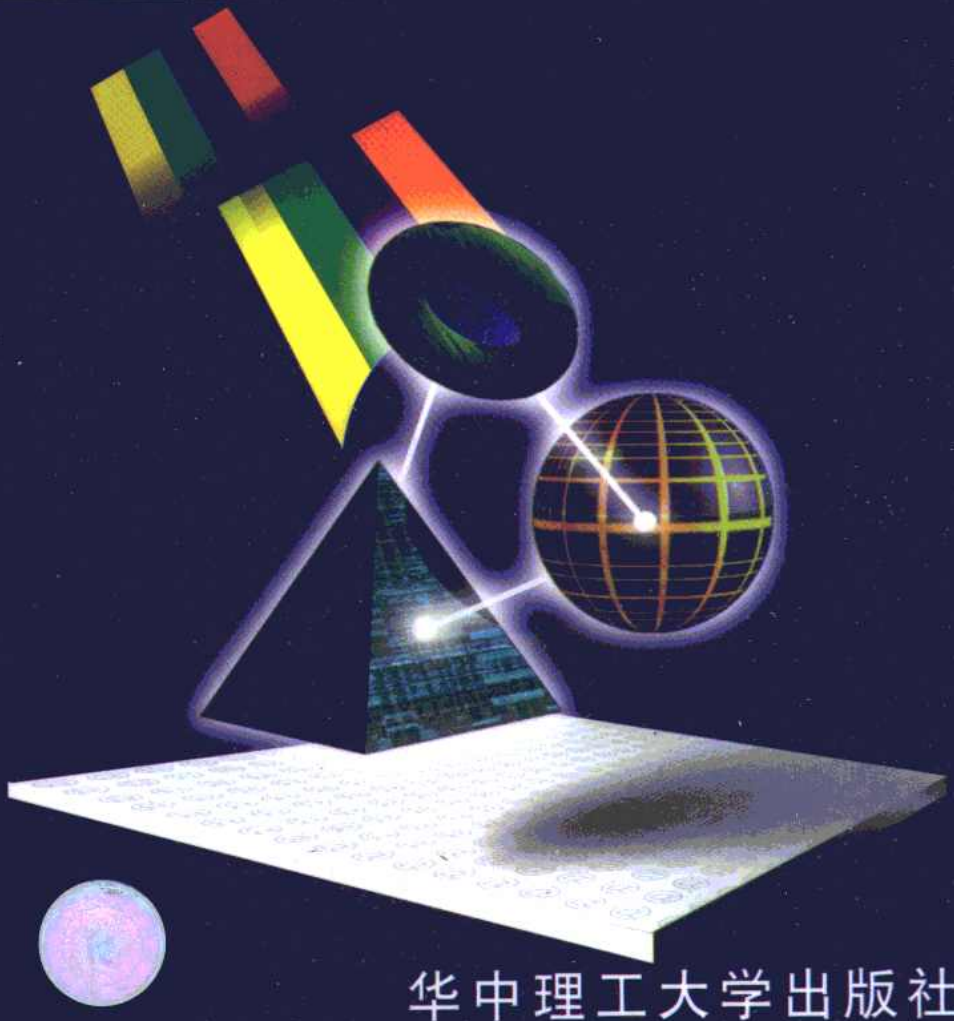
计算机实用技术培训·应用丛书

Delphi 5 网络编程

窦万峰 张子瑜 编著



华中理工大学出版社
HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY PRESS
E-mail: hustpp@wuhan.cngb.com

Delphi 5 网络编程

窦万峰　张子瑜　编著

华中理工大学出版社

前　言

Delphi 结合了可视化技术、面向对象技术、数据库技术、网络开发技术等多种先进的软件编程技术和思想，成为创建功能丰富、界面友好的 Windows 应用软件的工具之一。

Delphi 5 是 Inprise 公司 1999 年 8 月推出的可视化 Windows 应用程序开发工具，它是对 Delphi 4 的扩展和完善，并加入了许多的新的编程方法和工具。Delphi 5 加强了数据库与网络程序开发的功能，如 InterBase 数据库编程、基于 Web 的多层分布式应用编程等，并增加和完善了许多新的技术，如 CORBA、DHTML 等。

作者在使用 Delphi 5 进行应用开发的经验基础上，结合一些最新资料和大量开发实例，介绍 Delphi 5 实用软件开发技术。本书注重编程的实用性和技巧性，通过一些实例让读者能够深入地掌握 Delphi 5 的编程实质和强大功能。书中的许多例子都是从我们开发的程序和其他实际应用程序中部分摘选来的，因而具有实用性。我们衷心希望这些例程能够为读者的 Delphi 学习和程序开发带来有益的帮助。

本书的内容分为五个部分，共 10 章。第 1 章主要介绍 Delphi 5 新增功能与特点，介绍了集成开发环境，给出了一些基本应用实例，包括窗体与组件基本开发实例、菜单编程应用与实例、图形图像应用、多媒体应用和 Delphi 5 多线程应用等。这些应用实例读者可以直接应用到自己的开发工作中，其中包含的许多开发技巧可以帮助读者体会到 Delphi 5 编程的实质。

第 2~6 章主要介绍 Delphi 5 的 COM 与 COABA 技术与编程、自动化服务器与浏览器的开发、Web 编程技术和分布式应用技术等。Delphi 5 发展了其网络开发功能，如多层分布式应用编程、更加完善的 CORBA 技术、支持 DHTML 等。该部分详细地介绍了这些新增功能和编程方法。这一部分涉及到许多复杂的知识与概念，为了能够让读者全面掌握 Delphi 5 的网络应用，我们将采取理论与应用相结合的方法，同时介绍概念与程序代码，方便读者的理解和学习。另外，读者要知道网络开发的整个框架，这样才不致于由于太多的概念而导致混乱。

