，只要新程序员明白这本书不是为最初起步用的，那么有经验的和新的Delphi程序员就都能从本章(和本书)中获益！从Borland文档和简单的例子开始，一旦你明白了 IDE的工作原理和应用开发的流程，就请到这本书里来畅游一番！

1.1　Delphi产品家族

为了满足不同层次的要求，Delphi 5分为三种版本：Delphi 5标准版、Delphi 5专业版、Delphi 5企业版。每种不同版本面向不同的开发者。

Delphi 5标准版是一个入门级的版本。它能够编写简单的Delphi应用程序，对那些业余爱好者和想开始学习Delphi编程的学生来说是最理想的。这一版本包括以下功能：

- 优化的32位Object Pascal编译器。
- 可视化组件库(VCL)，包括组件选项板上85个以上的组件。
- 支持包，可以创建精巧的应用程序和组件库。
- IDE，包括编辑器、调试器、窗体设计器和许多其他功能。窗体设计器支持可视化窗体的继承和链接。

- Delphi 1，用以支持16位的Windows应用程序开发。
- 全面支持Win32 API，包括COM、GDI、DirectX、多线程以及Microsoft和第三方软件开发包(SDK)。

Delphi 5专业版适用于不需要客户/服务器功能的专业开发者。如果你是个正在创建和开发应用程序或Delphi组件的专业开发人员，那么这个产品对你是最适合的。除了包含标准版的所有功能外，Delphi 5还包含下列功能：

- 组件选项板上150个以上的VCL组件。
- 数据库支持，包括数据感知VCL控件、Boland数据库引擎(BDE)5.0、本地表的BDE驱动器、数据集结构(用来将其他的数据库引擎嵌入到VCL中)，数据库浏览器、数据共享库、支持ODBC数据源以及Interbase Express本地Interbase组件。
- COM组件生成向导，例如ActiveX控件、Active窗体、Automation服务器以及属性页等。
- QuickReport报表工具，可以建立基于数据库的报表。
- TeeChart图表组件，用于数据的可视化分析和显示。
- 单用户的Local Interbase Server(LIBS)，让你即使没有网络环境也可以开发基于SQL的Client/Server应用程序。
- Web发布功能，可以方便地在Web上分发ActiveX项目。
- InstallSHIELD Express应用程序制作工具。
- OpenTools API，可以用于开发自己的组件并集成到Delphi环境中，也可作为与PVCS版本控制功能的接口。
- WebBroker和FastNet向导和组件，用于开发Internet应用程序。
- VCL和RTL(运行期库)的源代码及属性编辑器。
- WinSight32工具，可以浏览窗口和消息信息。

Delphi 5企业版主要面向客户/服务器领域的开发者。如果需要开发访问SQL数据库服务器的应用程序，这个版本包含了客户/服务器应用程序开发过程需要的所有配套工具。除了包含前面两个版本的一切功能外，Delphi 5企业版还包括以下功能：

- 组件选项板上的200个以上的VCL组件。
- MIDAS(Multitier Distributed Application Services)的支持和开发许可，使多层应用程序的开发大大简化。
- 支持CORBA，包括3.32版的VisiBroker ORB。
- InternetExpress XML组件。
- TeamSource资源控制软件，允许进行小组开发，并支持不同版本引擎(包括ZIP和PVCS)。
- 支持本地Microsoft SQL Server 7。
- 对Oracle 8的高级支持，包括抽象数据类型字段。
- 对ADO(ActiveX数据对象)的直接支持。
- DecisionCube组件，使你能够进行可视化的、多维的数据分析。
- 提供访问InterBase、Oracle、Microsoft SQL Server、Sybase、Infomix和DB2数据库服务器的SQL Links BDE驱动器，并且允许无限制地分发这些驱动程序。
- SQL数据库浏览器，可以浏览和编辑特定服务器的元数据。
- 图形化查询建立工具SQL Builder。
- SQL监视器，可以监视与SQL服务器的通信，从而可以调整SQL应用程序的性能。
- Data Pump Expert，用于快速数据迁徙。
- 五用户的InterBase for Windows NT许可。

1.2 Delphi是什么

我们经常会问这样的问题：“到底什么使得Delphi如此优秀？”和“为什么和别的编程工具相比，我更愿意选择Delphi？”等等。这些年来，我们对这类问题已经得出了两种答案，一长一短。短的就是：高效性。要创建Windows应用程序，使用Delphi是我们能够找到的最为简捷的途径。当然，有些人(老板们和未来的客户们)并不满足于这个答案。因此，我们必须推出我们的详细解答，它阐述了使得Delphi如此高效的综合因素。我们把决定一个软件开发工具效率的因素归结为以下五点：

- 可视化开发环境的性能。
- 编译器的速度和已编译代码的效率。
- 编程语言的功能及其复杂性。
- 数据库结构的灵活性和可扩展性。
- 框架对设计和使用模式的扩充。

虽然还有许多其他因素应该包括进去，如配置、文档、第三方的支持等，但我们已发现这是向人们解释我们为什么选择Delphi的最确切、最简单的方式。当然，上述五点也可能包含了一些主观因素，但关键在于：你使用一种特定工具进行开发时，到底能有多大的效率？如图 1-1所示，对一种工具的各方面性能进行评估量化 (1到5之间)，并分别标在图1-1的各条轴线上，最后就能得到一个五边形。五边形的面积越大，则这种工具的效率越高。

毋需告诉你我们使用这种方法得到了什么答案 —— 你自己一试便知！下面让我们来仔细地看一下 Delphi在这几方面的性能如何，并把它们和其他Windows开发工具做一比较。

