第一部分 Delphi与Object Pascal

第1章 Delphi 4集成式开发环境

- AppBrowser编辑器
- Code Insight技术
- 设计窗体
- 项目管理器
- Delphi文件
- 对象库（Object Repository）

在诸如Delphi这样的可视化编程工具中，环境的作用非常重要，有时甚至较其编程语言还为重要。Delphi 4提供了全新的可视化开发环境，本章将详细介绍它的新特性。本章并不是泛泛意义上的教程，而主要向普通Delphi用户介绍一些技巧并提供一些建议。换句话说，本章不是面向初学者的。我们将介绍Delphi4集成式开发环境（IDE）的新特性以及Delphi 3中一些高级的，鲜为人知的特性，但在本章中，我们将不会深入展开。在本书的编写中，我们始终假设读者已经具备了IDE的基本操作能力，在这样的前提下，所有章节都把重点放在了编程问题与技术上。

如果你是一个编程的初学者，也不必担心。Delphi的集成式开发环境使用起来非常直观。Delphi自己也提供了一个带有教程的手册（在Delphi光盘上用Acrobat软件可以看到），其中详细介绍了Delphi应用程序的开发。

技巧： 这是向读者介绍的第一个技巧。在Delphi 4中，用户可以通过双击Windows Explorer中的文件打开文件或项目。尽管在Delphi 3中也能做到这一点，但是该操作将打开一份新的IDE，这样会花费大量时间。而Delphi 4则是在当前IDE中打开PAS或DFM文件。当用户选择DPR文件时，Delphi会在询问是否保存之后关闭当前项目（注意：当Delphi中已经有一个打开的对话框时，该对话框会被打开的新文件所掩盖，因而不能使用Delphi的菜单命令与工具栏，除非找到并关闭该对话框）。

Delphi 4的不同版本

在研究Delphi编程环境的技术细节之前，我们首先明确两个基本概念。第一，Delphi 4有多个版本；它们分别是：

- 标准版（或叫Standard版）面向的是刚入门或不常使用Delphi的编程人员。
- 第二种版本叫Professional（专业）版，它面向的是专业程序员。它包括了基本特性，以及扩展的数据库编程支持，一些Internet支持，与一些外部工具。本书假设读者使用专业版Delphi。
- 功能最全面的版本是Client/Server Suite（客户机/服务器）版，它面向的对象是开发大型应用程序的程序员。它包括扩展的Web服务器支持，用于本地客户机/服务器BDE连接的SQL Links，三级支持，以及其它很多工具，如SQL Monitor。本书部分章节介绍的特性涉及Client/Server Suite版；谈到这些内容时会进行提示。
- 还有一些较高级的Delphi版本支持AS/400（被称为“Delphi/400”）与Entera（被称为Delphi Enterprise企业版）。本书将不涉及这些版本的特性。

除了实现不同的版本之外，还有一些定制Delphi环境的方法。在本书的屏幕图示中，我们将尽量使用一种标准用户界面；当然，还可以有其它选择，我们可以安装很多附加工具，在一些屏幕场景中可以看到它们。

AppBrowser编辑器

Delphi 4的编辑器保留了Delphi 3编辑器的所有特性，但它向应用程序源代码的读写操作添加了一种全新的性能（这也启发Borland/Inprise的市场销售人员为编辑器起了一个新名称，AppBrowser）。总的来说，它有三项重要革新：一个Code Explorer窗口（可以列出一个单元的所有定义），支持漫游功能（与Web浏览器相似），以及Class Completion（一种新的代码生成技术）。

除了这些创新之外，编辑器还与Delphi 4环境的其它部分共享停放（docking）支持。这意味着，用户可以将大多数窗口停放在编辑器的一侧。这些可以停放的窗口包括Code Explorer、Object Inspector与Project Manager。用户可以使每个窗口具有自己的特定区域（如图1.1所示），或将它们放入一个多页的查看器中。

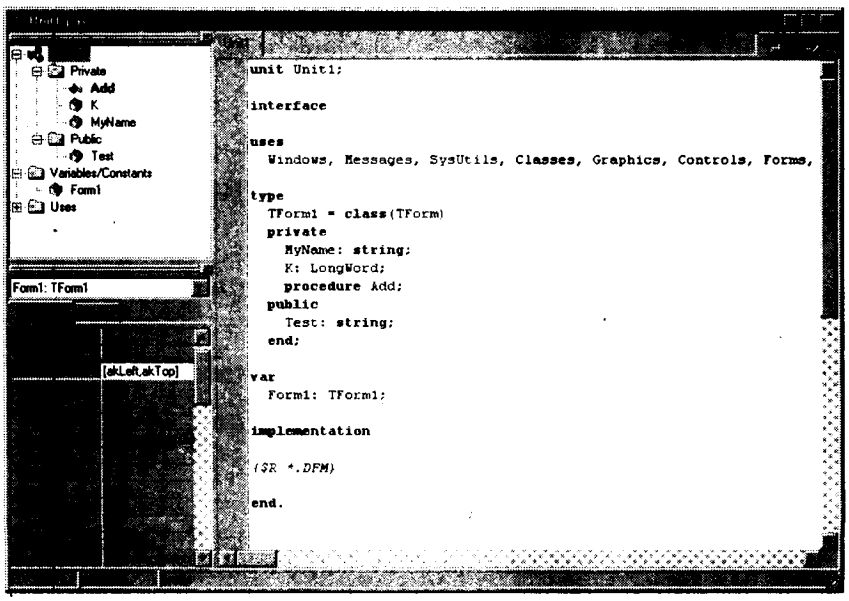

图1.1 在Delphi 4中，我们可以将很多窗口停靠在AppBrowser编辑器的一边

Delphi编辑器允许用户使用“带有标签的笔记簿”原理，一次对多个文件进行操作。通过按Ctrl+Tab 组合键，用户可以从编辑器的某一页跳到下一页（或Shift+Ctrl+Tab组合键向相反方向移动）。有一些影响编辑器的环境选项，大部分位于Environment Options对话框的Editor Options、Editor Display与Editor Colors页中。然而，用户必须翻到Preferences页，来设置编辑器的AutoSave特性，这样，每当运行程序时，源代码文件都会自动被保存。

