软件工程师参考手册

Delphi 7
组件编程参考手册

本书编写组　编著

附光盘
CD-ROM

人民邮电出版社
POSTS & TELECOM PRESS

软件工程师参考手册

Delphi 7
组件编程参考手册

⊙ 本书编写组　编著

本书配有光盘，需要的读者请到多媒体阅览室（新馆 301 室）联系。

人民邮电出版社

图书在版编目（CIP）数据

Delphi 7 组件编程参考手册/《Delphi 7 组件编程参考手册》编写组编.
—北京：人民邮电出版社，2003.11
ISBN 7-115-11707-1

I. D… II. D… III. 软件工具—程序设计—技术手册 IV. TP311.56-62

中国版本图书馆 CIP 数据核字（2003）第 098125 号

内容提要

本书是关于 Delphi 7 组件编程的一本十分完整的参考手册。本书涵盖了 Delphi 7 集成开发环境的使用、介绍包括基本窗口组件、列表框组件、数据库组件等百余种组件的功能、属性、事件以及方法。

书中大部分组件都辅以了合适的示例。这些示例力求短小精练、界面友好、可读性强，将对应的技术手段应用于实际，有助于读者理解吸收，同时也可供模仿和直接使用。

本书是从事 Delphi 应用程序开发和应用人员必备参考书，也可作为大专院校相关专业师生自学、教学参考用书。

软件工程师参考手册

Delphi 7 组件编程参考手册

◆ 编　　著　本书编写组
责任编辑　张立科

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号
邮编　100061　电子函件　315@ptpress.com.cn
网址　http://www. ptpress.com.cn
读者热线　010-67132692
北京汉魂图文设计有限公司制作
北京鸿佳印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：53.5
字数：1 675 千字　　2003 年 11 月第 1 版
印数：1-4 000 册　　2003 年 11 月北京第 1 次印刷

ISBN7-115-11707-1/TP・3628

定价：88.00 元（附光盘）

前　言

Delphi 7 是一种易学习、功能强、效率高的面向对象编程工具，便于快速掌握使用。目前它拥有着相当数量的用户群。本书是针对广大的 Delphi 编程人员编写的一本 Delphi 组件参考手册。它对于不同层次的 Delphi 编程人员来说都极具参考价值，是一本不可多得的参考书。

本书内容全面、讲解细致，体现“实用”原则。大量选自实际系统的示例与各项技术紧密结合，便于读者理解和掌握技术的实际应用；结构安排合理，按功能用途和复杂程度划分章节，目录上体现技术/知识的功能，既便于读者快速定位所需技术（非常适于读者已知功能要求，而不知道该选择什么组件的情况），又适于逐步学习。

本书按类别对 Delphi 7 的主要组件进行介绍，主要内容如下：

第 1 章　对 Delphi 的组件库的继承关系和优势做概括性的介绍，同时介绍 Delphi 组件的运行机制。

第 2 章　详细介绍了 Delphi 7 的集成开发环境以及常用的属性、方法和事件。

第 3 章　主要介绍了窗口类组件。包括：表头组件（THeaderControl）、状态栏组件（TStatusBar）、工具栏组件（TToolBar）、酷栏组件（TCoolBar）。

第 4 章　主要介绍了菜单类组件。包括：主菜单（TMainMenu）、快捷菜单（TPopupMenu）。

第 5 章　主要介绍了按钮类组件。包括：命令按钮（TButton）、位图按钮（TBitBtn）、快捷按钮（TSpeedButton）。

第 6 章　主要介绍了文本类显示组件。包括：标签（TLabel）、静态文本（TStaticText）。

第 7 章　主要介绍了文本类编辑组件。包括：单行文本编辑框（TEdit）、多行文本编辑框（TMemo）、格式输入编辑框（TMaskEdit）、RTF 编辑框（TRichEdit）。

第 8 章　主要介绍了列表类组件。包括：列表框（TListBox）、动作列表（TActionList）、图像列表（TImageList）、树状视图（TTreeView）、列表视图（TListView）。