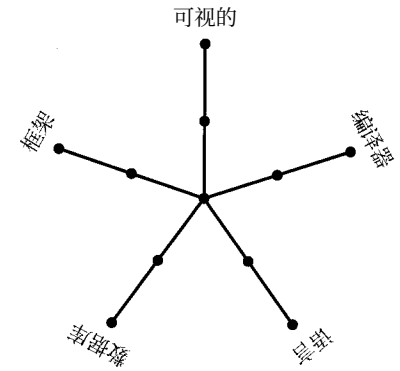

图1-1 开发工具效率图

1.2.1 可视化开发环境

可视化开发环境通常分为三个组成部分：编辑器、调试器和窗体设计器。和大多数现代 RAD(快速应用开发)工具一样，这三部分是协同工作的。当你在窗体设计器中工作时， Delphi在后台自动为你正在窗体中操纵的控件生成代码。你还可以自己在编辑器中加入代码来定义应用程序的行为，同时还可以在同一个编辑器中通过设置断点和监控点等来调试程序。

总的来说Delphi的编辑器和其他工具的编辑器类似，但它的 CodeInsight技术却省去了许多输入工作的麻烦。这一技术是建立在编译器信息之上的，而不是基于像 Visual Basic等使用的类型库，因此应用范围更广泛。虽然Delphi的编辑器也设置了许多不错的配置选项，但我觉得 Visual Studio的编辑器配置余地更大。

在版本5里，Delphi的调试器功能终于赶上了Visual Studio的调试器，具备了许多先进的功能，如远程调试、过程关联、 DLL和包调试、自动本地监控以及 CPU窗口等。Delphi还支持在调试时随意放置和停靠窗口并把这一状态保存为命令的桌面设置。由此，Delphi的IDE实现了对调试功能的良好支持。正如经常在一些集成环境 (如VB和某些Java工具)中见到的那样，一个性能非常完善的调试器的长处就在于：应用程序被调试时能修改它的代码，从而改变它的行为。遗憾的是，由于这种功能在编译成本地代码时过于复杂而无法实现，故不能为Delphi所支持。

对RAD工具(如Delphi、Visual Basic、C++Builder和PowerBilder等)来说，窗体设计器是一项独特的功能。一些更为经典的开发环境，如 VC++和BC++，都提供了对话编辑器，但却没有将窗体设计器集成到开发流程中。由图1-1的效率图可以看出，没有窗体设计器将会降低开发工具的整体效率。几年

来，Delphi和Visual Basic在完善窗体设计器的功能方面展开了激烈的竞争。它们的新版本功能一个比一个强。Delphi的窗体设计器的与众不同之处在于，Delphi是建立在一个真正面向对象的框架结构基础之上的。这样，你对基类所做的改变都将会传递给所有的派生类。这里涉及的一项关键技术就是VFI(visual form inheritance)，即可视化窗体继承。VFI技术使你能够动态地继承当前项目或对象库中的任何其他窗体。一旦基窗体发生改变，派生的窗体会立即予以更新。在第 4章“应用程序框架和设计”中有对这一重要功能的详细解释。

1.2.2 编译器的速度和已编译代码的效率

快速的编译器可以使你逐步递进地开发软件，经常地修改源代码、重新编译、测试、再修改、再编译、再测试.....形成这样一个良好的开发循环。如果编译速度很慢，开发者就不得不分批地修改代码，每次编译前进行多处修改以适应一个低效率的循环过程。提高运行效率、节约运行时间、生成的二进制代码更为短小，其优越性是不言而喻的。

也许Pascal编译器最著名的特点就是速度快，而Delphi正是建立在这种编译器的基础之上的。事实上，它可能是针对Windows的最快的高级语言本地代码编译器。以往速度很慢的 C++编译器在近年来取得了很大的进步，增加了链接和各种缓存策略，尤其是在 Visual C++和C++Builder中。但即便如此，C++的编译器还是比Delphi的慢了几倍。

编译速度一定能与运行效率成正比吗？当然不是。Delphi和C++Builder共享同一种编译器后端，因此生成的代码等效于由一个优秀的 C++编译器生成的代码。根据最新的可靠评估标准，Visual C++在许多场合都被认为在编译速度和生成代码长度方面是最有效的，这得益于一些极为有力的优化措施。虽然对通常的应用程序开发来说，这些细小的优越性难以被注意到，但如果你正在编写复杂的计算代码，那么它们就会发挥作用。

Visual Basic的编译技术有点特别。在开发过程中，VB以一种集成的方式运作，而且反应相当敏锐。这种编译器速度比较慢，生成的可执行代码的效率也远远不及 Delphi和C++工具。

Java是另一种有趣的语言。最新的基于 Java的工具语言JBuilder和Visual J++自称其编译速度能赶得上Delphi，但是生成代码的执行效率却不尽人意，因为Java是一种集成语言。虽然Jave在稳步地前进，但在大多数场合，其运行速度却仍与Delphi和C++相距甚远。

1.2.3 编程语言的功能及其复杂性

在旁观者的眼里，一种语言的功能和复杂程度是极为重要的，这也是许多争论的热点。对这个人来说简单的东西，对那个人来说可能很难；对这个人来说功能有限的东西，对另一个人来说却可能是非常完美的。因此，以下几点仅源于作者个人的经验和体会。

从根本上来说，汇编是一种最有力的语言。用它你几乎无所不能。但是，即便是用汇编开发最简单的应用程序，难度也非常大，还可能一无所获。不仅如此，要想在一个小组开发环境中保留一段汇编代码，不管保留多长时间，有时也是根本不可能的。因为代码从一个人传给另一个人、再到下一个人，设计思想和意图越来越不明朗，直到代码看起来如同天书。因此，我们对汇编的评价很低，它虽然功能很强大，但对几乎所有的开发者来说都太复杂了。