用户还可以使用Windows Registry（注册表）中两个未公开的项来设置编辑器的初始宽度与高度（例如，使编辑器与屏幕一样大小）。找到Registry键HKEY_CURRENTUSER/Software/Borland/Delphi/4.0/Editor，并添加两个新DWOED项，DefaultHeight与DefaultWidth，用象素表示编辑器的高度与宽度。为了改动Windows Registry，读者可以使用Windows 95与98中的RegEdit.EXE应用程序或NT中的RegEdt32.EXE应用程序。

技巧： 除了使用剪切与粘贴命令外，Delphi 4编辑器与Delphi 3编辑器一样，允许用户通过选择与拖放代码的字符、表达式或整行代码来移动源代码。用户还可以通过在拖动的同时按Ctrl键来复制文本。

Code Explorer

Code Explorer窗口（当停放在编辑器一侧时，通常用处最大）只是简单地列出了一个单元中定义的类型、变量、子例程以及单元。对于复杂的类型，如类，Code Explorer可以列出详细的信息，包括元素、属性与对象方法。只要用户在编辑器中键入代码，所有的信息都会被更新。用户可以使用Code Explorer在编辑器中漫游。如果双击Code Explorer中的一个条目，编辑器会跳到相应的声明处。

只经简单的操作，Delphi 4的所有这些特性都会显得非常明显，但Code Explorer还有一些特性却不是那么直观。重要的是，用户应该完全掌握信息的布局，并通过定制Code Explorer来减小该窗口中信息显示的复杂程度。这样可以帮助用户加快选择的速度。读者可以使用Environment Opitons的相应页来配置Code Explorer，如图1.2所示。

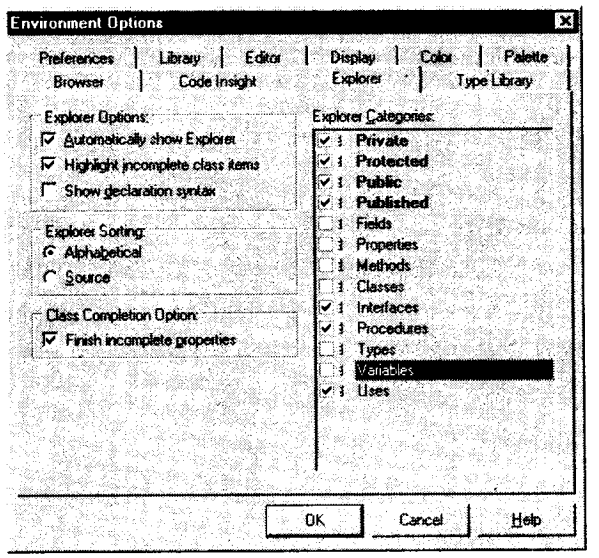

图1.2 Code Explorer的配置可以在Environment Options对话框中

注意，当删除对话框该页右边的一个Explorer Categories条目时，Explorer将不会从视图框中删除相应项，只会在树状结构中添加一个节点。例如，如果我们删除Uses复选框，Delphi不会将被使用单元的列表从Code Explorer中消除。相反，被使用单元会被列作主节点，代替先前列在Uses文件夹中的情况。还有一个例子，删除Types、Classes、Variables等选项，可以得到图1.3所示的输出结果。

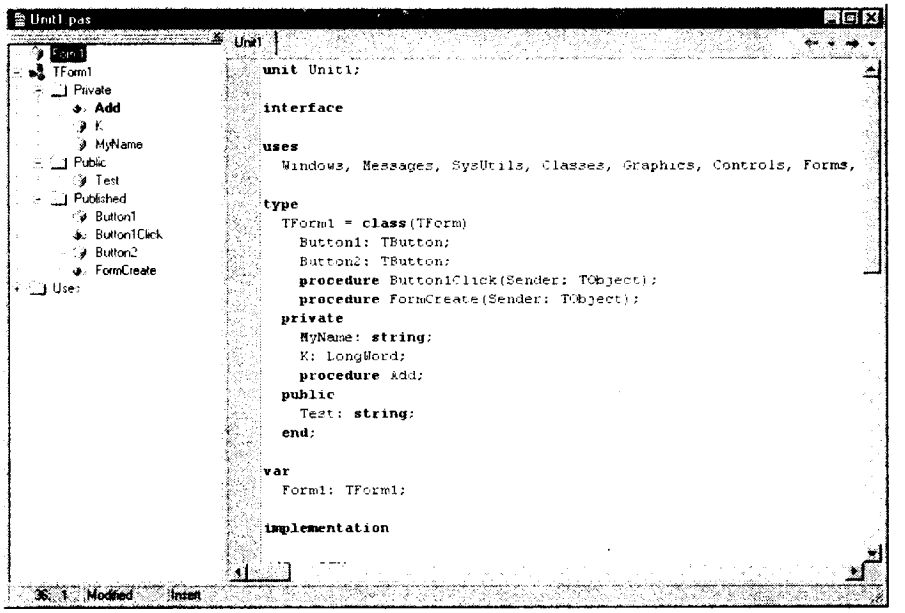

图1.3 CODE EXPLORER中一些文件夹的删除操作可以通过从相应设置（如图1.2所示）中删除选项来实现

最重要的设置可能与类有关。与类有关的定义可以用三种方式准备：

- 根据专用的、受保护的、公共的、发布的种类
- 根据对象方法与元素的种类
- 所有的集中在一组中

Code Explorer树型结构的每一项都有一个图标标志其类型，所以安装对象方法与元素并不象安排访问标志符那样重要。我们的建议是，将所有条目显示在一组中，这样，选择的速度最快。事实上，在Code Explorer中选择条目，为大型单元中源代码的漫游提供了一种非常方便的方法。当用户双击Code Explorer中的一个对象方法时，光标会移动到类声明（在单元的接口部分中）模块的定义处。用户可以在Delphi 4中使用Ctrl+Shift组合↑或↓箭头键，从单元接口部分中的对象方法或过程定义处跳到其在实现部分中的完整定义处（或返回）。

