徐新宇 编著
希望图书创作室 审订

Delphi 3 编程指南

下编

学报出版社

Delphi 3 编程指南

（下编）

徐新华　编著
希望图书创作室　审订

宇航出版社

图书在版编目（CIP）数据

Delphi 3 编程指南/徐新华编著. -北京：宇航出版社，1998.6
ISBN 7-80144-086-2

Ⅰ.D… Ⅱ.徐… Ⅲ.Delphi 语言 程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（98）第 04746 号

宇航出版社出版发行
北京市和平里滨河路 1 号（100013）
发行部地址：北京阜成路 8 号（100830）
北京媛明印刷厂印刷
新华书店经销

1998 年 6 月第 1 版　　1998 年 6 月第 1 次印刷
开本 787 × 1092　1/16　　印张：62.375　　字数：1444 千字
印数：1-5000 册　　总定价：80.00 元（上、下编）

目 录

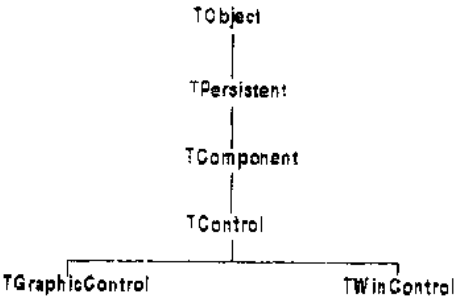
下 编

第 28 章　元件的公共特性和方法

从本章开始将逐一详细介绍 Delphi 3 的元件，尽管这些元件千差万别，但它们都有一些相同或相似的特性和方法，为了节省篇幅，这里首先介绍一些公共特性和方法，在以后将只介绍每个元件特有的属性和方法。

28.1　VCL 概述

VCL(Visual Component Library 的缩写)是 Delphi 3 的核心，VCL 是完全面向对象的，VCL 中的所有元件和对象都存在着继承与被继承的关系，下图是 VCL 的继承关系示意。



从示意图可以看出，最顶层的是 TObject 类，它是一切元件和对象的基类。

TPersistent 类是 TObject 类的下一级继承者，它是一个抽象类，主要用于为它的继承者提供对流的读写能力。

TComponent 类是 TPersistent 类的继承者，这个类是 VCL 中所有元件的基类，TComponent 类中定义了所有元件最基本的行为

尽管所有元件都是从 TComponent 类继承下来的，但直接继承的只有几个非可视的元件，如 TTimer 元件和 TDataSource 元件等，其它元件则是从 TComponent 类的下级 TControl 类继承下来的，从 TControl 类继承下来的元件是可视元件，这些元件也称为控制。TControl 类定义了 VCL 中所有控制的基本属性包括特性、方法和事件等。

TWinControl 和 TGraphicControl 这两个类都是从 TControl 类继承下来的，从 TWinControl 类继承的控制主要是用于窗口控制如按钮、对话框和列表框等，这些控制可以把它们当作是一些小窗口，这些小窗口有自己的窗口句柄，占用 Windows 的资源，并且允许接受用户输入。而从 TGraphicControl 类继承的元件没有窗口句柄，也不占用 Windows 资源，也不能接受键盘的输入，使用这一类元件的好处在于节约资源，例如 TLabel 元件、TSpeedButton 元件等。

VCL 的面向对象还体现在它的可扩展性，可以选择其中一个元件作为基类，派生出一个新的类(元件)，并把新创建的元件加入到 VCL 中。

28.2 TObject 类

TObject 类中主要定义了一些操纵对象的基本方法，其中有的是类方法，用于返回对象的类型信息，有的是虚方法，用于在派生类中重载，有的仅用于 Delphi 自身调用。

ClassInfo 类方法

声明： class Function ClassInfo: Pointer;

这个类方法用于返回一个指向对象的运行期类型信息表(RTTI)的指针，包括对象自身、对象的基类以及对象中公共特性的类型信息。

注意：类型信息一般是由 Delphi 内部使用的，程序员一般不必与它打交道。

ClassName 类方法

声明： class Function ClassName: ShortString;

这个类方法用于返回一个对象实例的类名。注意：假设声明了一个基类的变量，并且用此变量来引用派生类的对象实例，从语法上讲是允许的。但是，如果用此变量来调用 ClassName，返回的仍然是派生类的类名。