C++是另一种极为有力的语言。在它的潜在功能 (如预处理器宏、模板、操作符加载等等)的帮助下，你几乎可以使用C++设计你自己的语言。只要合理地使用其丰富的功能选项，就可以开发出简洁直观、易于维护的代码。然而，问题是，许多的开发者总滥用这些功能，这就很容易导致发生重大错误。事实上，写出糟糕的C++代码反倒比写出好的C++代码更容易。因为这种语言自己不会朝着好的设计方向前进 —— 这由开发者决定。

Object Pascal和Java给我们的感觉很相似，因为它们很好地把握住了复杂性和功能性的平衡。它们

都采取了这样一种途径，即限制其可用功能以加强开发者的逻辑设计。例如，两者都避免了完全面向对象但却容易被滥用的多重继承的观念，而是实现了一个执行多重接口功能的类。两者都不支持美观却危险的操作符加载。两者都有一些强大的功能，诸如异常处理、运行期类型信息 (RTTI)和生存期内存自管理字符串。同时，两种语言都不是由专门的编委会写出来的，而是来自于单个组织中对这种语言有着共同理解的的个人或小组。

Visual Basic最初是为了使编程初学者入门更容易、进步更快而设计的 (名字也由此而来)。但是作为一种语言，VB也要不断地取长补短，这使得它近年来也变得越来越复杂了。为了对开发者隐藏这些细节，VB仍然保留了一些向导以创建复杂的项目。

1.2.4 数据库结构的灵活性和可扩展性

由于Borland缺少一种数据库计划，因此Delphi保留了我们认为是所有工具中最灵活的数据库结构。对大多数基于本地、客户/服务器和ODBC数据库平台的应用程序来说，BDE的功能都非常强大。如果你对此不满意，可以避开使用BDE以支持新的本地ADO组件。如果你没有装ADO，可以自己创建数据访问类或者购买第三方数据访问解决方案。此外，MIDAS使对数据源的多层访问更易于实现。

Microsoft的工具(ODBC、OLE DB或者其他)从逻辑上来说趋向于支持Microsoft自己的数据库和数据访问解决方案。

1.2.5 框架对设计和使用模式的扩充

这是一项经常被其他软件设计工具忽略了的重要功能。 VCL是Delphi最重要的组成部分。在设计时操纵组件、创建组件、使用OO(面向对象)技术继承其他组件的行为，这些能力都是决定 Delphi效率的关键因素。在许多场合，编写VCL组件都采用固定的OO设计方法。相比之下，其他基于组件的框架经常过于死板或过于复杂。比如ActiveX控件具有和VCL控件相同的设计期性能，但却不能被继承以创建一个具有其他不同行为的新类。传统的类框架，如OWL和MFC，需要你有大量的内部结构知识，而且如果没有RAD工具的设计期支持，其功能将会受到抑制。将来能够与 VCL的功能相媲美的一个工具是Visual J++的WFC(Windows Foundation Classes)，即Windows基础类。但是由于Sun Microsystems对Java问题的诉讼仍悬而未决，Visual J++的前景还不明确。

1.3 历史回顾

从核心上说Delphi其实是一个Pascal编译器。自从15年前Anders Hejlsberg写下第一个Turbo Pascal编译器以来，Boland就一直在推动着Pascal编译器向前发展，而Delphi 5是迈出的又一步。Turbo Pascal具有稳定、优雅以及编译速度快等特点， Delphi 5也不例外，它综合了数十年来编译器的经验和最新的32位优化编译技术。虽然近年来编译器的功能有了显著增加，它的速度却只减慢了很少。另外，Delphi的性能仍然非常稳定。

下面就让我们循着记忆的足迹再回过头去看一看Delphi以前的各个版本以及每一版本发行的背景。

1.3.1 Delphi 1

在DOS的年代，程序员只有两种选择：要么是易于使用但速度慢的 BASIC语言，要么是效率高但却复杂的汇编语言。 Turbo Pascal以其结构化语言的简练和真编译器的性能，综合了两者的优势。而Windows 3.1的程序员同样面临两种选择：一种是强大却难以使用的 C++，一种是容易使用但语言有局限的Visual Basic。对此，Delphi 1提供了一种完全不同的开发 Windows程序的方法：可视化的开发环境、编译后的可执行软件、 DDL、数据库以及可以毫无限制地给可视环境命名。而 Delphi 1是第一个

综合了可视化开发环境、优化的源代码编译器、可扩展的数据库访问引擎的 Windows开发工具，它奠定了RAD的概念。

综合了RAD工具和快速数据库访问的编译器 —— Delphi对众多VB程序员来说极具吸引力，因此它赢得了许多忠诚的用户。同时，很多的 Turbo Pascal程序员也转向了这一功能强大的新工具。而 Object Pascal由于和我们在大学学过的Pascal语言不同而给人们的编程工作带来了困难，因此更多的程序员开始使用Delphi这种由Pascal支持的稳健的设计模式。 Microsoft的VB小组因为在Delphi面前缺少严肃的竞争意识而失败了，迟钝而臃肿的Visual Basic 3显然不能和Delphi 1同日而语。

这些都发生在 1995年。当时Boland由于一桩侵权案而起诉 Lotus要求赔偿巨额损失，同时还从Microsoft中引进人才以求与Microsoft在应用程序领域一比高低。而后Boland把Quattro的业务出售给了Novell，并用dBASE和Paradox进行数据库开发。当Boland正忙于开发应用程序市场时， Microsoft以其平台业务从Boland手里悄然夺走了很大一部分 Windows开发工具的市场。于是 Boland重新把重点放在了它的核心 —— 开发工具上。

1.3.2 Delphi 2

一年后的Delphi 2在32位的操作系统Windows 95和Windows NT下实现了原有的一切功能。另外，Delphi 2还增加了许多Delphi 1没有的功能，例如32位的编译器能生成速度更快的应用程序，对象库得到进一步丰富和扩展，完善了数据库支持，改进了字符串处理，支持 OLE和可视化窗体继承以及与16位的Delphi兼容等。Delphi 2成为衡量其他RAD工具的标准。