在Code Explorer中进行编辑

Code Explorer不只是一个浏览与输出工具。事实上，我们可以使用它在每一类中输入新条目，如图1.4所示。新条目的类型实际上通常要根据键入的内容而定：以过程或函数关键字开头的名称会自动被认为是对象方法，而带着分号与数据类型的名称则会被认为是元素。

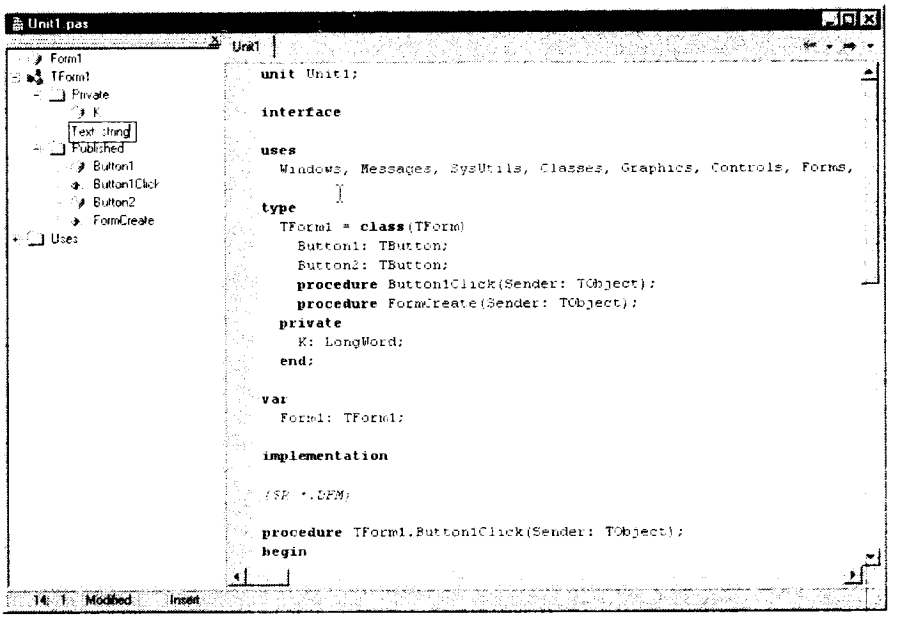

图1.4 CODE EXPLORER可以用于添加新元素与对象方法，并改动它们

注意： 元素、对象方法、公用、专用…？如果读者不熟悉Object Pascal语言的术语，可以阅读第3章的有关介绍。我们在这里使用它们，而没有解释它们，是因为本书的大部分读者可能对Delphi及其编程语言已经有了一定的了解。

Code Explorer的编辑能力十分有限，所以与通常的编辑方式（在源代码窗口中编辑）相比，它不能提供真正优秀的功能。具有拖动功能还是很不错的，例如，可以将元素或对象方法移到另一个部分或复制到另一个类中。

在编辑器中浏览

AppBrowser编辑器的另一个新特性是“符号提示框”。将鼠标移到编辑器中的一个符号上，提示框将显示标志符声明的位置。如图1.5所示。该特性对于跟踪正编写的应用程序中的标志符、类以及函数，还有引用可视组件库（VCL）的源代码是非常重要的。

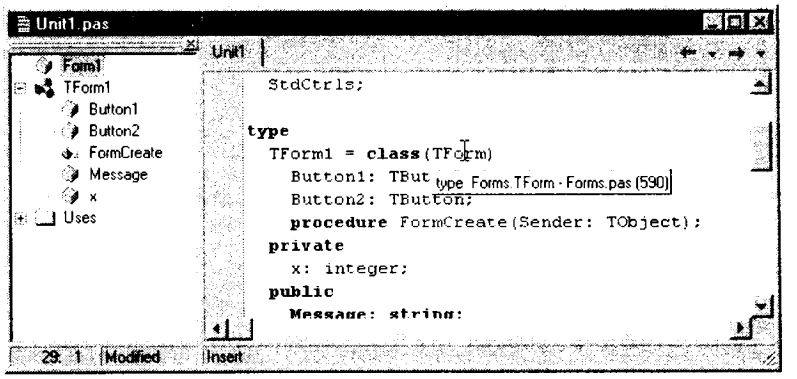

图1.5 Delphi 4的“符号提示框”新特性是编辑器浏览功能的开端

警告： 尽管看起来象个好注意，但实际上，我们不能使用提示框来查找声明某标志符的单元。事实上，如果没有包括相应的单元，提示框将不会出现。

然而，该特性真正的优点在于，我们可以利用它实现漫游功能。事实上，当用户按着Ctrl键将鼠标移到标志符上时，Delphi会自动连接定义来代替提示框显示。编辑器会用蓝色带下划线字符形式来表示链接，这是Web浏览器的标准性质，每当鼠标位于链接之上时，鼠标形状都会变成手形，如图1.6所示。