第 7 章介绍了 Internet/Intranet 编程技术，介绍了各种通信协议组件功能，通过大量实例介绍基于各种协议的网络数据传输，包括邮件服务与数据解码/编码、媒体数据流传输、文件传输应用、WWW 浏览器应用等。

第 8~10 章主要介绍数据库及其网络编程技术，包括数据库编程、网络数据库应用开发等。该部分通过许多实例介绍了数据显示、数据输入、数据查询和字段处理、主细表应用、数据缓存更新、InterBase 数据库应用、存储过程应用、数据批复制，以及事务处理和网络数据库应用等。Delphi 5 数据库的强大功能使得数据库开发变得更加方便快速，从而使开发人员能集中精力致力于数据库事务处理和灵活应用方面。

第 11~13 章介绍了分布式数据库应用与开发技术，包括 MTS 应用开发、ADO 编程、MIDAS 多层应用开发等。该部分介绍了 Delphi 5 极其强大的数据库应用功能，如多层的数据库应用开发模式使客户应用程序变“瘦”，更加灵活，提高了通信效率。MTS 提供了

强大的远程数据事物处理机制，使得数据处理更加安全可靠。

本书第 1 章及第 7~10 章由窦万峰同志执笔，第 2~6 章由张子瑜同志完成，刘菁华和董爱春同志参加了部分章节的录入和校对工作。全书由窦万峰统稿。本书在编写过程中还得到了王保保、李华和冯象初同志的大力支持，在此表示感谢。

本书中列举了大量的编程实例，深入地介绍了从 Delphi 5 基本程序编写到数据库及网络应用程序开发，使读者能深入地掌握 Delphi 5 编程技巧及其实用开发技术。

本书适用于广大计算软件开发人员和计算机爱好者学习和使用。

由于时间和水平有限，不妥之处在所难免，敬请广大读者提出宝贵意见并与我们联系，不胜感激。

编 者

2000 年 5 月于南京大学

第一章 Delphi 5 开发基本应用

1.1 Delphi 5 新增功能

Delphi 5 软件开发工具是一个完全导向的可视化系统开发工具，具有功能强大、运行速度快、易于使用以及开发迅速等特点。它结合了可视化技术、面向对象编程、数据库和分布式应用技术等先进的软件编程技术和思想，并使用了全特征的代码编辑器，使其成为创建功能丰富、界面友好的 Windows 应用软件的工具之一。

下面是 Delphi 5 的新增功能：

● Object Pascal 语言扩展：Delphi 5 扩展了 Object Pascal 语言的一些功能，包括动态数组定义、方法重载和缺省参数设置等等。

● 增加的工程管理功能：Delphi 5 提供了一个新的工程管理器。该工程管理器允许开发人员将多个工程组合在一个工程组中进行处理，这种功能可以帮助开发人员组织和操作多个相互关联的工程（如多列应用的单列处理或 DLLs，以及可执行文件等）一起工作。

● RTL 支持 2000 年表示：TwoDigitYearCenturyWindow 变量用于 StrToDate 和 StrToDateTime 函数中，控制进行日期转换时的两位数字的解释，可以避免 2000 年问题。

● 增强的 MIDAS 功能：Delphi 5 提供了许多对多层应用的控制，包括更新同步支持和对数据包中包含的数据和更新如何应用等方面的控制。一个新类 TDataSetProvider 允许用户使用嵌套表从一个支持 Master/detail 的数据集中取得数据或发送数据到它们之中。另外，用户数据集的增强功能使参数传递到应用服务器或保存用户信息到数据包中更为方便。

● 增强数据库功能与客户机数据集：Delphi 5 的数据连接组件允许开发人员调整数据模型，以便建立数据模块或窗体。修改的数据连接组件和 BDE 允许用户使用新的数据服务器类型，包括 Access'97 和 Oracle8 到 SQL 的新扩展（包括 ADTs）、数据、索引和嵌套表。全新的 InterBase Express 数据库组件使得 InterBase 数据库连接和服务器的事务控制更方便快捷。可视化 Query Builder 已被 SQL Builder 替代，这是一个新的智能查询创建器。Delphi 5 客户机数据集支持过滤表达式变量和维护总量，以及允许面向对象的域类型。

● 强大的 COM 技术支持：Delphi 5 的 DAX 技术有力地将 COM 技术融入任意的网络开发中，使其更方便地支持 MTS、ActiveX、MIDAS 和自动化对象与服务器开发。

下面是更新到 Delphi 5 时将会遇到的兼容问题：

● Delphi 5 包的扩展名被改动，由原来的.DPL 替换为.BPL。

● 如果你已经创建了自己的设计包，Delphi 5 要求你在安装到 Delphi 5 之前必须重建该包。为了能够识别其它包的制造者(如 C++Builder3、Delphi 5 等)，内部包格式也做了修改。

● C++Builder3 的.BPL 文件不能在 Delphi 5 上使用。如果你购买了第三方的 Delphi 组

件，那么你必须使用它们提供的源代码。

● Delphi 5 不能打开 Delphi3 的 DPK 文件，而是自动转换成 Delphi 5 包新的源格式。

● 改变所有 SYSDATE 实例到 GetDate。不能使用 Year、Month、Day、Hour、Minute 或 Second 提取时间、日期等。

● OnDragOver 和 OnDragDrop 事件原参数被修改。在 Delphi4 中，OnDragOver 和 OnDragDrop 事件可以使用 TDragControlObject 的源参数执行，不需要知道其内部拖动控制。在 Delphi 5 中，事件源参数实际上控制拖动操作的开始。