这是1996年的事。在此前一年(即1995年)的下半年， 32位的Windows 95出台了。这是自Windows 3.0以来最重要的Windows平台。Boland迫切希望Delphi成为这一平台的最佳开发工具。曾经有一件有趣的事， Delphi 2最初被命名为Delphi 32，以强调它是为32位Windows设计的。但在出版前改成了Delphi 2是为了表明Delphi 2是一种成熟的产品。

Microsoft试图用Visual Basic 4予以反击，但却由于其性能不完善、缺少16位到32位的兼容、存在致命的设计缺陷而倍受困扰。不过不管怎样，仍然有相当数量的人在继续使用 Visual Basic。Boland希望Delphi能进入被PowerBuilder等工具垄断的高端客户/服务器市场，但这一版本还不具有这种实力。

在这段时期公司的战略重点不可否认地集中在顾客身上。作出这样一个方向性调整，毫无疑问是由于dBASE和Paradox所占市场份额的缩小和在C++市场所得收入的减少。为了使这一努力尽快见效，Boland公司做出了一项错误的决定，即兼并了 Open Environment公司。这家公司主要生产两种中间产品：一种过了时的基于DCE的中间产品(可被称为CORBA前身)和一种即将被DCOM取代的分布式OLE专利技术。

1.3.3 Delphi 3

在研制Delphi 1的时候，Delphi开发小组集中精力想推出一个震撼性的产品。在研制Delphi 2的时候，开发组主要考虑把Delphi升级为32位代码，同时又保持对16位版本的兼容。为了满足IT产业的需要，Delphi 2增强了数据库和客户/服务器的功能。到了研制Delphi 3的时候，开发组开始考虑要为Windows开发者所遇到的棘手问题提供一套完整的解决方案。 Delphi 3使本来极其复杂的COM、ActiveX、WWW应用程序开发、“瘦”客户应用程序、多层数据库系统体系结构等技术变得非常容易使用。虽然Delphi 3和Delphi 1编写应用程序的基本方法大都相同，但Delphi 3的代码内视(Code Insight)技术却简化了代码编写过程。

这是在1997年。市场竞争也出现了一些有趣的现象。在低端， Microsoft的Visual Basic 5终于开始有所改观，它采用了一个新的编译器以解决长期存在的性能问题，同时还具有对 COM/ActiveX的良好支持和一些新的平台功能。而在高端， Delphi已成功地战胜了PowerBuilder和Forte等产品。

在Delphi 3的开发过程中，Delphi的首席设计师Anders Hejlsberg决定转到Microsoft公司工作，因此Delphi小组失去了一个重要成员。不过该小组并没有失去任何优势，因为资深设计师 Chuck Jazdzewski有能力承担起领导角色。在此前后，公司还失去了首席技术总裁 Paul Gross，他也是去了Microsoft。有人认为，这一损失与其说是对日复一日的软件开发事务的一个冲击，不如说是影响了公共关系。

1.3.4 Delphi 4

Delphi 4致力于使Delphi更易于使用。Module Explore技术的引入使程序员能够以一致的图形界面浏览和编辑代码。代码导航和类自动生成的功能使程序员只需关注应用程序本身，而不必在输入代码上花费太多精力。IDE经过重新设计可支持浮动和可停靠的工具栏和窗口，调试器也做了改进。Delphi 4不愧为一个领先的开发工具，它的MIDAS、DCOM和CORBA等技术使Delphi 4的应用范围扩展到企业级。

这些都发生在1998年。这一年Delphi有效地巩固了它在竞争中的地位。虽然Delphi仍在持续而缓慢地占领市场，其前沿却在某种程度上得到了加固。几年来 Delphi一直是市场上最稳定的开发工具，Delphi 4在长期的Delphi用户中赢得了信誉，因为它使用简单、稳定性好。

1.3.5 Delphi 5

Delphi 5在几个方面取得了进步：首先，Delphi 5和Delphi 4一样，通过增加更多的功能使程序的编写更简单，程序员可以把精力都集中在想写什么而不是怎样写上。这些新功能包括：进一步增强了IDE和调试器的功能、提供了TeamSource小组开发软件和转换工具等。第二，Delphi 5也为简化Internet的开发增加了许多新功能，包括：Active Server Object Wizard用于创建ASP、InternetExpress组件用于支持XML和新的MIDAS功能，使Delphi成为Internet的一个通用数据平台。第三，Delphi 5最重要的特征——稳定性。就像好酒一样，伟大的软件不可能产生在匆匆忙忙之中，Boland直到Delphi 5完全令人满意才将它推出。

Delphi 5是在1999年下半年出版的。这一年里Delphi继续向企业渗透，而Visual Basic也继续在低端和它竞争。不过战线看起来还很坚固。Inprise(Boland于1998年改名为Inprise)除了继续赢得长期客户的信赖外，还有信心在整个市场上重新恢复Boland的声誉。由于CEO (首席执行总裁)Del Yocam的突然离去和Internet-savvy CEO Dale Fuller的加盟，公司的执行部门经历了一段纷乱时期。而Fuller将公司的重点重新放在了软件开发上。希望Inprise能最终回到正确的轨道上。

1.3.6 未来

尽管历史很重要，但更重要的是Delphi的未来。以历史为导引，我们可以肯定在未来的很长一段时间内，Delphi都将继续是一种优秀的Windows开发工具。我想，真正的问题是我们能否不断地见到针对Win32以外的平台的Delphi版本。根据Boland公司传出的信息，似乎这也正是他们所关心的问题。在1998年的Boland董事会上，Delphi的首席设计师Chuck Jazdzewski演示了一种能生成Java代码的Delphi编译器，这种编译器从理论上来说能用于任何一种带有 Java Virtual Machine的计算机。虽然这一技术还存在一些明显的障碍，但它肯定了这样一种观点，即将 Delphi移植到其他的平台是未来计划的一部分。在最近召开的1999年度Boland董事会上，CEO Dale Fuller在致开幕辞时无意中透露了将开发一个用于Linux平台的Delphi版本的计划。