第 9 章　主要介绍了选择类组件。包括：组合框（TComboBox）、分组框（TGroupBox）、单选分组框（TRadioGroup）、单选框（TRadioButton）、复选框（TCheckBox）、带复选框的列表框（TCheckListBox）。

第 10 章　主要介绍了分组类组件。包括：Tab 组件（TTabControl）、多页组件（TPageControl）、Tab 表组件（TTabSheet）。

第 11 章　主要介绍了对话框类组件。包括公共对话框（TCommonDialog）、“打开”对话框组件（TOpenDialog）、“另存为”对话框组件（TSaveDialog）、能预览图像的“打开”对话框组件（TOpenPictureDialog）、“字体”对话框（TFontDialog）、“颜色”对话框（TColorDialog）、“打印”对话框（TPrintDialog）、“打印设置”对话框（TPrinterSetupDialog）、“查找”对话框（TFindDialog）、“替换”对话框（TReplaceDialog）。

第 12 章　主要介绍了程序控制类组件。包括：滚动条（TScrollBar）、分界组件（TBevel）、尺寸调节杆（TSplitter）、控制条（TControlBar）、滚动条（TScrollBox）、窗格（TPanel）、跟踪条（TTrackBar）、进程条（TProgressBar）、页滚动条控制（TPageScroller）、加/减组件

（TUpDown）、热键组件（THotKey）。

第 13 章　主要介绍了日期显示与控制类组件。包括：日期与时间（TDateTImePicker）、月历（TMonthCalendar）、定时器（TTImer）。

第 14 章　介绍文件系统类组件。包括：文件列表框（TFileListBox）、目录列表框（TDirectoryListBox）、驱动器组合框（TDriveComboBox）

第 15 章　主要介绍了打印类组件。包括：屏幕组件（TScreen）、打印机（TPrinter）。

第 16 章　主要介绍了数据交换类组件。包括：数据交换组件（TClipboard）、动态数据交换的数据端（TDDEClientConv）、客户端 DDE 会话组件（TDDEClientItem）、动态数据交换的服务端（TDDEServerConv）、服务器端 DDE 会话项目（TDDEServerItem）、OLE 容器（TOLEContainer）。

第 17 章　主要介绍了图形类组件。包括：自绘栅格（TDrawGrid）、字符串栅格（TStringGrid）、几何图形（TShape）、画板（TPaintBox）。

第 18 章　主要介绍了图像类组件。包括：图像（TImage）、位图（TBitmap）。

第 19 章　主要介绍了多媒体类组件。包括：媒体播放器（TMediaPlayer）、AVI 播放器（TAnimater）。

第 20 章　主要介绍了数据库访问类组件。包括：数据源（TDataSource）、数据库表（TTable）、查询数据库（TQuery）、存储过程（TStoredProc）、数据集（TDatabase）、会话（TSession）、批量移动数据（TBatchMove）、缓存更新（TUpdateSQL）。

第 21 章　主要介绍了数据感知类组件。包括：显示数据集的数据（TDBGrid）、数据库导航器（TDBNaviator）、显示字符串字段值（TDBText）、显示和编辑当前记录字段（TDBEdit）、显示和编辑多行文本的字段（TDBMemo）、窗格（TPanel）、以窗格形式显示数据集数据（TDBCtrlGrid）。

第 22 章　主要介绍了 ADO 类组件。包括：连接 ADO 数据库（TADOConnection）、ADO 数据集（TADODataSet）。

第 23 章　主要介绍了 Interbase 类组件。包括：访问数据库一个表或视图（TIBTable）、查询 Interbase 以及 ODBC 数据库（TIBQuery）、获取数据库信息（TIBDatabase）、执行 SQL 语句（TIBSQL）、响应由 Interbase Server 发出的事件（TIBEvents）。

第 24 章　主要介绍了客户服务器类组件。包括：包括：以 TCP/IP 连接应用服务器（TSocketConnection）、以 HTTP 方式连接应用服务器（TWebConnection）、提供数据集（TDataSetProvider）。