ClassNameIs 类方法

声明： class Function ClassNameIs(const Name: string): Boolean;

这个类方法用于判断对象的类名是否与 Name 参数匹配，如果匹配，就返回 True。

ClassParent 类方法

声明： class Function ClassParent: TClass;

这个类方法用于返回类的基类，对于 TObject 来说，结果肯定是 NIL，因为 TObject 已经是最顶层的了，它没有父类。应用程序一般不要直接调用此方法。

CleanupInstance 过程

声明： Procedure CleanupInstance;

这个过程用于清除对象中长字符串、可变类型的变量，它把长字符串清为空，把可变类型的变量设为 Unassigned。注意：一般不需要直接调用此过程，因为调用 Free 时会自动调用 CleanupInstance。

Create 构造

声明： constructor Create;

这是个常用的方法，用于构造类的对象实例，并且对其进行初始化。

DefaultHandler 过程

声明： Procedure DefaultHandler(var Message); virtual;

如果一个对象没有提供对某个消息的处理句柄， DefaultHandler 能提供对某消息的默认处理。派生类可以重载这个方法，例如， TWinControl 就重载了 DefaultHandler，并且换名为 DefWindowProc。

Destroy 析构

声明： destructor Destroy; virtual;

Destroy 被称为析构，用于删除对象实例。不过，我们建议不要直接调用 Destroy 来删除对象实例，而是调用 Free，因为 Free 在删除对象实例之前会检查对象实例是否存在，这

样就能避免对值为 NIL 的指针进行操作。

Destroy 被声明为虚拟的，派生类通常要对它重载，针对特定的对象做一些“清场”的工作。在派生类中重载 Destroy 时，一般在最后要加上这么一句：Inherited Destroy;

Dispatch 过程

声明：Procedure Dispatch(var Message);

这个过程用于调用对象的消息处理方法，究竟调用哪个消息处理方法，由 Message 参数指定的消息来决定。如果对象中没有找到处理该消息的方法，Dispatch 就从父类中查找，如果一直到最顶层还没有找到，就调用 DefaultHandler 。

注意：Message 是个无类型的参数，从语法上讲，可以传递任何类型的数据，不过，最好是 TMessage 类型，这是个 Object Pascal 语言的记录类型。

FieldAddress 函数

声明：Function FieldAddress(const Name: ShortString): Pointer;

这个函数用于返回公共的(Published)字段的地址，Name 参数指定字段名，如果指定的字段不存在，函数返回 NIL 。注意：这个函数通常是 Delphi 内部调用的，在程序中，可以直接用字段名访问字段。

Free 过程

声明：Procedure Free;

这个过程用于删除对象实例，释放对象占用的内存。正如前面提到的那样，Free 能够检查对象实例是否存在，因此，调用 Free 是安全的。

FreeInstance 过程

声明：Procedure FreeInstance; virtual;

FreeInstance 是与 NewInstance 配对使用的，NewInstance 用于建立一个新的对象实例，而 FreeInstance 用于删除 NewInstance 建立的对象实例。

注意：FreeInstance 一般是由析构自动调用的，一般不要直接调用它，除非派生类重载了 NewInstance ，也要相应地重载 FreeInstance

GetInterface 函数

声明：Function GetInterface(const IID: TGUID; out Obj): Boolean;

这个函数检索指定对象的接口，一般是由 Delphi 内部调用的。

GetInterfaceEntry 类方法

声明：class Function GetInterfaceEntry(const IID: TGUID): PInterfaceEntry;

其中，PInterfaceEntry 的声明如下：

```
Type
    PInterfaceEntry = ^TInterfaceEntry;
    TInterfaceEntry = Record
            IID: TGUID;
        VTable: Pointer;
        iOffset: Integer;
    End;
```

这个类方法返回类的指定接口的入口指针，接口由 IID 参数指定。注意：这个类方法

一般是由 Delphi 内部调用的。

GetInterfaceTable 类方法

声明： class Function GetInterfaceTable: PInterfaceTable;

其中， PInterfaceTable 的声明如下：

```
Type
    PInterfaceTable = ^TInterfaceTable;
    TInterfaceTable = Record
        EntryCount: Integer;
        Entries: array[0..9999] of TInterfaceEntry;
    End;
```