● DWORD、UINT、HResult、OLE_HANDLE 和 API 处理类型被改变。DWORD、UINT、HResult、OLE_HANDLE 和所有的 API 处理类型被定义为无符号的 2 位 LongWord，它们不再与 Integer 类型兼容。

● 新的 Int64/LongWord 类型。为了进行算术运算，将整型结果转到 Int64，避免溢出。因此，某些不适合于 Integer 类型的变量将处理成 Int64(64 位 Integer)/LongWord (无符号 32 为 Integer)类型。

● Delphi 5 提供了 NetManage 的 Internet 组件。

● 与 Oracle8 兼容。在 SELECT 陈述中，关键字 OR UPDATE 会锁住结果集，你必须将 QLPASSTHRU MODE 设为 SHARED NOAUTOCOMMIT。

1.2　Delphi 5 开发环境

Delphi 5 是一个运行在 Windows 环境下的可视化编程工具软件，可以用来创建 Windows 的各种应用程序。Delphi 提供了易于使用的开发工具，供软件开发人员创建程序中的可视化部分。使用者只需简单地使用鼠标点取 Delphi 5 组件模板页上的任意一个组件，在自己的窗体上任意拖放和设置。Delphi 5 的集成开发环境(Integrated Development Environment 简称 IDE)如图 1.1 所示，其四周由标签形式的控件构成。单击一个标签，就可以得到一个不同的选项页面。

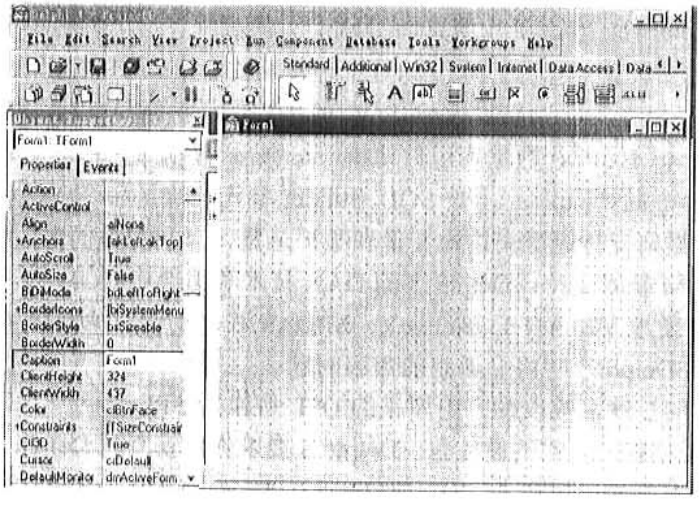


图 1.1　Delphi 5 的集成开发环境

1）主屏幕菜单与加速条

主屏幕菜单和一般 Windows 的各种应用程序菜单类似，只是其菜单选项有所不同，功能上有一些差异而已。加速条是一些菜单命令的快捷按钮，它们可用来快速执行菜单命令。它们分别是：打开工程项目设计、保存工程项目设计、向工程中加入文件、从列表中选单元、从列表中选择窗体、程序运行、程序暂停、打开文件、保存文件、从工程中去掉一个文件、窗体/ 单元切换、新窗体、跟踪和单步执行等。

2）对象监视器

对象监视器(Object Inspector)可以设置程序中使用的组件对象的属性和事件。对象监视器有两个标签页，即属性页和事件页。属性(Property)页用于设置窗体和窗体上组件的属性。比如一个按钮(Button)，可用对象监视器来设置它的大小、位置、颜色、显示效果、字体、字型、名字等属性。 如选择窗体中的一个组件，在对象监视器中对其属性进行设置或修改，其效果可立刻在窗体中反映出来。

窗体和组件的事件用于编写窗体和组件在程序运行过程中所要产生的动作和对动作的响应，比如窗体的 OnClose 事件就是用于完成窗体的关闭。

3）组件板

组件是建立 Delphi 应用程序时的要素，它包含了一个应用程序所有可见的部分，如对话框和按钮，以及程序执行时隐藏的部分，如系统计时器或是动态数据交换(DDE)服务器。

组件板(Component Pallette)位于窗体的右上部，它有多个组件页，每个组件页上又含有一些不同用途的组件，使用者可根据窗体设计要求进行组件选择和布局。Delphi 5 提供了非常丰富的组件，设计者使用它们可以设计出各种 Windows 应用程序所需要的窗体界面。

若要选取某个页上的组件，使用者首先须激活该组件页，然后加亮要选取的组件，接着用鼠标在窗体合适的位置上单击，并放置或拖动组件的位置和大小。

4）窗体、单元、项目及表单

在缺省情况下，每次打开 Delphi 时，Delphi 的集成开发环境会自动地使用缺省窗体(在屏幕的右下部)。该窗体是一个空白框，上面什么组件也没有，使用者可在上面设计自己的界面。当然，用户可以通过 Delphi 提供的窗体模板来设置不同的缺省窗体。用户可将窗体想象成一个可以放置其它组件的容器，应用程序的主窗体和所属的组件与其它的窗体与组件交互地组成应用程序的用户界面。主窗体是应用程序的主要界面(Main Interface)，其它的窗体可以是对话框、数据录入框等等。