第 25 章　主要介绍了图报表类组件。包括 Quick Report 报表（TQuickRep）、报表容器（TQRBand）。

第 26 章　主要介绍了 Web 服务器类组件。包括：Web 调度器（TWebDispatcher）、描述和解释 HTTP（TWebRequest）。

由于 Delphi 编程所涉及的知识面极为广泛，而笔者的知识又很有限，所以书中可能出现错误和疏漏，希望广大读者批评指正。

编者

2003 年 11 月

目　录

第 1 章　Delphi 组件概述

1.1　可视组件库（VCL）简介

VCL（全称是 Visual Component Library，可视化组件库），支持 Object Pascal 语言。它是 C++ Builder 和 Borland Delphi 共同使用的一个符合工业标准的组件库，也是真正意义上的面向对象的组件库。正因为如此，Delphi 的组件封装了一些数据集和数据访问的过程与函数，从祖先类中继承了数据和行为。尽管每个组件有其特殊性，但是所有组件都从它们的共同祖先 TComponent 那里继承某些公共属性，因此可以说 TComponent 定义了组件用于 Delphi 环境所必需的最小属性集。VCL 类的继承关系如图 1-1 所示。

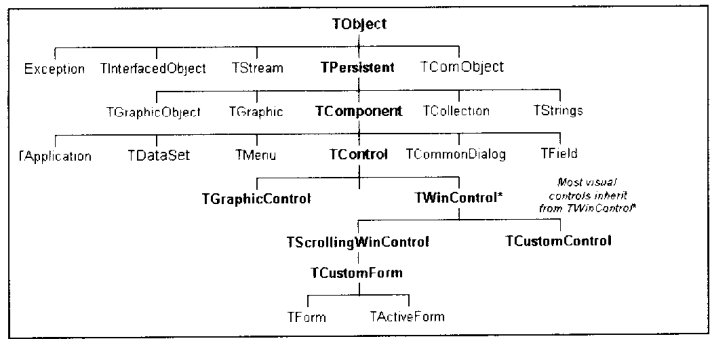

图 1-1　VCL 类图的主要分支

从图中可以看到，TObject 是 VCL 的祖先类，这也是 Object Pascal 语言所规定的。TObject 以及 TObject 声明所在的 system.pas 都是编译器内置支持的，因此用户无法修改、删除 system.pas 中的任何内容，也无法将 system.pas 加入到自己的 project 中，否则会得到“Identifier redeclared 'system'”的错误提示，因为 project 中已经被编译器自动包含了 system 单元。TObject 封装了 Object Pascal 类/对象的最基本行为。

Delphi 的核心是层次结构。系统中的每一个类都是 TObject 类的子类，整个类的层次结构只有一个 TObject 根类。这允许用户在系统中用 TObject 数据类型替代任何类的数据类型。VCL 定义了相当数量的 TObject 子类。TPersistent 由 TObject 派生，它自身及其派生类对象具有自我保存、持久存在的能力。TComponent 由 TPersistent 派生，这条分支之下所有的类都可以被称为“组件”。组件的一般特性如下：

（1）可出现在开发环境的“组件板”上；

（2）能够拥有和管理其他组件；

（3）能够存取自身（这是因为 TComponent 派生自 TPersistent）。

TControl 派生自 TComponent，其分支之下所有的类，都是在运行时可见的组件。TWinControl 派生自 TControl，这个分支封装了 Windows 系统的屏幕对象，也就是一个真正的 Windows 窗口（拥有窗口句柄）。TCustomControl 派生自 TWinControl。从 TCustomControl 开始，组件拥有了 Canvas（画布）属性。

组件是 TComponent 类的子类，也是 Delphi 应用程序的核心元素和界面设计的重要组成部分。当用户编写程序时，首先要选择一些组件并定义它们的相互作用。VCL 封装了 Windows 底层的 API 函数，使用户在不需要了解更多有关 Windows 编程知识的前提下，就可以开发出界面美观和功能强大的 Windows 程序。Delphi 中有不同种类的组件，大部分组件都包含在组件板中。它们可以以流的形式储存在 dfm 文件中，并且可拥有属性和可视化处理的事件。组件可以分为可视化和非可视化两种。可视