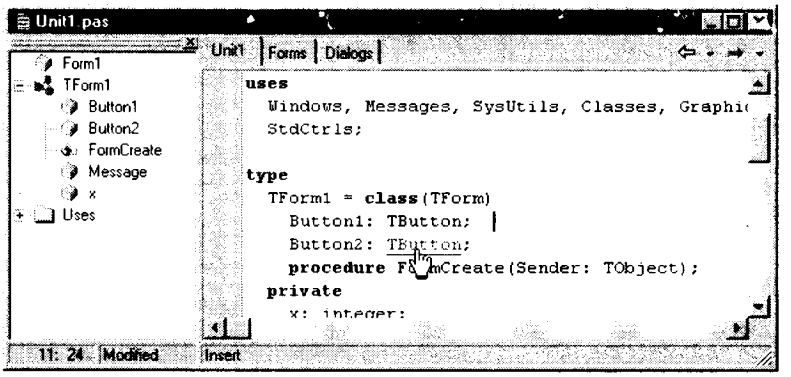

图1.6 Delphi 4的浏览功能需要通过按着CTRL键并将鼠标移到一个标志符上来激活

Environment Options对话框的Code Insight页允许我们激活或禁用这些技术，并设置延迟时间。

例如，用户可以Ctrl+单击TForm标志符，来打开其在VCL源代码中的定义。当用户选择引用时，编辑器会跟踪用户到达的不同位置，而且可以在这些位置中前后移动——这也与Web浏览器一样。为了更好地控制前后移动，用户还可以单击Back或Forward按钮旁边的下箭头来查看源代码文件当前行的详细清单。图1.7显示了该特性的一个例子。

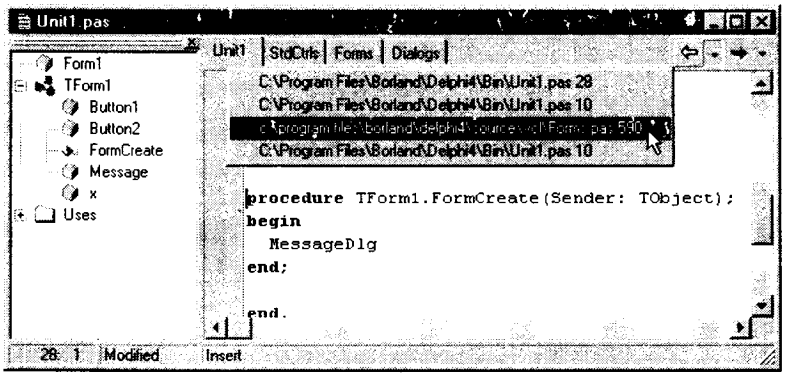

图1.7 AppBrowser编辑器可以追踪访问过的源代码文件

如果VCL源代码不是项目的组成部分，那么如何直接跳到该代码中呢？AppBrowser编辑器不仅可以在Search路径中查找单元，还可以在Delphi的Debug Source、Browsing与Library路径中进行查找。按照顺序查找这些目录，并可以在Project Options对话框的Directiories/Conditionals页中以及Environment Options对话框的Library页中设置它们。缺省情况下，Delphi会在Browsing路径中添加VCL源代码目录，其声明如下：

```
$(DELPHI)\source\vcl;$(DELPHI)\source\rtl\Corba;
$(DELPHI)\source\rtl\Sys;$(DELPHI)\source\rtl\Win;
$(DELPHI)\source\Internet
```

在这些路径中，声明$（DELPHI）表示Delphi安装的目录。

技巧：为了改变Delphi 4环境中的路径，用户可以使用编辑器，单击编辑框或组合框（显示路径）旁边的椭圆形按钮激活该编辑器。该路径编辑器会逐行列出每个路径，使得输入或删除它们变得更容易了。

Class Complete

Delphi的AppBrowser编辑器中新添的第三个特性是Class Complete，可以按下Ctrl+Shift+C组合键来激活它。向应用程序添加一个事件处理程序是很快的操作，Delphi会在类中自动添加一个新的对象方法声明来处理事件，并在单元的实现部分为用户提供对象方法的框架。这是Delphi对可视化编程提供支持的一部分。

Delphi 4还用了一种相似的方法，为需要在事件处理程序后面编写少许附加代码的编程人员提供了方便。事实上，新的代码生成特性涉及常用的对象方法、消息处理对象方法以及属性。例如，如果用户在类声明中键入下列代码：

```
public
  procedure Hello (MessageText: string);
```

然后按Ctrl+Shift+C组合键，Delphi将在单元的实现部分提供对象方法定义，于是生成下列代码：

```
{ TForm1 }

procedure TForm1.Hello(MessageText: string);
begin

end;
```

与很多Delphi编程人员使用的传统方法（复制并粘贴一个或多个声明，添加类名称，最后还要为每个对象方法复制begin ..end代码）相比，这确实非常方便。

Class Complete还能以其它方式使用。用户可以直接编写对象方法的代码，并按Ctrl+Shift+C，在类声明中生成所需的条目。

回头看一下图1.2所显示的Explorer设置，读者会看到一个用于Class Complete的选项——用户可以使用它完成某个属性的定义。如果在一个新窗体类中简单输入，

```
property X: Integer;
```

并激活Class Complete功能，Delphi会为属性生成一个SetX对象方法，并向类中添加FX元素。代码变为：

```
type
  TForm1 = class(TForm)
  private
    FX: Integer;
```

```
    procedure SetX(const Value: Integer);
  public
    property X: Integer read FX write SetX;
  end;

implementation

procedure TForm1.SetX(const Value: Integer);
begin
  FX := Value;
end;
```

这节省了大量的输入劳动。实际上，用户甚至可以部分地控制Class Complete的工作方式，为属性生成Set与Get对象方法，在第4章与属性有关的部分中将讨论该问题。

Code Insight

除了Code Explorer、Code Completion以及漫游特性外，Delphi 4仍支持并部分改进了原来由Delphi 3引入的Code Insight技术。总得说来，这些Code Insight技术是基于对背景的分析，一方面针对用户编写的源代码，另一方面针对这些源代码引用的系统单元的源代码。Code Insight包括五项功能：