单元是用来编写程序代码的文件。每个窗体都对应着一个单元，用以对窗体和窗体上的组件进行控制或者完成某些动作或处于某种状态。双击一下窗体，就会自动弹出一个单元文件的代码编辑器。单元文件以扩展名 .PAS 为结尾，其代码是用 Pascal 语言编写。单元文件主要由 Unit(单元名)、Interface(接口)、Uses 子句、Type(类型定义)、Var(变量定义)、Implementation(实现)和 Procedure(过程)等几大部分组成。Delphi 具有自动编写窗体和窗体中组件对象定义和类型及部分变量的代码的功能，从而大大减轻了开发人员的工作量。

Delphi 还提供了工程文件(扩展名为.DPR)和窗体表单文件(扩展名为.DFM)。工程项目文件用于组织和管理多个窗体和单元，这与 C 语言中的工程文件相类似。

窗体表单文件用于记录窗体和窗体上的组件的各种属性设置。表单文件是一个二进制

形式的文件，而非 ASCII 码文件。不过，它们可以以 ASCII 码的形式进行存储，以便修改和重用等。

5）对象属性、方法和事件

属性是组件本身所含的数据，可以在对象监视器中看到组件的一部分属性项，另一部分是实时属性，可以使用代码在程序运行时设定。

用户可以通过改变组件的属性值来改变这一对象的特征及行为。比如，使一个按钮组件有效，则可设置其 Enabled 属性为 True，也可使用代码：

```
Button1.Enabled:=true;
```

对象本身所含有的过程或函数成为方法或对象方法，通过对象方法来操纵该对象或者取得其内部的私有数据。比如，要显示一个窗体，则使用其 Show 方法：

```
Form1.Show;
```

事件是指外界的激发或某种状态的改变，例如，一个单击或移动鼠标等。Windows 是基于事件运行，当“事件”发生时，程序才会执行，若没有事件发生，整个程序将处于等待状态。

事件句柄(Event Handler)就是当事件发生时程序应作出的响应。在 Delphi 中，事件句柄实际包含了用户为了某个过程所编写的执行代码。

1.3　Delphi 5 模板与专家

Delphi 5 产生的窗体可以在 Delphi 的其它工程中被重复使用。而且它也可以被编译成动态链接库(DLL)，因此可以被利用 C++、Paradox、dBASE、Visual Basic 及 Power Builder 开发的程序的人员使用。Delphi 5 提供了多种窗体工具，可以让设计者比以往更轻松、更有效率地生成一个窗体和报表。

每当启动 Delphi 5 时，Delphi 5 会自动打开一个缺省的项目文件 Project1.dpr 表单文件和单元文件，这是由于 Delphi 5 提供的模板所致。

窗体模板使设计者可以从多种事先定义好的窗体中选择所需要的界面，设计者也可将自己的窗体存入窗体模板中。为了提高软件设计的效率，Delphi 5 提供了一些专门的例程供开发人员选用，从而加快开发速度。使用者可以利用这些模板和专家来设置这些缺省文件。选择 File|New 命令，则出现如图 1.2 所示的模板与专家对话框。

Delphi 的模板与专家对话框包括：

- New(新对象单元)：包括 Application(应用程序)、Component(组件)、DataModule(数据模块)、DLL(动态链接文件)、Form(窗体)、Package(新包)、Remote Data Module(远程数据模块)、Report(报表)、Text(文本文件)、Thread Object(线程对象)、Unit(单元)和 Web Server Application（Web 服务器应用）等。
- ActiveX：包括 Active Form(Active 窗体)、Active Control(Active 控制)、Active Library(Active 库)、Automation Object(自动化对象)、Property Page(属性页)和 Type Library(类型库)等。
- Multitier：包括 CORBA Data Module、CORBA Object、MTS Data Module、MTS Object

和 Remote Data Module 等。

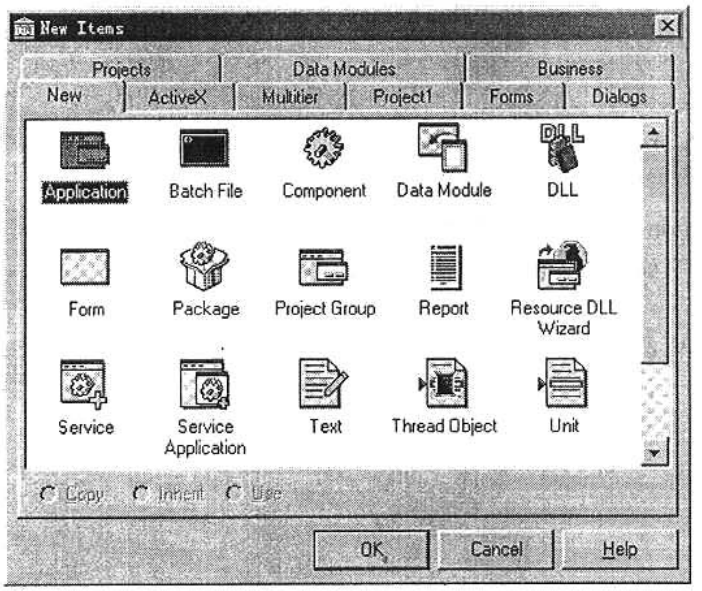


图 1.2　Delphi 5 的模板与专家对话框

● Project1（新工程）：建立窗体 Form1。

● Forms(窗体)：包括 About Box(信息框)、Dual List Box(多列表框)、Quick Report Labels(快速报表标签)、Quick Report List(快速报表列表)和 Quick Report Master/Detail(主细报表)。