化组件又称为“组件”。

组件（Controls）是从 TControl 派生出来的类。组件在屏幕上有位置和大小，并且设计时在窗口显示的位置与运行时相同。组件有两种不同的规格，基于窗口的和基于图形的。基于窗口的组件，即窗口组件（Windowed control），是系统窗口的可视化组件，都有窗口句柄，并可以接受输入焦点，可以含有其他组件。图形组件（Graphical），即非窗口组件，没有窗口句柄，不能接受焦点，也不能包含其他组件。这些组件继承于 TGraphicControl 并由它们的父窗口显示，它们在最小化使用系统资源方面起着关键作用。

非可视化组件是指除组件以外的所有组件，代表所有从 TComponent 继承但不从 TControl 继承的类。在设计时，非可视化组件以图标的形式出现在窗体上，在运行时看不见。可以将鼠标移到组件或组件上查看它的名称及其类型，也可以通过选定窗体设计器（Form designer）栏的 Show Component Captions 选项，来查看非可视化组件的名称。

1.2 Delphi组件的优势

Delphi 的优秀很大程度上得益于 VCL 的优秀，VCL 是 Delphi 所提供的基本组件库，也就是所谓的 Application Framework，它对 Windows API（应用程序接口）进行了全面的封装，为桌面开发（不限于桌面开发）提供了整套的解决方案，使得程序员可以在不知晓 API 的情况下进行 Windows 编程。

不过，作为专业的程序员，不知晓 API 是不可能的。VCL 还是一个应用程序框架（Application Framework），可以将 VCL 作为一个平台，程序员在其基础上构建应用程序，便可以忽略很多系统 API 的细节，从而使开发速度更快。

VCL 的组件也不同于 ActiveX 组件，VCL 组件通过源代码连接到可执行文件中，因此其速度更快。而且，企业版的 Delphi 带有全部 VCL 库的源代码，这样程序员不单单可以知道如何使用 VCL 组件，更可以了解其运行机制与构架。了解 VCL 的构架，编写自己的 Application，设计程序框架，或者创建自己的组件/类，融入到 VCL 构架中，都是必须和大有裨益的。这也符合某种规律：在学习的时候，求甚解；而在应用的时候，则寻找捷径。Delphi 和 VCL 都能满足这两种需求，因为使用它可以不隐藏任何想知道的细节，可以忽略任何不想知道的细节。

Delphi 作为 RAD（通过组件拖拽）工具，以其快速编译和友好的可视化界面受到广泛欢迎。Delphi 提供了很多现成组件，而且随着版本更新不断增加新组件（即从第三方买到其开发的特色组件，或从下载免费组件）。这些组件足以支持一般应用系统开发，但应用开发人员仍有必要自己制作组件。采用组件形式可以把对象严密封装，并加上一层直观外壳，有利于软件调试和代码重用。开发群体以组件为功能单位分工协作，比较容易实现工程化管理，从软件规划设计到测试修改都可以减少意外差错，大大提高工作效率。成熟的组件还可以作为商品软件出售，带来附加效益，且有利于软件开发的社会化分工协作。Delphi 的组件使用和组件制作采用同样的工作环境和相似的编程方法，只要弄清基本原理，在制作组件时无需学习多少新东西。

VCL 使得开发周期比传统的面向对象开发方法（OPP）缩短了 1/2～1/4。它出色地封装了 Windows 编程元素，其数据隐藏特性提高了编程效率和准确率，并有助于隐藏组件关键的数据结构和提供正确、抽象的公共接口。由于有了基于 VCL 库的可视化开发工具，Delphi 在组件技术的支持、数据库支持、系统底层开发支持、网络开发支持、面向对象特性等各方面都有相当不错的表现，并且学习使用较为容易，充分提现了所见即所得的可视化开发方法，开发效率较高。而且因为是 Borland 公司的产品，自然继承了该公司一贯以来的优良传统：代码执行效率高。