1.4 Delphi 5的IDE

如图1-2所示，Delphi的IDE主要包括七部分：主窗口、组件面板、工具栏、窗体设计器、代码编

辑器、对象观察器(Object Inspector)和代码浏览器。

图1-2 Delphi 5的IDE

1.4.1 主窗口

主窗口可以认为是Delphi IDE的控制核心。它具有其他Windows应用程序的主窗口所具有的一切功能。主窗口主要包括三部分：主菜单、工具栏和组件面板。

1. 主菜单

与其他Windows应用程序一样，可以通过主菜单创建、打开或保存文件、调用向导、查看其他窗口、修改选项等等。主菜单的每一项都可以通过工具栏上的一个按钮来实现。

2. Delphi工具栏

工具栏上的每个按钮都实现了IDE的某项功能，诸如打开文件或创建项目等。注意工具栏上的按钮都提供了描述该按钮功能的tooltip。除了组件面板，IDE中有五个独立的工具栏：Debug、Desktops、Standard、View和Custom。图1-2显示了这些工具栏上缺省的按钮配置。不过你只需在一个工具栏上右击，在弹出的菜单中选择“ Customize(定制) ”，就可以增加或去掉一些按钮。图1-3所示即为Customize对话框，如果要添加按钮，只要把它们从该对话框中拖到工具栏上即可；如果要去掉按钮，则把它拖离工具栏。

IDE工具栏的定制功能并不仅限于配置需要显示的按钮，还可以调整工具栏、组件面板和菜单栏在主窗口中的位置。要做到这一点，只需拖动工具栏右首凸起的灰色条即可。当拖动时，如果鼠标落在了全窗口区域的外部，就会看到另一种定制形式：工具栏可以在主窗口内浮动，也可以停靠在它们自己的工具窗口内。图1-4显示的是浮动的工具栏。

3. 组件面板

组件面板是一个双层工具栏，它包含了IDE中安装的所有的VCL组件和ActiveX控件。各选项页和组件在面板中的顺序和外观可以通过右击它或从主菜单中选择Component|configure Palette进行调整。

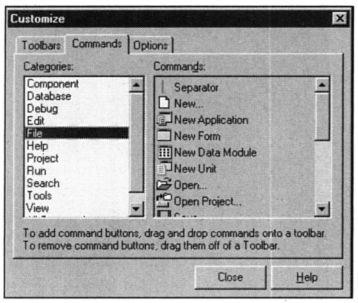

图1-3 Customize对话框

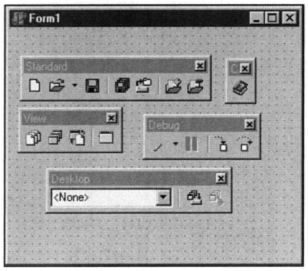

图1-4 浮动的工具栏

1.4.2 窗体设计器

窗体设计器刚开始的时候是一个空白的窗口。可以把窗体设计器看作是艺术家的画布，在这块画布上可以描绘出各种各样的Windows应用程序。应用程序的用户界面正是由窗体实现的。只要从组件面板上选择一个组件并把它放到你的窗体上，就能够实现与窗体设计器的交互。可以用鼠标调整组件在窗体设计器上的位置和大小，还可以用Object Inspector和代码编辑器来控制组件的外观和行为。

1.4.3 Object Inspector

利用Object Inspector，可以修改窗体或组件的属性，或者使它们能够响应不同的事件。属性(propery)是一些数据，如高度、颜色、字体等，它们决定了组件在屏幕上的外观。事件 (event)则是一种消息处理机制，它能够捕捉某种情况的发生并做出反应，像鼠标单击和窗口重画就是两种典型的事件。Object Inspector类似一个带标签的多页笔记本，包括Properties页和Events页，切换时只需在窗口上部点击所需页的标签即可。至于Object Inspector中显示哪个组件的属性和事件，取决于在窗体设计器中当前选择哪个组件。

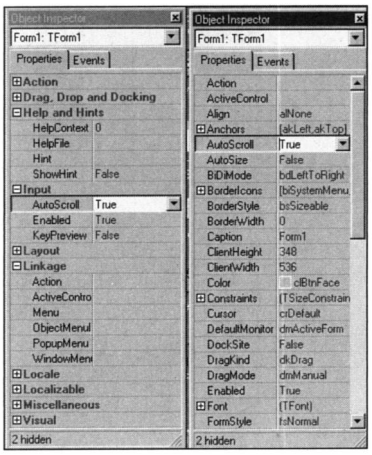

图1-5 按种类或名称查看Object Inspector

Delphi 5新增的一项功能是可以按对象的种类或名字字母顺序来排列Object Inspector的内容。要做到这一点，只需在Object Inspector中右击任何一处并从快捷菜单中选择Arrange即可。图1-5中并列显示了两个Object Inspector，左边一个按种类排序，右边一个按名字排序。你还可以从快捷菜单中选择 View来指定你想看到的对象种类。

一个Delphi程序员必须应该知道的，也是最实用的一点就是，帮助系统是和Object Inspector紧密结合在一起的，如果你想了解某个属性或事件的帮助信息，只要在该属性或事件上按下F1键。

1.4.4 代码编辑器

代码编辑器是输入代码来指定应用程序行为的地方，也是 Delphi根据应用程序中的组件自动生成代码的地方。代码编辑器类似于一个多页的笔记本，每一页对应着一个源代码模块或文件。当向应用程序中加入一个窗体时， Delphi会自动创建一个新的单元，并添加到代码编辑器顶部的标签中。当进行编辑的时候，快捷菜单提供了很多的选项，如关闭文件、设置书签等。