这个类方法返回一个指针，这个指针指向由类的所有接口组成的列表，不包括基类的接口。如果要获得基类的接口列表，首先要调用 GetParentClass 返回基类，再由基类调用 GetInterfaceTable 。

InheritsFrom 类方法

声明： class Function InheritsFrom(AClass: TClass): Boolean;

这个类方法用于判断两个对象类型的继承关系，如果 AClass 参数指定的类是本对象类型的基类，这个类方法就返回 True ，否则就返回 False 。

InitInstance 类方法

声明： class Procedure InitInstance(Instance: Pointer): TObject;

这个类方法一般不需要直接调用，它是由 NewInstance 自动调用的，用于对新建立的对象实例初始化。

注意： InitInstance 不是虚拟方法，派生类不能重载它，但派生类可以重载 NewInstance ，在重载 NewInstance 时要确保最后一句是调用 InitInstance 。

InstanceSize 类方法

声明： class Function InstanceSize: Longint;

这个类方法一般不直接调用，而是在重载 NewInstance 时，调用这个类方法返回创建类实例所需的字节数。

MethodAddress 类方法

声明： class Function MethodAddress(const Name: ShortString): Pointer;

这个类方法返回某个公开的(Pblished)的方法的地址，方法名由 Name 参数指定。如果指定的方法不存在，就返回 NIL 。一般不直接调用。

MethodName 类方法

声明： class Function MethodName(Address: Pointer): ShortString

这个类方法与 MethodAddress 恰好相反，返回一个指定地址的方法的名称。如果给定的指针并没有指向某个公共的方法，这个类方法返回空字符串。

NewInstance 类方法

声明： class Function NewInstance: TObject; virtual;

这个类方法用于创建类的对象实例。注意：构造(Constructors)会自动调用 NewInstance 。派生类可以重载 NewInstance ，相应地，派生类必须重载 FreeInstance ，因为这两个类

方法是配对使用的。

SafeCallException 类方法

声明： Functio SafeCallException(ExceptObject:TObject;ExceptAddr:Pointer):Integer;virtual;

这个类方法主要用于供 COM 和 OLE 类重载，以处理异常。

28.3　TPersistent 类

TPersistent 类是一个抽象类，它提供了对象的赋值和读写流的能力。从 TPersistent 类继承下来的对象，可以相互赋值，并且能从流中访问对象的特性。

TPersistent 类的大部分方法是从 TObject 类继承的，因此，这里只介绍几个 TPersistent 类特有的方法。

Assign 过程

声明： Procedure Assign(Source: TPersistent);

这个过程用于把 Source 参数指定的对象赋值给自己。Source 参数的类型是 TPersistent，也就是说，只要是 TPersistent 的派生类都是可以相互赋值的，只有一些直接从 TObject 类继承下来的类如 TComObject、TAutoObject 等才是不可赋值的。

需要注意的是，对象之间的相互赋值是有前提的，那就是两个对象必须是赋值相容的，换句话说，两个对象的类型要么完全一致，要么具有继承关系。如果不符合赋值相容的条件，调用 Assign 将触发 EConvertError 异常。

也许有的读者要问，能不能把 Object Pascal 语言的赋值语句用于对象之间的相互赋值呢？一般是不可以的，除非用于某些特定的属性之间的赋值，这些特定的属性是指那些本身又是对象的属性。

AssignTo 过程

声明： Procedure AssignTo(Dest: TPersistent); virtual;

这个过程与 Assign 恰好相反，用于把自己赋值给 Dest 参数指定的对象。

DefineProperties 过程

声明： Procedure DefineProperties(Filer: TFiler); virtual;

这是个虚拟方法，TPersistent 的派生类重载它是为了把对象的非公开数据写到流中，Filer 参数指定要写的流。

Destroy 过程

声明： Procedure Destroy; override;

TPersistent 的 Destroy 重载了 TObject 的 Destroy，用于删除 TPersistent 的派生类的对象实例。注意：我们建议读者用 Free 来删除对象实例，这样更安全些。

GetNamePath 函数

声明： Function GetNamePath: string; dynamic;

这个函数返回对象在 Object Inspector 中显示的名称，对于元件来说就是元件名。一般不要直接调用这个函数。

GetOwner 函数

声明： Function GetOwner: TPersistent; dynamic;