1.3 组件运行机制

本节将带领读者游历 VCL 库的核心，剖析 VCL 的代码。

首先看一段 TObject 的声明：

```
TObject = class
    constructor Create;
```

```
    procedure Free;
    class function InitInstance(Instance: Pointer): TObject;
    procedure CleanupInstance;
    function ClassType: TClass;
    class function ClassName: ShortString;
    class function ClassNameIs(const Name: string): Boolean;
    class function ClassParent: TClass;
    class function ClassInfo: Pointer;
    class function InstanceSize: Longint;
    class function InheritsFrom(AClass: TClass): Boolean;
    class function MethodAddress(const Name: ShortString): Pointer;
    class function MethodName(Address: Pointer): ShortString;
    function FieldAddress(const Name: ShortString): Pointer;
    function GetInterface(const IID: TGUID; out Obj): Boolean;
    class function GetInterfaceEntry(const IID: TGUID): PInterfaceEntry;
    class function GetInterfaceTable: PInterfaceTable;
    function SafeCallException(ExceptObject: TObject;
      ExceptAddr: Pointer): HResult; virtual;
    procedure AfterConstruction; virtual;
    procedure BeforeDestruction; virtual;
    procedure Dispatch(var Message); virtual;
    procedure DefaultHandler(var Message); virtual;
    class function NewInstance: TObject; virtual;
    procedure FreeInstance; virtual;
    destructor Destroy; virtual;
  end;
```

从 TObject 的声明中可以看到，TObject 包含了诸如实例初始化、实例析构、RTTI、消息分发等相关实现的方法。在 TObject 类中，有一个 Dispatch()方法和一个 DefaultHandler()方法，它们都是与消息分发机制相关的。Dispatch()负责将特定的消息分发给合适的消息处理函数。首先它会在对象本身所属的类中寻找该消息的处理函数，如果找到，则调用它；如果没有找到而该类覆盖了 TObject 的 DefaultHandler()，则调用该类的 DefaultHandler()；如果两者都不存在，则继续在其基类中寻找，直至寻找到 TObject 这一层，而 TObject 已经提供了默认的 DefaultHandler()方法。

TObject 提供了最基本的消息分发和处理的机制，而 VCL 真正对 Windows 系统消息的封装则是在 TControl 中完成的。TControl 将消息转换成 VCL 的事件，把系统消息融入到 VCL 框架中。上面已经介绍过消息分发机制，下面以鼠标消息变鼠标事件的过程来解释系统消息是如何变成事件的。TControl 声明了一个 OnMouseDown 属性，该属性读写一个称为 FOnMouseDown 的事件指针。因此，FOnMouseDown 会指向 OnMouseDown 事件的用户代码。TControl 声明了 WMLButtonDown、WMRButtonDown、WMMButtonDown 这 3 个消息处理函数，它们分别处理 WM_LBUTTONDOWN、WM_RBUTTONDOWN、WM_MBUTTONDOWN 这 3 个 Windows 消息，对应于鼠标的左键按下、右键按下、中键按下 3 个硬件事件。以下是 3 个消息的处理函数：

```
procedure TControl.WMLButtonDown(var Message: TWMLButtonDown);
begin
  SendCancelMode(Self);
  inherited;
  if csCaptureMouse in ControlStyle then
```

```
            MouseCapture := True;
        if csClickEvents in ControlStyle then
            Include(FControlState, csClicked);
        DoMouseDown(Message, mbLeft, []);
    end;
    procedure TControl.WMRButtonDown(var Message: TWMRButtonDown);
    begin
        inherited;
        DoMouseDown(Message, mbRight, []);
    end;
    procedure TControl.WMMButtonDown(var Message: TWMMButtonDown);
    begin
        inherited;
        DoMouseDown(Message, mbMiddle, []);
    end;
```

当 TObject.Dispatch()将 WM_LBUTTONDOWN 消息、WM_RBUTTONDOWN 消息或者 WM_MBUTTONDOWN 消息分发给 TControl 的派生类的实例后，WMLButtonDown()、WMRButtonDown()或 WMMButtonDown()被执行。然后它们都用如下的代码调用 DoMouseDown()：

```
            DoMouseDown(Message,mbRight,[])
    procedure TControl.DoMouseDown(var Message: TWMMouse; Button: TMouseButton;Shift:
TShiftState);
    begin
        if not (csNoStdEvents in ControlStyle) then
            with Message do
                if (Width > 32768) or (Height > 32768) then
                    with CalcCursorPos do
                        MouseDown(Button, KeysToShiftState(Keys) + Shift, X, Y)
                else
                    MouseDown(Button, KeysToShiftState(Keys) + Shift, Message.XPos, Message.YPos);
    end;
```