- **Code Completion** 允许用户查看一个列表，或键入对象的首字母来选择对象的属性和过程。为了激活它，我们只需键入一个对象的名称，如Button1，然后添加圆点，并等待即可。要强行显示列表，可以按Ctrl+空格键，当不想显示它时，按Esc键就会清除它。Code Completion还允许用户在赋值语句中查寻相应的值。当我们在一个变量或属性后键入:=时，Delphi将列出该类型的所有变量或对象以及具有该类型属性的对象。
- **Code Template** 允许用户插入事先定义的代码模板，如带有一个内部“begin-end”块的复杂语句。Code Templates必须人工激活，使用Ctrl+J组合键显示所有模板的列表。如果我们在按Ctrl+J前键入几个字母（作为关键字），Delphi将只列出从这些字母起始模板。查看下页阴影中“定制代码模板”的说明，可以知道向哪些Delphi提供的模板中添加新模板的方法。
- **Code Parameters** 在一个提示窗口中显示函数或过程参数的数据类型，这是在用户键入它们的时候。只需键入函数或对象方法名称与左括号，参数名与类型就会立刻显示在一个弹出式提示窗口上。强行显示Code Parameters，可以使用Ctrl+Shift+空格键。作为进一步的帮助，当前参数会用黑体显示。Delphi 4中的Code Parameters支持缺省参数与重载的对象方法，这将在第2章中介绍。
- **Tooltip Expression Evaluation** 是一个调试时使用的特性。它用来显示光标下标志符、属性或表达式的值。我们将在第21章重谈这个话题。
- **Tooltip Symbol Insight** 可以在提示框中显示标志符的定义，这已经讨论过了（“在编辑器中浏览”一节开始处），它是Delphi 4新添的功能。

可以在Environment Options对话框的Code Insight页中，选择与关闭或配置这些功能，如图1.8所示。

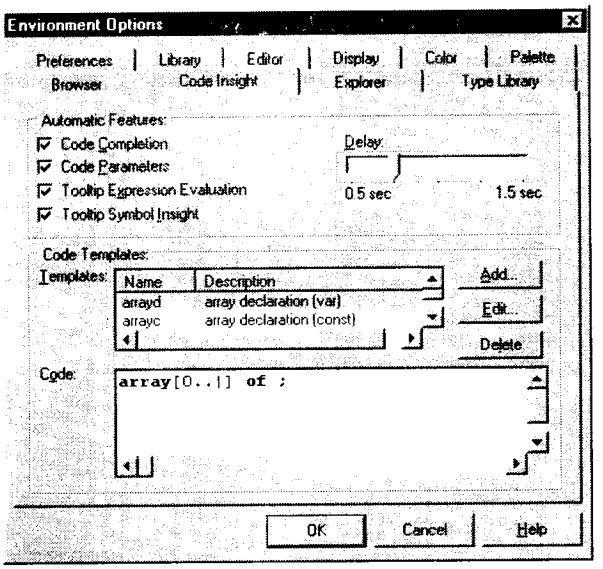

图1.8 Environment Options对话框的Code Insight页允许我们激活或禁用这些技术并设置延迟时间

定制代码模板

我们可以使用代码模板为常用表达式或函数调用赋名。例如，如果我们经常使用MessageDlg函数，可以为它添加一个模板。移到Environment Options对话框的Code Insight页上，在Code Template区中单击Add按钮，键入新模板名（例如，mess），输入说明，然后向Code备注控件中的模板中添加下列文本：

```
MessageDlg ('',
  mtInformation, [mbOK], 0);
```

现在，每当我们需要建立一个消息对话框时，只需输入mess，然后按Ctrl+J键，就可以获得完整的文本。竖线符说明了在展开模板后，光标在编辑器上源代码中的位置。

Code Templates与语言关键字没有直接对应关系；它们是一种更通用的机制。Code Templates保存在DELPHI32.DCI文件中，所以可以在不同的计算机之间复制模板。合并两个Code Template文件看来不太容易，也没有第三方工具用于添加更多的模板。这些增强需要等到Borland/Inprise将.DCI文件的内部结构公开的时候才能实现。

使用编辑器书签

与Delphi 3的编辑器相比，Delphi 4的AppBrowser编辑器为书签提供了一种更好的支持。当光标位于编辑器的某行上时，用户可以使用弹出菜单的Toggle Bookmarks命令设置或删除书签，并在二级菜单中选择书签号。

技巧： Delphi的编辑器有与位置相关的菜单（有时被称作“背景菜单”），用户可以在AppBrowser的窗口上单击鼠标右键激活它。作为典型的Windows弹出菜单，它包括大多数与编辑器自身有关的常用命令。读者自己可以在每个Delphi窗口中试着单击鼠标右键看一看。

为了加快该操作的速度，用户可以使用Ctrl+Shift与从0到9的数字键组合。书签会出现在编辑器一旁狭窄的“装订线”空白区，如图1.9所示。为了跳回书签，用户可以使用弹出