● Projects（工程文件专家）：包括 Application(应用专家)、MDI Application(多窗体应用)、SDI Application(单窗体应用)和 Win95 Logo Application（Win95 日志应用）等。

● Dialogs（对话框）：包括 Dialog with Help（对话框帮助)、Dialog Wizard(对话框专家)、Password Dialog(口令对话框)、Reconcile Error Dialog(错误调节对话框)、Standard Dialog(标准对话框)等。

● Data Modules（数据模块）

● Business（图表应用）：包括 Database Form Wizard(数据库专家)、Decision Cube Sample(精度块例程)、Quick Report Wizard(快速报表专家)和 Tee Chart Wizard(图表专家)等。

工程专家可以根据设计者的实际需要来设置，以产生各种不同类型的例程。Delphi 提供的这些专家例程能够引导开发人员进入某些复杂的操作步骤之中。比如选择一个标准下拉式菜单，Application Expert 能够带领开发人员一步步地创建一个应用程序。

1.4　Delphi 5 窗体与组件应用实例

本节通过一些例子介绍 Delphi 5 的窗体与组件应用编程方法。

窗体和可视化组件是应用程序的界面部分，界面的美观性和友好性直接影响程序的质

量和使用方便性。用户界面的制作是与窗体和组件的属性、事件与方法密切相关。

1.4.1 窗体自动缩放程序

窗体自动缩放程序可以根据给定的比例大小自动调整窗体的大小。在窗体上放置两个按钮，将它们的 Name 属性设为 ScaleButton 和 RestoreButton，Caption 属性分别设为 Scale 和 Restore，然后放置一个 Label1 和一个 SpinEdit1，用于设定缩放比例。

双击 ScaleButton 按钮，创建 ScaleButtonClick 过程，用于控制比例缩放窗体大小：

```
procedure TForm1.ScaleButtonClick(Sender: TObject);
begin
  AmountScaled := SpinEdit1.Value;  {取得缩放大小}
  ScaleBy (AmountScaled, 100);    {按百分比缩放}
  ScaleButton.Enabled := False;
  RestoreButton.Enabled := True;
end;
```

注意 AmountScaled 变量存储缩放大小，应在程序说明部分说明。

双击 RestoreButton 按钮，创建 RestoreButtonClick 过程，用于恢复上一次的大小：

```
procedure TForm1.RestoreButtonClick(Sender: TObject);
begin
  ScaleBy (100, AmountScaled);  {相对于前一次的反缩放}
  ScaleButton.Enabled := True;
  RestoreButton.Enabled := False;
end;
```

运行这个程序，你可以随意地缩放窗体。

1.4.2 组件随窗体大小自动调整程序

在开发应用程序时，用户经常要遇到这样的问题：当窗体大小发生变化时，窗体上组件大小和位置可能发生混乱，甚至有看不见组件的情况，这是因为没有建立组件随窗体大小变换自动调整代码所致。下面，我们给出一个窗体上组件大小与位置随窗体大小变化自动调整程序，用户将这段程序稍作修改就可以应用到自己的程序中。

建立一个新窗体，在窗体上放置两个按钮组件 Button1 和 Button2，它们的 Caption 属性设置为图 1.3 所示的字符串。

双击 HelloButton，创建 HelloButtonClick 过程，用于修改窗体颜色和按钮文本：

```
procedure TForm1.HelloButtonClick(Sender: TObject);
begin
  color:=clBtnface;
  MessageDlg ('Hello, guys', mtInformation, [mbOK], 0);
```

```
    HelloButton.Caption := '谢谢您!';
  end;
```

单击窗体激活 FormClick 事件过程：

```
  procedure TForm1.FormClick(Sender: TObject);
  begin
    color:=clred;
    MessageDlg ('You have clicked outside of the button',
      mtWarning, [mbOK], 0);
  end;
```

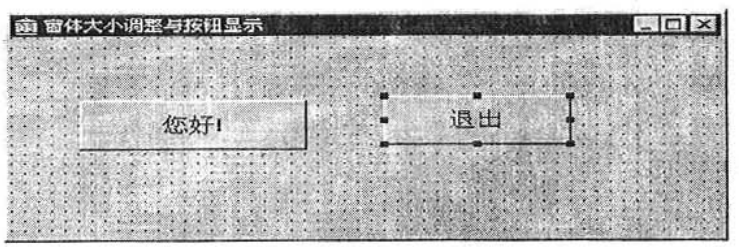


图 1.3　组件随窗体大小自动调整

FormResize 过程在窗体大小改变或窗体第一次显示时被调用。使用 FormResize 过程可以完成窗体上组件的大小和位置的自动调整：

```
  procedure TForm1.FormResize(Sender: TObject);
  begin
    HelloButton.Top :=
      Form1.ClientHeight div 2-HelloButton.Height div 2;  {确定按钮的顶部}
    {确定按钮的左部}
    HelloButton.Left :=
      Form1.ClientWidth div 2-(HelloButton.Width+closeBtn.Width) div 2;
    closeBtn.Top :=
      Form1.ClientHeight div 2 - closeBtn.Height div 2;
    closeBtn.Left :=
      HelloButton.Left+HelloButton.Width+ closeBtn.Width div 4;
  end;
```