在 DoMouseDown()进行一些必要的处理工作后（特殊情况下重新获取鼠标位置），就会调用 MouseDown()：

```
    procedure TControl.MouseDown(Button: TMouseButton; Shift: TShiftState; X,Y: Integer);
    begin
        if Assigned(FOnMouseDown) then
            FOnMouseDown(Self, Button, Shift, X, Y);
    end;
```

在 MouseDown()中，才会通过 FOnMouseDown 事件指针去执行用户定义的 OnMouseDown 事件的代码。

至此，Windows 消息系统到 VCL 事件的转换过程就完成了。由此可见，从 TControl 派生的类都可以拥有 OnMouseDown 事件，只不过该事件属性在 TControl 中被定义成 Protected，只有其派生类可见，并且在派生类中可以自由选择是否公布这个属性。要公布该属性只需要简单地将其声明为 Published 即可。如：

```
    TMyControl = class (TControl)
    published
```

```
    property OnMouseDown
  end;
```

如果读者曾经使用纯 API 编写过 Windows 程序，一定知道 Windows 应用程序的每一个窗口都有一个大的消息循环以及一个窗口函数（WndProc）用以分发和处理消息。VCL 作为一个 Framework，当然会将这些东西隐藏起来，而重新提供了一种易用的、易理解的虚拟机制给程序员。只要代码单元中包含了 Form.pas，就会得到一个对象——Application。Application 对象是 VCL 提供的，在 Form.pas 中可以看到如下这个定义：

```
var
    Application:Tapplication
```

从表面上看，TApplication 类定义了一个应用程序的特性及行为，可以从 Application 对象得到应用程序的可执行文件名称（ExeName），设置应用程序的标题（Title）等属性，也可以执行最小化（Minimize）、打开帮助文件（HelpCommand）等操作。

当创建一个默认的应用程序时，会自动得到以下几行代码：

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

这几行代码很简单地展示了 TApplication 的功能：初始化、创建必要的窗体、运行。

TApplication 的构造函数主要完成了两件事情：注册窗口类及窗口函数，创建 Application 窗口实例。TApplication 类的 Run 方法中有这样一段代码：

```
repeat
        try
          HandleMessage;
        except
          HandleException(Self);
        end;
until Terminated;
```

这是主消息循环。看上去似乎没有取消息、分发消息的过程，其实它们都包含在 HandleMessage()方法中了。HandleMessage()方法其实是对 ProcessMessage()方法的调用，而在 ProcessMessage()中就可以看到取消息、分发消息的动作了，以下是 TApplication 中的 ProcessMessage()方法的源代码，请注意其中的注释：

```
function TApplication.ProcessMessage(var Msg: TMsg): Boolean;
var
  Handled: Boolean;
begin
  Result := False;
  // 取消息
  if PeekMessage(Msg, 0, 0, 0, PM_REMOVE) then
  begin
    Result := True;
    if Msg.Message <> WM_QUIT then
    begin
      Handled := False;
      if Assigned(FOnMessage) then FOnMessage(Msg, Handled);
```

```
        if not IsHintMsg(Msg) and not Handled and not IsMDIMsg(Msg) and
          not IsKeyMsg(Msg) and not IsDlgMsg(Msg) then
        begin
         // 熟悉的分发消息过程
          TranslateMessage(Msg);
          DispatchMessage(Msg);
        end;
      end
      else
        // 如果取到的消息为 WM_QUIT，则将 Fterminate 设为真
        // 以通知主消息循环退出
        // 这和 WindowDemo 程序中判断 GetMessage()函数返回值是否为 0 等效
        // 因为 GetMessage()函数取出的笑傲西如果是 WM_QUIT，它的返回值为 0
        FTerminate := True;
    end;
  end;
```