菜单的Goto Bookmarks命令或按Ctrl键与数字键的组合键。

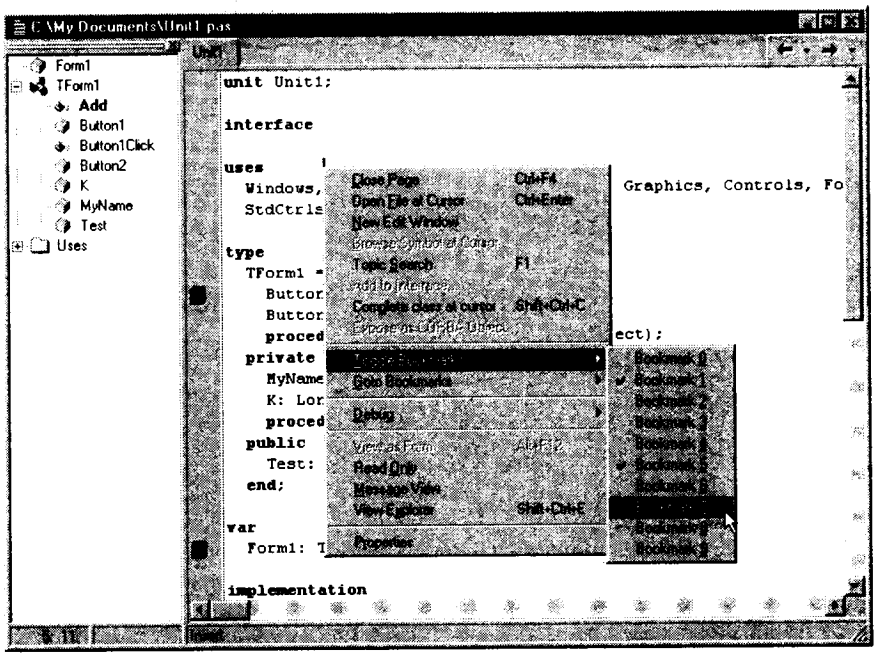

图1.9 Delphi编辑器装订线内的书签，及与位置相关的背景菜单

当用户编写长文件并同时编辑多个对象方法，或想从类定义跳到其对象方法的定义时，书签显得非常有用。然而，书签的使用还存在着两个问题。第一问题，如果在没有察觉的情况下赋给新书签已有的书签号，老书签将被删除。第二个问题，也更严重的一个问题，书签是不能保留的；也就是说，它们不能与文件一起保存下来。

技巧：一些第三方的Delphi IDE扩展，包括Eagle Software的CodeRush提供了可以保留/命名的书签。

其它编辑器快捷键

编辑器还有其它很多快捷键，它们的使用要根据用户选择的编辑器风格而定。下面是一些大家不太了解的快捷键（大部分都很有用）：

- Ctrl+E用于连续的查寻：用户可以使用Ctrl+E，然后直接键入想查寻的代码，而不需要打开某个特定的对话框并按Enter键来查找。
- Ctrl+Shift+I可以一次缩进多行代码。缩进的空格数要在Environment Options对话框的Editor页由Block Indent选项设定。Ctrl+Shift+U是用于执行反操作的相应组合键。
- Ctrl+O+U用于转换被选代码的大小写；或使用Ctrl+K+E将所选代码转换为小写；使用Ctrl+K+F转换为大写。
- Ctrl+Shift+R用于开始记录一个宏，之后可以使用Ctrl+Shift+P快捷键执行该宏。宏记录了在源代码中进行的所有输入、移动以及删除操作。执行宏可以简单地重复所记录的操作，一旦换了一个不同的源代码文件，该功能就显得没什么意义了。尽管我们猜测只是为了测试目的使用了该快捷键，但我们还是需要为该技术找到用处。
- 按下Alt键，用户可以拖动鼠标选择编辑器的矩形区域，而不只是连续的代码行与代码。

窗体设计器

如果说编辑器是我们使用非常频繁的Delphi窗口之一的话，那么另一个就一定是窗体设计器了。在Delphi 4中，窗体设计器并没有太多的改进。在Delphi新版本中，窗体设计器只是新增了一些工具提示：

- 当用户将鼠标移到一个组件上时，提示框会显示组件的名称与类型。当然，使用Show Component Captions环境设置也可以做到这一点。
- 当用户改变一个控件的大小时，提示框会显示当前的大小（Width与Height属性）。当然，该特性只适用于控件，而不适用于非可视化组件（在窗体设计器中用图标表示）。
- 当用户移动一个组件时，提示框会说明当前位置（Left与Top属性）。读者可以在图1.10中看到该组件提示的例子。

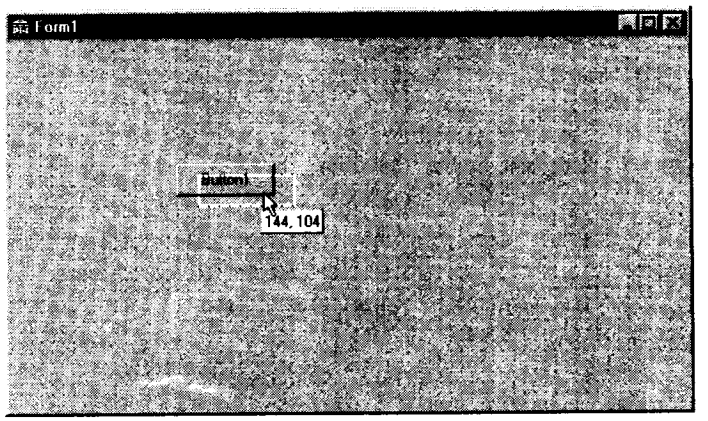

图1.10 当我们在窗体设计器中移动一个组件时，显示的提示框

在窗体设计器中，用户可以使用鼠标或通过Object Inspector（当控件隐藏在另一个控件之后或非常小时，使用它非常方便）直接选择某个组件。如果一个控件完全覆盖了另一个控件，可以使用Esc键选择当前控件的父控件。还可以按一次或多次Esc键来选择窗体，或按住Shift键同时单击所选的组件，这样也会取消组件的选定并缺省地选择窗体。

有两种方法可以代替使用鼠标来设置组件的位置：用户也可以设置Left与Top属性的值，或可以在按住Ctrl键的同时使用箭头键。使用箭头键在微调组件位置时格外有用（Snap to Grid选项只适用于鼠标操作）。同样，在按住Shift键的同时使用箭头键，可以微调组件的大小（如果使用Shift+Ctrl+箭头的组合键，将只能以网格间隔移动组件）。遗憾的是，在这些微调操作期间，不会出现组件提示框。

为了对齐多个组件或使它们大小一致，用户可以选定多个组件并同时为它们设置Top、Left、Width或Height属性。要选择多个组件，用户可以在按下Shift键的同时单击它们。或如果所有组件在一个矩形区域内，可以拖动鼠标“画”一个矩形围住它们。当选择了多个组件之后，还可以使用Alignment对话框（与窗体弹出菜单的Align命令）或Alignment面板（使用View菜单下的Alignment Palette菜单命令打开）设置它们的相对位置。

在完成一个窗体的设计之后，我们可以使用Edit菜单的Lock Controls命令来避免出现组件在窗体中以外的位置改变。当窗体上没有真正的Undo操作时，该命令对我们很有帮助，但该设置不能保存。

技巧：窗体的弹出菜单新添了一个命令——Flip Children，并含两个子命令。该命令是对书写方式从右向左的语种的支持的一部分；它会将控件从窗体左端翻到右端的镜向位置。