1.4.3　进度指示程序

打开一个新项目和一个新窗体，给窗体上放一个计量器(Gauge)组件，计量器组件在组件模板的 Sample 组件页上。在 Object Inspector 中将 Gauge1 的 Progress 属性设置为 100，计量器使用 0 到 100 的百分点来测量进度量，并根据进度大小移动指针。在窗体中放入两个按钮 Button1 和 Button2，。双击 Button1，激活 Button1Click 事件，并加入如下代码：

```
    Procedure   TForm1.Button1Click(Sender:TObject);
```

```
Var
    B:byte;
begin
  if gauge1.progress=100 then   gauge1.progress:=0;
  B:=gauge1.progress+10;
  gauge1.progress:=B;
end;
```

这里用 "if gauge1.progress=100 then　gauge1.progress:=0;"语句来使计量器 Gauge1 复位。

双击 Button2，激活 Button2Click 事件，并加入如下代码：

```
Procedure  TForm1.Button2Click(Sender:TObject);
Var
    B,C:byte;
begin
  C:=gauge1.progress;
  B:=C-10;
  gauge1.progress:=B;
  if B=0 then
   begin
     B:=100;
     gauge1.progress:=100;
   end;
end;
```

其中，"if B=0 then　gauge1.progress:=100;"的意义是：当变量 B 等于 0 时，则把变量 B 和 gauge1.progress 设为 100，以免变量 B 和计量器的 Progress 属性值降到 0 以下。

执行这个程序，不断单击 Button2 按钮，使计量器降到 0%，然后单击 Button1 按钮，则计量器从 0%变到 100%。如果再次单击，计量器则从 100%变到 0%，如果再单击 Button2 按钮，计量器又回到 100%。

1.4.4　Delphi 的 Frames

本节介绍 Delphi 5 的新的可视化的 Container 类，并通过示例让用户理解其功能和使用编程。

Delphi 5 引入了一个新的可视化的 container 类 TFrame，用于表示一个重要的进行快速应用开发的高级应用。TFrame 提供给开发者一可视地配置一组组件以及容易、方便地在应用中重用这个配置的能力。这种能力能够使 Delphi 5 的 IDE 进行重新设计而扩展利用 Frames。我们首先讨论 Frames 的一般概念与功能及其属性方法，然后介绍 Frame 的使用与编程，最后用一个例子介绍了 Frame 的创建和设计等，包括如何加 Frames 到组件模板和对象仓库中等等。

使用 Frames 有两个主要的好处：一个是在某些情况下 Frames 能够动态地缩减需要存

储在工程中的资源的数量；另一个也是最重要的，Frames 允许你可视地创建可以被复制和扩展的对象。在某些方面，Frames 具有与组件模板类似的功能，即可视化设计。它们的不同点在于，Frames 是一个类的实例，是可以改变的，而组件模板则是一些组件的集合，对组件模板的修改将不会对创建的对象有什么作用。

1) 创建 Frame

下面的步骤介绍如何创建一个 Frames 对象：

(1) 选择 File | New Application 命令创建一个新的工程。

(2) 选择 File | New Frame 创建一个新的 Frame。然后在这个 Frame 上放置三个 labels 和三个 DBEdits 组件，放置一个 DBNavigator 和一个 DataSource，如图 1.4 所示。

设置三个 label 的 caption 属性分别为 ID, First Name 和 Last Name, 设置三个 DBEdit 和 DBNavigator 的 DataSource 属性为 DataSource1。

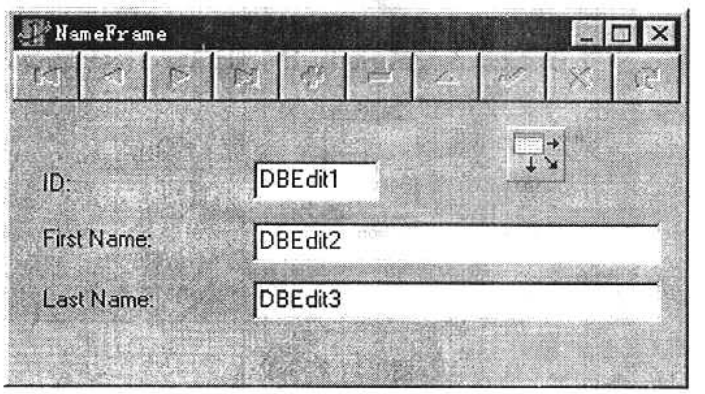


图 1.4 一个简单的 Frame

(3) 选择 Frame 对象，设置 Name 属性为 NameFrame。

(4) 选择 File | Save As 保存为 NAMEFRAM.PAS。

到此，我们已经创建了一个 Frame。下面介绍如何使用这个 Frame。

2) 使用 Frame

一个 Frame 实质上是一个组件，但它的使用不同于组件模板上的其它组件。

(1) 选择上面创建的 Form1。

(2) 加入两组 Frame 到窗体中，一个在另一个的上面。设置第一个 Frame 的 caption 为 Customers，第二个为 Employees。

(3) 增加 Frame。使用组件板上的 Frame 组件并放在 Customers Frame 上。

(4) 选择 NameFrame，Frame 出现在 Customers Frame 内。重复这个过程放置 Frame 在 Employees Frame 内。你可以制定每个 Frame 的大小和位置，结果如图 1.5 所示。

(5) 放置两个 Table 组件在窗体上，设置两个 Table 组件的 DatabaseName 属性为 IBLocal，Table1 的 TableName 设置到 CUSTOMER，Table1 的 TableName 设置到 EMPLOYEE。设置两个 Table 的 Active 属性为 True。

(6) 选择 Customers Frame 的 DataSource，设置 DataSet 属性为 Table1。通常你不能直接选择组件内的对象，但 Frame 是特殊的，你可以选择出现在 Frame 内的任何对象和