窗口函数是一个回调函数，它被 Windows 系统所调用，其参数会给出消息编号、消息参数等信息，以便进行处理。TApplication 的 Createhandle 函数将 Application 窗口的窗口函数设置为 WndProc()。整个 WndProc()方法基本只包含了一个庞大的 case 分支，其中给出了每个消息的处理代码，"WM_"打头的为 Windows 定义的窗口消息，"CM_"开头的为 VCL 库自定义的消息。

需要注意的是，这里给出的 WndProc 是属于 TApplication 的，而每个 Form 另外有自己的窗口函数。

至此，读者应该清楚了 VCL 框架是如何封装 Windows 程序框架的了。知道 VCL 做了哪些工作，它想要提供的是怎样的一个世界，这对于更好地融入 VCL 是大有好处的。

第 2 章　公共属性、事件和方法

2.1　Delphi集成开发环境简介

Delphi 是一种基于 Object Pascal 语言的可视化集成开发工具。利用 Delphi 编程，可以快速、高效地开发出基于 Windows 环境的各类程序，尤其在数据库和网络方面，Delphi 更是一个十分理想的软件开发平台。

可视化开发环境通常分为 3 个组成部分：编辑器、调试器和窗体设计器。与大多数现代 RAD（快速应用开发）工具一样，这 3 部分是协同工作的。当用户在窗体设计器中工作时，Delphi 在后台自动为正在窗体中操纵的组件生成代码。用户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。