技巧 如果想同时看到多个代码编辑器，可以选择主菜单下的View，New Edit Window。

1.4.5 代码浏览器

代码浏览器以一种树状视图的方式显示了列在代码编辑器中的单元文件。通过代码浏览器，可以方便地在单元文件中漫游或在单元文件中加入新的元素或者把已有的文件改名。要记住代码浏览器和代码编辑器有一对一的关系。在代码浏览器中右击一个节点即可以看到该节点的可用选项。也可以通过修改主菜单下的Environment Options对话框中的Explorer页来控制代码浏览器的行为，如排序和过滤等。

1.4.6 源代码生成器

当对窗体设计器中的可视化组件进行操作时，Delphi IDE会自动生成Object Pascal源代码。最简单的例子就是，当用File | New | Application菜单命令创建一个新的项目时，将看到屏幕上出现一个空白的窗体设计器，同时，代码编辑器中自动出现了一些代码，如清单1-1所列。

清单1-1 一个空白窗体的源代码

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs;

type
  TForm1 = class(TForm)
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

end.
```

应该注意到和任何窗体对应的源代码模块都驻留在单元文件中。虽然每个窗体都对应着一个单元文件，但并不是每个单元文件都对应着一个窗体。如果对 Pascal语言不太熟悉、不清楚单元的概念，请参见第2章“Object Pascal语言”，这一章针对从C++、Visual Basic、Java或其他语言转到Pascal的新手，详细介绍了Object Pascal的语法。

在上述源代码清单中，有这么几行代码：

```
type
  TForm1 = class(TForm)
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

可以看出，窗体对象是从TForm继承下来的。Delphi已经清楚地标出了可以插入公共(public)和私

有(private)变量的地方。现在不必管对象(object)、公共(public)、私有(private)到底有什么含义，第2章会详细介绍这些概念。

下面这一行非常重要：

```
{$R *.DFM}
```

Pascal语言中的$R指令用于加载一个外部资源文件。上面这一行表示把 .DFM(代表Delphi 窗体)文件链接到可执行文件中。 .DFM文件中包含了在窗体设计器中创建的表单的二进制代码。其中的“ * ”不代表通配符，而是表示与当前单元文件同名的文件。例如假设上面一行是在一个名为“ Unit1.pas ”的文件中，则“ *.DFM ”就代表名为“ Unit1.dfm ”的文件。

注意 Delphi 5新增的一项功能是IDE可以将新的DFM文件存为文本文件而不是二进制代码。这一选项是默认的，不过你可以通过在Environment Options对话框的Preferences page页中重新设置New forms as text复选框来加以修改。将表单存为文本格式，虽然会因为代码长度的增加而使执行效率略有下降，但从几个方面来说，它都不失为一个好的经验：首先，在任何文本编辑器里对DFM文本做一些小的修改都很容易实现。其次，如果 DFM文件被破坏，那么修复一个被破坏的文本文件远比修复一个二进制文件要容易得多。还要记住，前面几个版本的Delphi支持的DFM文件是二进制文件，因此如果你想创建一个可以被其他版本的 Delphi兼容的项目，你就应该让这一选项失效。

应用程序的项目文件也值得注意。项目文件的扩展名是 .DPR(代表Delphi project)，它只是一个带有不同扩展名的Pascal源文件。项目文件中有程序的主要部分。和其他版本的 Pascal不同，大多数的编程工作都是在单元文件中完成的，而不是在主模块中。可以选择主菜单下的 Project | View Source把项目源文件调入代码编辑器。下面是一个应用程序示例的项目文件：

```
program Project1;

uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

当向应用程序中添加表单和单元的时候，它们将出现在项目文件的 uses子句中。这里也要注意，在uses子句中的单元文件名之后，相应表单的名字将以注释的形式出现。如果搞不清楚哪个单元对应哪个文件，可以选择View | Project Manager来打开Project Manager(项目管理器)窗口。

注意 每个表单都对应着一个单元文件，而有的单元文件却只有代码而不对应任何表单。在Delphi中，大部分情况下你都是对单元文件进行编程，而几乎不需要编辑.DPR文件。

1.5 创建一个简单的应用程序

如果从组件选项板上选择一个按钮放到表单上， Delphi会在表单对象的声明中加入下列代码：

```
type
  TForm1 = class(TForm)
    Button1: TButton;
  private
```

```
private
  { Private declarations }
public
  { Public declarations }
ends;
```

正如你看到的，该按钮是TForm1类的一个实例变量(Button1)。以后如果要在TForm1以外的地方引用Button1，必须加上对象限定符，即Form1.Button1。第2章会详细介绍作用域的概念。

当在窗体设计器中选定了这个按钮时，可以通过 Object Inspector来改变它的行为。假设要在设计期把按钮的宽度改为100个像素，并使它在运行时能够响应鼠标单击事件而使其高度扩大一倍。要改变按钮的宽度，可以到Object Inspector中找到width属性并把它的值设为100，然后按下Enter键或把输入焦点从width属性上移走，表单上的按钮宽度即改为 100个像素。要使按钮能响应鼠标单击的事件，首先要翻到Object Inspector的Events页，找到Onclick事件，然后双击它右边的一栏， Delphi将自动生成一个响应鼠标点击的过程的框架，在这里是 TForm1.Button1click()，并把输入焦点移到这个事件响应方法的Begin和End之间。所要做的就是在Begin和End之间插入代码，以使按钮的高度扩大一倍：