设置它们的属性。重复这个操作，设置 Employees Frame 中的 DataSource 的 DataSet 属性为 Table2。

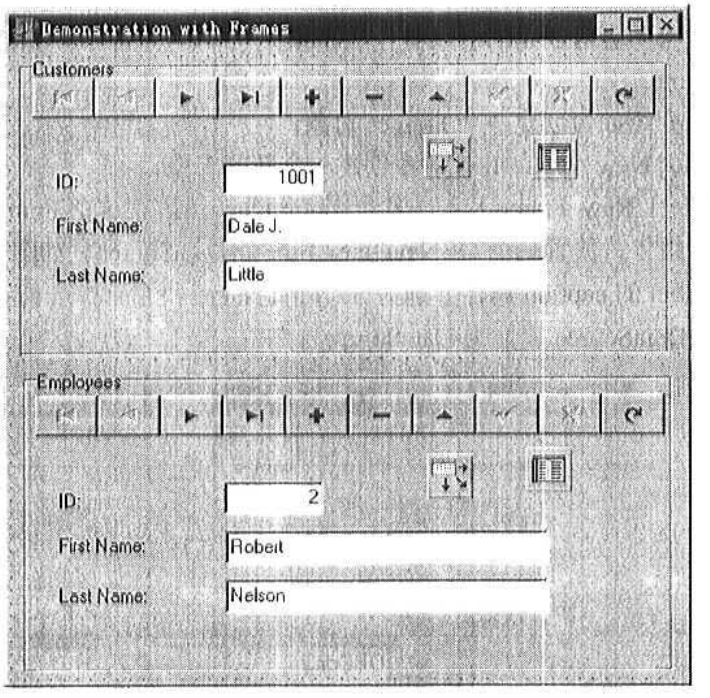


图 1.5　窗体设计

(7) 连接所有的 DBEdit，指定 Customers Frame 上的 DataField 属性分别为 CUST_NO，CONTACT_FIRST 和 CONTACT_LAST。对 Employees Frame，设置 DataField 分别为 EMP_NO，FIRST_NAME 和 LAST_NAME。

(8) 保存文件，然后运行该程序。

3) Frame 与继承

到此，我们已经看到了使用 Frame 的一些好处。当然，如果在各种不同情况下使用同一个 Frame，然后改变 Frame 的实例，则 Frame 的强大功能就会更为明显。例如，如果希望改变 NameFrame 为只读的，只须简单地原始 Frame，那么每一个 Frame 的实例就会立即继承所有的修改。

下面的过程演示了这个功能：

(1) 使用前面创建的窗体，按[Shift]+[F12]并选择 NameFrame。

(2) 设置 DataSource 的 AutoEdit 属性为 False.

(3) 选择 DBNavigator，展开 VisibleButtons 属性，设置 nbInsert，nbDelete，nbEdit，nbPost 和 nbCancel 为 False。

(4) 这时注意到 NameFrame 的后代类都继承了你刚才所做的变化。

下面介绍重载 Frame 包含的组件属性。

Frame 的特点之一就是可以改变与 Frame 内对象关联的属性和事件句柄。这些改变重载了继承的值，而且后续的变化不会影响继承的值。下面的步骤演示了这个过程：

(1) 选择 Customers Frame 中的"ID" label 组件，改变它的 Caption 属性为 Customer No:，同样改变 Employees Frame 中的"ID" label 组件为 Employee ID:。

(2) 按[Shift]+[F12]，选择 NameFrame，改变 ID label 组件的 Caption 属性为 Identifier。

(3) 返回到主窗体，注意到 label 的 Caption 属性并没有改变为 Identifier，而是仍然使用它们重载的值。

下面介绍 Frame 包含的对象事件句柄。

包含在 Frame 中的对象常常有事件句柄。尽管事件具有方法指针类型的简单属性，但是它们的处理不同于其它属性类型。首先我们开始为 Frame 对象定义一个事件句柄。这个 Frame 包含两个 button，分别为 Help 和 Done，它们有 OnClick 事件句柄，代码如下：

```
procedure TTwoButtonFrame.Button1Click(Sender: TObject);
begin
  if (TComponent(Sender).Tag = 0) or
       (Application.HelpFile = '') then
    MessageBox(Application.Handle,'Help not available',
                              'Help',MB_OK)
  else
    Application.HelpContext(TComponent(Sender).Tag);
  end;

procedure TTwoButtonFrame.Button2Click(Sender: TObject);
var
  AParent: TComponent;
begin
  AParent := TComponent(Sender).GetParentComponent;
  while not (AParent is TCustomForm) do
    AParent := AParent.GetParentComponent;
    TCustomForm(AParent).Close;
end;
```

Frame 中的对象的事件句柄都是 published 类型方法。你可能会注意到与 Frame 关联的事件句柄引入了一个有意思的行为对象（artifact）。特别地，Self 代表 Frame 而不是窗体 Form1。因此对 Button2Click ，不可能简单地通过 Close 方法关闭窗体。当一个不适合的方法被请求时，编译器认为你想将方法作用于自身，因为 TFrame 对象没有 Close 方法，如果简单地请求 Close，编译器产生一个错误。

由于上面的例子把 Frame 设计在窗体内部，所以使用 Frame 的 GetParentComponent 方法攀升到 Frame 嵌套的包含层次关系。一旦发现 TcustomForm 实例(可以是 TForm 的后代，也可以是一个基于 TcustomForm 定制的窗体)，则实例将被用于请求窗体的 Close 方法。