启动 Delphi 后，将出现如图 2-1 所示的集成开发环境的界面。

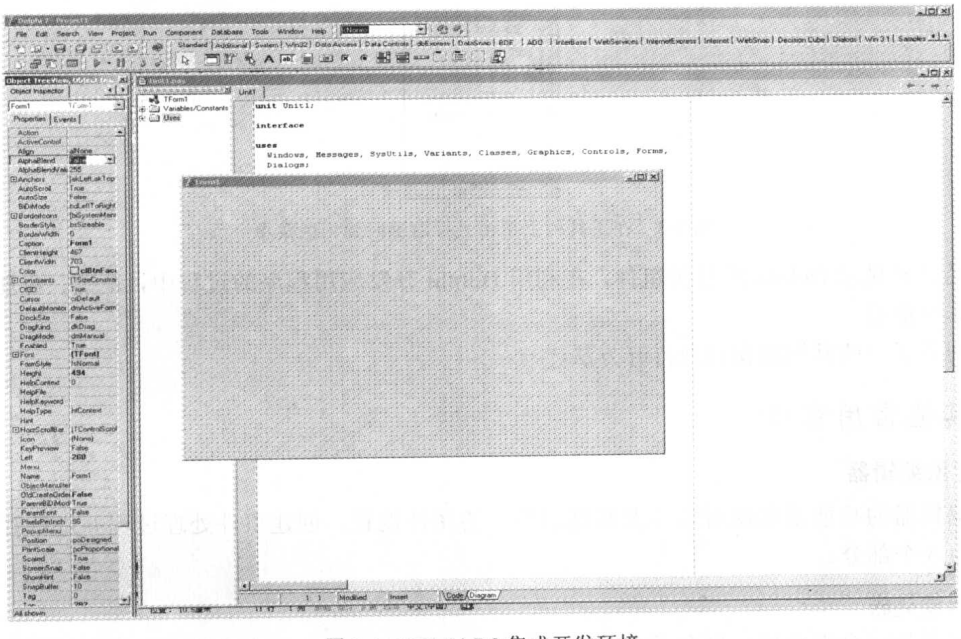

图 2-1　Delphi 7.0 集成开发环境

Delphi 集成开发环境的界面包括标题栏、主菜单、工具栏、组件栏（如图 2-2 所示）和一些窗口。

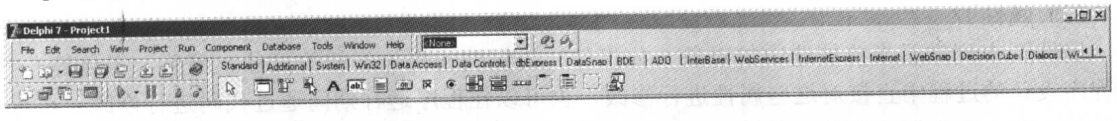

图 2-2　Delphi 7.0 操作界面

2.1.1　标题栏、主菜单、工具栏和组件栏

标题栏显示了当前的工程名，位于最上部。

Delphi 7 的菜单栏紧挨在主窗口标题下面，它的所有功能都可以通过菜单栏上的命令来实现。例如，要向项目中加入新的对象，可以使用“File”菜单上的“New”命令；要打开一个已有的项目，可以使用“File”菜单上的“Open”命令；要安装新的元件，可以使用“Component”菜单上的“Install Component”命令；要设置环境选项，可以使用“Tools”菜单上的“Environment Options”命令。

菜单栏下面的左半部分就是工具栏，工具栏上有若干个 16×16 的位图按钮，一些常用的菜单命令

就是用这些快捷键按钮实现的。工具栏是可以自定义的，读者可以改变工具栏的宽度，在工具栏上增加、删减按钮或者调整按钮的顺序。

组件栏在菜单栏的下面，工具栏的右边，这里集中了 Delphi 7 将近 200 个组件的图标。要使用其中某个组件，读者可以在组件栏上双击组件的图标，或者先单击该组件的图标，然后再单击 Form 中的任意某个位置，组件就会显示在 Form 上。

读者可能注意到，工具栏上的按钮和组件栏上的图标都具有提示功能，称为 ToolTip。提示的内容主要是关于按钮或组件的简短描述。只要把光标指向一个按钮或组件的图标并停留一秒钟以上，就会激活提示功能。

通过主菜单可以实现集成开发环境中的绝大部分命令，在程序开发的过程中，用户可以在不同的情况下，在界面的不同部位单击光标右键，看是否会弹出菜单，以及通过这些弹出菜单能实现什么新的功能或快捷操作，Delphi 在主菜单栏中将“Component 组件”和“Database（数据库）”作为独立的菜单项列出来，可以看出，利用 Delphi 进行程序开发时，组件和数据库将是两个非常重要的方面。在工具栏上单击光标右键会弹出如图 2-3 所示的菜单，通过该菜单可以方便地控制工具栏和组件栏的显示及隐藏。

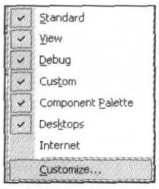

图 2-3　在工具栏上单击光标右键时弹出的菜单

组件栏分页显示 Delphi 的各种组件，在利用 Delphi 开发应用程序的过程中，正确、合理、恰当地使用组件非常重要。

下面列举了一些常用的窗口及打开方法。

2.1.2　其他常用窗口

1. 对象编辑器

对象编辑器的功能是实现对象（尤其是组件）的属性设置，创建事件处理过程并进行管理。对象编辑器分为 3 个部分。

（1）对象列表

对象列表是一个组合框，包含了当前窗体上的所有组件。有时一些组件因为太小或没有明显的标志，利用光标单击不容易选中，这时可以利用对象列表组合框来选择该组件。

（2）属性页

属性页中列举了当前被选中的对象（如组件）的属性（例如组件的尺寸、颜色、字体等）。可以在程序设计的过程中直接对这些属性进行修改，也可以在程序运行期间通过代码进行修改。在有些属性的前面有一个带有方框的加号标志，表明该属性是由一些子属性组成的。单击该加号标志就可以展开该属性，同时加号标志变成了一个带有方框的减号标志。

属性页在显示属性时分成两列，左边一列是属性标题，叫做属性的 Caption 列，右边一列是属性的值叫做属性的 Value 列。在设置属性的值时，首先必须选中该属性，然后在属性的 Value 列中修改。按快捷键 F11 或使用菜单 View|Object Inspector 可以激活窗口，窗口如图 2-4 所示。

（3）事件页

通常情况下，事件处理过程为空。可以双击事件右面的组合框来添加事件的处理过程。如果要共用已存在的事件处理过程，可通过下拉式组合框进行选择。

与属性页相同，事件页也同样分为两列，左边一列是事件的标题，右边一列是事件的处理过程。