```
Button1.Height := Button1.Height * 2;
```

要编译和运行这个程序，可以按下F9键。

注意 自动生成的过程与它所对应的组件之间的引用关系是由Delphi维护的。编译或保存源代码模块时，Delphi会自动检查源代码，凡是Begin与End之间没有任何输入代码的过程就认为是多余的，Delphi将把这个过程删掉。由此可见，不能随便删除一个Delphi生成的过程。即使要删除一个过程，也只需删掉你插入的代码，让Delphi去删除过程的框架。

运行了上述程序后，可以退出程序回到Delphi的IDE中。其实，要使按钮能响应鼠标单击的事件，不必使用Object Inspector，而只要在表单上双击这个按钮即可。双击一个组件将自动调用与它相关的组件编辑器。对大多数组件来说，这将会生成处理Object Inspector中列出的该组件的第一个事件的处理程序。

1.6 事件机制的优势在哪里

过去，如果用传统的方式开发 Windows应用程序，将不得不手工捕捉 Windows的消息，然后再分析这个消息，取出其中的窗口句柄、消息的 ID、WParams参数和LParam参数。如果改用Delphi，毫无疑问就简单多了。第5章“理解Windows消息”会详细介绍消息与事件的概念。

事件通常是由 Windows消息触发的。例如 TButton组件的OnMouseDown事件，实际上是由 Windows的WM_xBUTTONDOWN消息触发的。注意， OnMouseDown事件给出了诸如哪个按钮被按下、被按下时鼠标的位置等信息。一个表单的OnKeyDown事件提供了相似的有关键被按下的信息。例如下面是Delphi生成的处理OnKeyDown事件的处理程序：

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
Shift: TShiftState);
begin
end;
```

可以看出，所有需要知道的信息这里都提供了。如果你曾经用传统的方式写过 Windows应用程序，就会惊喜地发现，再也不必为分析窗口句柄、 WParams参数和LParam参数犯愁了。它不再是以你所知道的“消息流”方式，因为一个 Delphi事件能够代表几种不同的 Windows消息，就像它处理OnMouseDown(它处理了一系列鼠标消息)，每一种消息参数都是以一种容易理解的方式传递的。第 5章将深入讨论Delphi处理消息的内在机制。

无约定编程

与传统的Windows消息机制相比， Delphi的事件处理机制的最大优势在于所有的事件都是无约定

的。对程序员而言，无约定就是指可以在事件处理程序中什么都不干。而在传统的 Windows消息机制中，就不得不调用基类的消息处理程序，还要把信息回传给Windows.

当然，Delphi的事件处理机制的无约定的编程方式可能无法对消息进行直接的灵活有效的处理，你要受制于谁引发了这一事件以及应用程序对这一事件要做出什么响应等。例如，可以在 OnKeyPress处理程序中修改和取消击键，但OnResize事件处理程序却只能提供这一事件已发生的通知而无法进一步防止或控制尺寸的改变。

不过，不必担心。Delphi仍然允许在事件处理程序中直接处理 Windows消息。只是这就不像事件处理机制那么简单，因为消息处理需要程序员对要处理的消息有详细了解。可以通过 message这个关键字处理所有Windows消息。第5章将详细介绍有关消息处理的技术。

使用Delphi开发应用程序的好处是，你既可以用高级的、易于理解的方式来编程，也可以在需要时直接访问低层的信息。

1.7 加速原型化

当对Delphi使用了一段时间后，你可能会注意到，对 Delphi的学习可谓一帆风顺。事实上，即使你只是一个Delphi新手，你也会发现，用Delphi编写第一个项目就能马上获益：开发周期短、应用程序稳健。设计用户界面(UI)是令许多Windows程序员头疼的问题，而这恰好是Delphi的长处。

有时候，用户界面的设计和程序的布局被称为原型化。在非可视化的开发环境中，应用程序的原型化经常比真正实现程序的时间还要长。的确，一个简洁直观、令人愉悦的用户界面是应用程序的一大部分，但如果一个通信程序只有漂亮的窗口和对话框而没有通过调制解调器发送数据的功能，那么这个用户界面又有什么用呢？对人来说，美丽的面孔固然好看，但也必须要有实质性的东西，才能成为我们的生活中的合格一员；对应用程序来说也是一样的。

Delphi能够用它的自定义控件来快速建立漂亮的用户界面。当你能熟练运用表单、控件和事件响应方法以后，将发现过去要花很大精力的原型化工作现在大大简化了。同时，用 Delphi开发的用户界面丝毫不比用传统的开发工具建立的用户界面逊色。在 Delphi中你在设计期看到的往往就是最终的产品。

1.8 可扩展的组件和环境

由于Delphi是面向对象的，你除了可以从头开始创建你自己的组件外，还可以在 Delphi已有组件的基础上创建新组件。第21章“编写自定义组件”将介绍怎样扩展 Delphi已有组件的行为以创建新组件。另外，第7章“使用ActiveX控件”描述了怎样把ActiveX控件嵌入到你的Delphi应用程序中。

不仅可以把自定义的组件集成到IDE中，还可以把称为“专家”(Experts)的整个子程序集成到IDE中。Delphi的Expert接口允许向IDE中添加自定义的菜单项和对话框。专家的典型例子是Database Form Expert，这在Database菜单上可以找到。第26章“使用Delphi Open Tools API”将介绍怎样创建专家以及怎样把专家集成到IDE中。

1.9 IDE最重要的十点功能

为了在你继续学习这本书之前能够掌握必须的工具并知道怎样使用它们，下面列出了 IDE最重要的10点功能：

1. 类的自动生成

对开发者来说，最浪费时间的莫过于敲入所有那些代码了。常常是明明知道自己要写些什么却还要不得不受制于自己的输入速度。而Delphi提供的类自动生成(class completion)的功能极大地减轻了这一繁重的工作。