组件编辑器

针对某些组件的弹出菜单会显示一些特别命令。这些菜单命令是由组件编辑器添加的，具体操作我们将在第15章中介绍。表1.1列出了当某些组件被选定时向窗体弹出菜单添加的命令（在该表中，我们省略了TeeChart，Quick Report，Internet，Decision Cube与MIDAS等组件的命令）。在某些情况下，这些行为也是组件的缺省行为，当我们在组件设计器中双击组件时会出现的这些行为。

表1.1 窗体设计器弹出菜单中的特别命令

菜单命令	组件
Menu Designer	MainMenu, PopupMenu
Action List Editor	ActionList
Input Mask Editor, Masked Text Editor	MaskEdit
SQL Builder	Query
Fields Editor	Table, Query, StoredProc, ClientDataSet, NestedTable
Explore	Table, Query, StoredProc, Database
Create Table, Update Table Definition, Delete Table, Rename Table	Table（根据其状态）
Define Parameters	Query, StoredProc
Database Editor	Database
Assign Local Data	ClientDataSet
Update SQL Editor	UpdateSQL
Execute	BatchMove
Panels Editor	StatusBar
Sections Editor	HeaderControl
Bands Editor	CoolBar
Columns Editor	DBGrid, ListView
Edit Report	Report
ImageList Editor	ImageList
New Page, Next Page, Previous Page	Page控件
Items Editor	ListView, TreeView
Next Frame, Previous Frame	Animate
Insert Object, Paste Special	OleContainer
New Button, New Separator	ToolBar
Properties, About	所有ActiveX控件
Action Editor	WebDispatcher
ResponseEditor	QueryTableProducer, DataSetTableProducer

组件面板

组件面板的使用非常简单，但是有些细节大家可能还不知道。在窗体上放置组件，有四种简单的方法：

- 在组件面板上选择一种控件后，单击窗体设置控件的位置，并用鼠标调整控件的大小。
- 在任意选择的一种组件后，单击窗体放置组件，使用缺省的高度与宽度。
- 双击组件面板上的图标在窗体中央添加该类型的组件。
- Shift-单击组件图标，可以在窗体上放置多个相同种类的组件。终止该操作，只需单击组件面板左端的标准选择符（箭头图标）即可。

用户可以在组件面板的弹出菜单上选择Properties命令，在不同页中重新设置组件，添加新元素或只是将它们从某页移到另一页。在对话框的该页中，用户可以简单地将一个组件从Components列表框中拖到Pages列表框中，来实现不同页之间组件的移动。

技巧： 当组件面板上有太多的页时，用户需要翻动它们来寻找一个组件。在这种情况下，有一个小技巧可以使用：用较短的名称为页重新命名，这样所有页都可以显示在屏幕上了。显然...这个技巧你一定也想到了。

组件面板真正未公开的特性是“hot-track”。通过设置Registry的特殊键，用户可以简单地通过页标的移动来选择组件面板的页，而不需要任何鼠标单击。同样的特性还可以应用到组件面板两端的组件滚动钮上，当组件太多时会显示滚动钮。

为了使用这个隐含的特性，我们必须在HKEY_CURRENT_USER\Software\Borland\Delphi\4.0下添加Extras关键字。在该关键字下，我们还必须输入两个字符串值，AutoPaletteSelect和AutoPaletteScroll，并且将每个值都设置为字符串“1”。

定义事件处理程序

多种技术，用户可以用于为组件的某个事件定义处理程序：

- 选择组件，移到Events页上，双击事件右端的空白区，或在该区内输入一个名称并按Enter键。
- 对于很多控件来说，用户可以双击它们来执行缺省操作，该行为会为OnClick、OnChange或OnCreate事件添加处理程序。

当用户想删除某个事件处理程序（已经写入了一个Delphi应用程序的源代码中）时，可以删除所有对它的引用。然而，更好的方法是从相应的过程中删除所有代码，只留下声明与begin与end关键字。当用户首先决定处理事件时，文本应该与Delphi自动生成的文本相同。当用户保存或编译项目时，Delphi会从源代码与窗体描述（包括在Object Inspector的Events页中对它们的引用）中删除所有的空对象方法。反过来，为了保留空事件处理程序，可以考虑向它添加一条注释（甚至只是简单的//字符），这样就不会删除它了。

复制与粘贴组件

窗体设计器有一个有趣的特性，那就是从一个窗体向另一个窗体或在窗体内复制/粘贴组件的特性。在该操作期间，Delphi会复制所有的属性并保留相连的事件处理程序，而且偶尔会改动控件的名称（在每个窗体内控件名称必须是唯一的）。

窗体设计器与编辑器之间的组件复制也是可能的。事实上，Delphi是将组件与其文本描述一起放在剪贴板中的。用户甚至可以编辑组件的文本版本，向剪贴板复制文本，然后将它作为新组件粘贴回窗体。例如，如果用户在窗体上放置了一个按钮，复制它，然后将它粘贴到编辑器（可以是Delphi自己的源代码编辑器或其它字处理器）中，会得到以下描述：

```
object Button1: TButton
  Left = 152
  Top = 104
  Width = 75
  Height = 25
  Caption = 'Button1'
  TabOrder = 0
end
```

现在，如果改动对象的名称，标题或位置，或添加新属性，这些改变可以被复制并粘贴回窗体。下面是改动的例子：

```
object Button1: TButton
  Left = 152
  Top = 104
  Width = 75
  Height = 25
  Caption = 'My Button'
  TabOrder = 0
  Font.Name = 'Arial'
end
```