下面介绍重载包含的对象事件句柄。

Frame 后代不使用继承来为嵌入在其中的对象请求事件句柄，而是直接地请求祖先 Frame 的方法。例如，如果你放置名为 TwoButtonFrame 的 Frame 在窗体上(见图 1.6)，那

么双击它，Delphi 将会生成下面的代码：

```
procedure TForm1.TwoButtonFrame1Button2Click(Sender: Object);
begin
    TwoButtonFrame1.Button2Click(Sender);
end;
```

这里 TwoButtonFrame1 是一个 TwoButtonFrame 的 Frame 后代，Button2Click 是 Done 按钮的事件句柄。结果代码请求了原始的事件句柄，并传递给它 Sender 参数（Frame 实例）。这意味着事件处理引入了另外有意思的特征，特别是在这种情况下，Sender 一般不是 Self 对象的成员。实际上，Sender 通常是 Form 对象的成员，而 Self 是 Frame 对象。

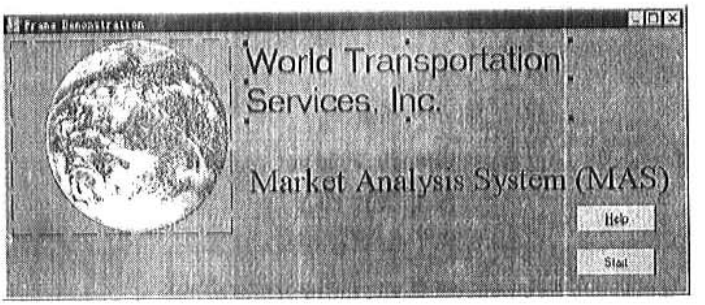


图 1.6　重载包含的对象事件句柄演示窗体

下面的过程说明了 TwoButtonFrame 后代的重载事件句柄。这种情况下，原来的行为已被注释掉了，新的行为完全替换了 Done 按钮。

```
procedure TForm1.TwoButtonFrame1Button2Click(Sender: TObject);
begin
    with TForm2.Create(Self) do begin
      ShowModal;
      Release;
    end;
     //    TwoButtonFrame1.Button2Click(Sender);
end;
```

按钮的 caption 被重载，显示为 Start。

4) 使用 Frame 保存资源

图 1.6 中包含了两个 Frame，一个是上面讨论的 TwoButtonFrame，另一个是企业日志 LogoFrame。LogoFrame 出现在 FramDemo 工程的多个窗体上。使用 Frame 显示日志等信息的作法是在窗体上放置一个 Image 对象，使用 Frame 的目的可以大大减少编译的。EXE 文件中的资源，从而使可执行文件体积减小。使用 Frame 生成的 DFM 文件文本部分代码为：

```
inline LogoFrame1: TLogoFrame
    Left = 6
    Top = 6
```

```
        Width = 211
        Height = 182
        inherited Image1: TImage
        Width = 211
        Height = 182
        end
    end
```

如果直接在窗体上使用 Image 组件，则 DFM 文件包含了日志的整个代码，而且多个窗体同时使用这个日志时会重复使用这个资源。使用 Frame 时，资源仅定义一次。

对单个、体积小的应用，直接使用 Frame 组件是比较合适的，但对大型的工程，或者在多个应用中使用同一个 Frame，则要求更容易处理。幸运的是，Delphi 允许你放置单个 Frame 到组件板上，并允许这些 Frame 重复使用而不需要其它额外的步骤。Frame 也可以放进 Object Repository 中，允许被方便地复制。下面分别介绍它们的使用。

放置特定的 Frame 到 Component palette 上非常方便。使用 Component palette 上标准页上的 Frame 组件需要四个步骤，这限制了放置工程中已经存在的 Frame。放置特定的 Frame 到 Component palette 上需要下面的步骤：

(1) 保存已制作的 Frame 到磁盘。如果要使用这个 Frame 到多个应用场合，建议你把该 Frame 保存到一个永久的路径下，例如：c:\Program Files\Borland\DelphiFrames，以防止卸载 Delphi 时删除这个 Frame 文件。

(2) 选择该 Frame 并右击它，在弹出式菜单中选择 Add to Palette。Delphi 显示 Component Template 信息对话框。

(3) 在 Component name 域定义 Frame 组件的名字，在 Palette page 域选择 Frame 出现的 Component palette 页。如果你已经创建了一个 24×24 像素、16-color 的图标文件，可单击 Change 按钮并选择这个.BMP 文件，单击 OK 确定。

到此就完成了字定义 Frame 组件的安装。使用这个组件，只需选择 Component palette 上该 Frame 图标，然后放置在想要使用这个 Frame 后代类的窗体上。

加 Frame 到 Object Repository 中，可以很容易地复制它到一个新工程中。使用 Object Repository ，其重要性是使用 Object Repository 提供的继承能力来放置一个继承的 Frame 到新的工程中，因此你需要维护 Frame 与它的祖先的关系。

把 Frame 加入到 Object Repository 中需要下面的步骤：

(1) 保存 Frame。另外保存该 Frame 到 Delphi 的 OBJREPOS 路径或一个共享的路径下。保存在一个共享的路径下对使用共享对象 repository 特别有好处，可允许多个开发者共享这个 Frame。

(2) 右击该 Frame，在弹出式菜单中选择 Add To Repository，出现 Add To Repository 对话框。

(3) 填充 Add To Repository 对话框，最后单击 OK 即可。

从 Object Repository 中使用一个 Frame 的步骤如下：

(1) 选择 File | New 命令。

(2) 选择 Object Repository 页。