类自动生成的最重要的功能特点就是不需要当面完成工作。只需要敲入类声明的一部分，按下Ctrl+Shift+C键，类自动生成功能就将推断出你想做什么并生成正确的代码。例如，你如果把名为 Foo 的过程的声明放到你的类中并调用类自动生成功能，它将在单元的 implementation部分自动建立对这一方法的定义。声明一个从某个字段读并用一个方法写入的属性，再调用类自动生成，它将为这个字段生成代码并声明和实现这个方法。

如果你还不常用类自动生成，就多用用。你很快就会离不开它了。

2. AppBrowser导航

你有没有曾经对代码编辑器中出现的一些代码产生过疑问：“这个方法到底在哪里声明的？”按下Ctrl键并单击你想查找的标识符，你就很容易找到答案。IDE能利用编译器在后台产生的调试信息跳到该标识符的声明处。就像一个 Web浏览器一样，IDE有一个历史记录堆栈，你可以利用代码编辑器右上角的箭头前后导航。

3. interface/implementation 导航

想对一个方法的interface和implementation部分之间进行导航吗？只要把光标放在这个方法处，按下Ctrl+Shift+(或(，就可以在这两个位置间跳转。

4. 停靠

IDE允许你将多个窗口停靠。代码编辑器在左侧、右侧和底部提供了三个停靠位。你可以拖动一个窗口到另一个窗口的边框处实现边对边停靠，也可以拖到中间处进行缩进停靠。一旦你对屏幕上的窗口安排有了一个方案，一定要用 Desktops工具栏进行保存。如果要防止一个窗口停靠，只需在拖动它时按下Ctrl键，或者双击该窗口并在快捷菜单中去掉对 Dockable的选择即可。

5. 一个真正的浏览器

Delphi 1到Delphi 4使用了同一种对象浏览器，许多人从来就不用它，因为它根本没什么作用。而Delphi 5却提供了一个全新的对象浏览器！如图 1-6所示，在主菜单下选择 View | Browser就可以打开这个浏览器。这一工具以树状视图的方式为你列出了全局变量、类、单元等，共有三种分类方式（范围、继承关系、引用关系）。

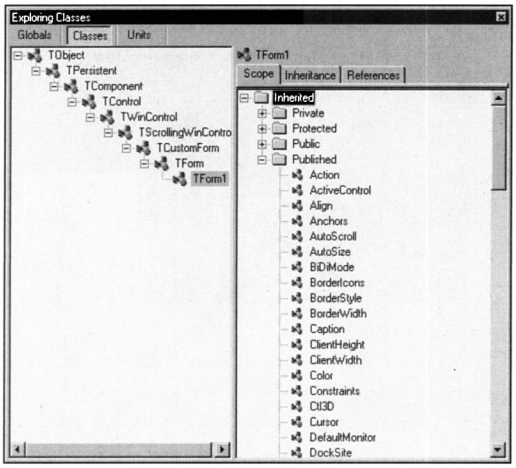

图1-6 新的对象浏览器

6. GUID

在代码编辑器中按 Ctrl+Shift+G键将会放入一个新的GUID，这会为声明新的接口节约时间。

7. 高亮显示C++语法

当你用Delphi进行工作的时候，你可能经常要阅读C++文件，如 SDK头文件等。由于 Delphi和C++Builder使用了相同的编辑器源代码，对用户而言，其中的一个好处就是 C++文件的高亮显示。在代码编辑器中加载进一个 C++文件，如.cpp或.H模块，它就会自动完成这一功能。

8. TO DO...

在你的源程序中，你可以使用To Do列表来管理程序进程。从主菜单下选择View | To Do List就可以打开To Do列表。这个表是根据你的源代码中所有以标识符 ToDo为开头的注释自动组装的。对任何一个To Do项你都可以在To Do Items窗口中设置它的ower、priority和category，如图1-7中源代码编辑器底部所示。

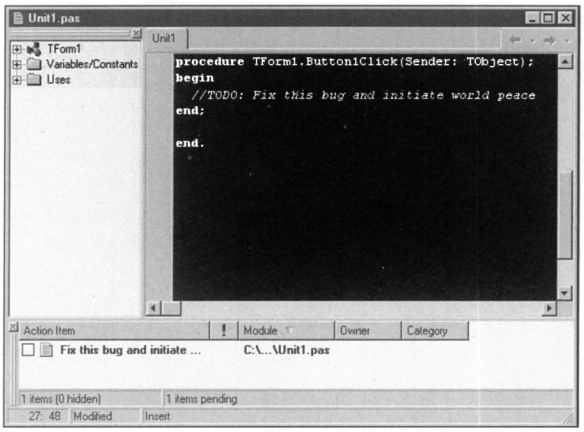

图1-7 To Do Items窗口

9. 使用项目管理器

当需要在一些大的项目(尤其是那些由多个EXE或DLL模块组成的项目)中导航时，项目管理器可以为你节省不少时间。但是许多人都忘了这一点。你可以在主菜单中选择 View|Project Manager打开项目管理器。Delphi 5为项目管理器增加了一些新的功能，比如在项目之间复制和粘贴等。

10. 使用Code Insight技术完成声明和参数说明

当键入“ 标识符 ”后，一个窗口就会自动弹出，窗口中列出了该标识符的属性、方法、事件等。可以右击这个窗口，使列表按名字或作用域排序。如果窗口过早消失，只需敲 Ctrl+空格键即可使它重新出现。

对一个函数来说，所有的参数都会编程工作带来麻烦，而 Code Insight技术在这一方面给了我们帮助：当在代码编辑器中键入函数名后，它会自动提示参数列表。如果要让提示重新出现，需按 Ctrl+Shift+空格键。

1.10 总结

现在你一定对Delphi 5的产品家族及Delphi的IDE等有所了解了。本章的目的在于让你开始熟悉Delphi和本书使用的一些概念。后面将会为真正的技术人员详细介绍 Delphi。在你进一步深入学习本书之前，你一定要确保已经掌握了IDE的用法，并且知道怎样创建一些简单的项目。