复制上面的描述并将其粘贴到窗体中，会在指定位置建立一个标题为My Button（Arial字体）的按钮。

为了使用该技术，需要知道编辑组件文本描述的方法，对于组件可以使用的属性，为字符串属性编写方法，设置属性与其它特殊属性。当Delphi解释某个组件或窗体的文本描述时，它还可能改动其它与已改变的组件有关的属性值，而且它可能会改动组件的位置，这样它就不会与前一次复制重叠了。用户可以通过将组件复制回编辑器来看看Delphi时如何改动组件属性的。例如，如果将上面的文本粘贴到窗体中，然后再复制回编辑器，将会得到的结果：

```
object Button2: TButton
  Left = 152
  Top = 160
  Width = 75
  Height = 25
  Caption = 'My Button'
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'Arial'
```

```
    Font.Style = []
    ParentFont = False
    TabOrder = 1
  end
```

可以看到，自动添加了几行代码，来确定字体的其它属性并改正Tab Order，以避免出现重复的值。当然，如果用户写入的内容出现了错误，并将它粘贴到窗体中，Delphi会显示一条错误消息说明错误的所在。

还可以一次选择多个组件并复制它们，将它们粘贴到另一个窗体或文本编辑器中。当需要对一系列相似组件进行操作时，这会显得很有用。用户可以将一个组件复制到编辑器中，然后重复若干次，进行相应的改动，然后再将整组组件复制到窗体中。

复制组件模板

当用户从一个窗体向另一个窗体复制一个或多个组件时，需要复制它们的所有属性。Delphi 3还引入了一个功能更强的方法，就是建立组件模板，它复制了一份属性与事件处理程序的源代码。当用户将组件模板粘贴到新窗体中时，通过从组件面板上选择伪组件（Pseudo-component），Delphi会在新窗体中复制事件处理程序的源代码。

为了建立组件模板，需要选择一个或多个组件，并使用Component菜单下的Create Component Template命令。该命令会打开Component Template Information对话框（见图1.11），用户需要在对话框中输入模板名称、显示模板的组件面板页以及图标。

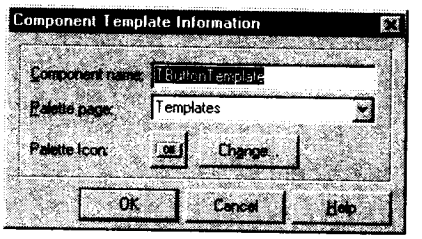

图1.11 组件模板信息对话框

缺省情况下，组件模板的名称是在字Template后加上所选第一个组件的名称。缺省的模板图标是所选第一个组件的图标，但用户可以用一个图标文件放置图标。用户为组件模板所起的名称将用于在组件面板上描述它（当Delphi显示弹出式提示框时）。

有关组件模板的所有信息都保存在一个单独文件（DELPHI32.DCT）中，但是显然无法检索该信息并编辑模板。然而，我们可以将组件模板放置在一个新窗体中，编辑它，并使用相同名称重新作为组件模板安装它。使用该方法可以覆盖以前的定义。

技巧： 将组件模板放在一个公共目录中，在Registry中向Software\Borland\Delphi\4.0\Component Templates键中添加CCLibXir项，多个Delphi程序员可以共享它们。

项目管理

Delphi 4 IDE的新特性之一是，多目标项目管理器，用户可以使用View菜单下的Project

Manager命令，打开该项目管理器并可以将它随意停靠在其它窗口中。事实上，新的项目管理器是对一个项目组进行操作，项目组中可以有一个或多个项目。例如，用户可以拥有包括一个DLL和一个可执行文件的项目组。

从图1.12中可以看出，项目管理器显示为树型结构，它显示了项目组的层次结构、项目以及组成每个项目的所有窗体和单元。用户可以使用简单的工具栏与项目管理器中更复杂的弹出菜单对它进行操作。注意，弹出菜单会根据所选对象的不同发生变化：有用于向项目组添加新项目或已有项目的菜单命令，有用于编译或建立特殊项目的命令，以及用于打开单元的命令。

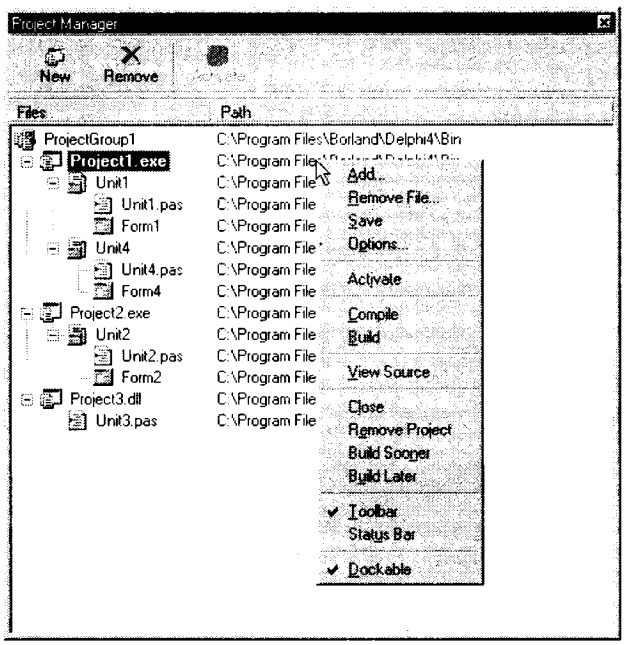

图1.12 Delphi 4的项目多目标管理器

项目组的所有项目中，只有一个处于活动状态，而且是用户在选择如Project菜单下的Compile这样的命令时所操作的项目。然而，Project菜单中添加了新命令，用户可以用于编译或建立项目组的所有项目（非常奇怪，在项目管理器的弹出菜单中却找不到这些用于项目组的命令）。当需要建立多个项目时，可以使用Build Sooner或Build Later命令设置相对顺序。

Delphi使用新的扩展名.BPG（代表Borland Project Group）来保存项目组。该特性来源于C++ Builder与过去的Borland C++编译器，而且当我们打开一个项目组的源代码文件时会清晰的看到这一点，这基本上是C/C++开发环境中makefile的源代码。下面是一个简单的例子：

```
#------------------------------
VERSION = BWS.01
#------------------------------
!ifndef ROOT
```