这个函数返回一个对象的拥有者(Owner)，对于元件来说，就是元件的 Owner 属性。

28.4 TComponent 类

TComponent 类是 Delphi 所有元件的抽象基类，它提供了元件的基本特征：元件可以出现在元件选项板上，可以被加到 Form 上，可以拥有和管理其它元件，增强的流和归档能力，可以作为 ActiveX 控制的外套。

TComponent 类是个抽象类，不要直接创建它的实例。尽管 Delphi 的所有元件都是从 TComponent 类继承下来的，但直接继承于 TComponent 类的只是少数非可视的元件，大部分元件是从 TComponent 类的派生类 TControl 继承下来的。

ComObject 特性

声明： Property ComObject: IUnknown;

这个只读的特性返回支持 COM 接口的 VCL 元件的接口。

ComponentCount 特性

声明： Property ComponentCount: Integer;

这个只读的特性返回元件拥有的子元件的数目，例如，对于 TForm 元件来说，它的 ComponentCount 属性就是 Form 上元件的个数。

元件拥有的子元件放在 Components 特性中，这是个由所有子元件组成的数组，其索引号从 0 开始，程序示例如下：

```
Procedure TForm1.Button1Click(Sender: TObject);
Var
    I: Integer;
Begin
    For I := 0 To ComponentCount -1 Do
        If Components[I] Is TButton Then
            TButton(Components[I]).Font.Name := 'Courier';
    Edit1.Text := IntToStr(ComponentCount) + ' 元件';
End;
```

上例中，把 Form1 上所有的 TButton 元件的字体名改为 Courier，然后把 Form1 上的元件数转换成字符串后显示在编辑框 Edit1 中。

ComponentIndex 特性

声明： Property ComponentIndex: Integer;

这个只读的属性返回某个子元件的索引号，程序示例如下：

```
Procedure TForm1.Button1Click(Sender: TObject);
Begin
    Edit1.Text := IntToStr(Button1.ComponentIndex);
End;
```

上例中，把 Form1 上的 Button1 元件的索引号值转换成字符串后显示在编辑框 Edit1 中。

Components 特性

声明： Property Components[Index: Integer]: TComponent;

这个只读的特性是由所有子元件组成的数组。

注意：不要把 Components 特性与 Controls 特性混淆起来，前者包含的是元件，后者包含的是某个控制上的子控制。

ComponentState 特性

声明： Property ComponentState: TComponentState;

这个只读的特性返回元件当前的状态。TComponentState 是个集合类型，在 Classes 单元中它是这样声明的：

要判断元件是否在某个状态，程序可以这样写：

```
If (csDesigning in ComponentState) Then
    ShowMessage('Please Register This Component');
```

ComponentStyle 特性

声明： Property ComponentStyle: TComponentStyle;

这个特性设置元件的风格，如果设为 csInheritable(默认)，表示这个元件是可继承的，一般不设为 csCheckPropAvail 。

DesignInfo 特性

声明： Property DesignInfo: Longint;

注意：这个特性是 Delphi 内部使用的，用于返回 Form 设计器的有关信息。

Name 特性

声明： Property Name: TComponentName;

这个特性用于设置元件的名字，当从元件选项板上选择一个元件放到 Form 上时，Delphi 给这个元件取一个默认的名字，如 Button1 、Edit1 等，当然可以重新给元件取一个名字。注意：在同一个应用程序中元件的名字不能重名，如果不是很有必要，不要在运行期修改期间元件的名字。

Owner 特性

声明： Property Owner: TComponent;

这个只读的特性返回元件的拥有者。拥有和被拥有这种关系，是在创建元件的实例时确定的，也就是调用 Create 时确定的。假设 Create 是这样调用的：

```
Button1 := TButton.Create(TForm);
```

凡是 TForm 对象就成为 TButton 对象的拥有者。

当拥有者被删除时，所有被拥有的元件也将同时被删除。例如，当 Form 被删除时，Form 上的元件都将被删除。要注意的是，元件与它的拥有者之间的关系不同于控制与它的子控制之间的关系，Owner 通常是 Form 和 Form 上的元件的关系，而控制与子控制通常是一个对话框与对话框上某个控制的关系。

Tag 特性

声明： Property Tag: Longint;

这个特性也是相当有用的，通常用于给一组元件加上相异的序号，这样就可以在运行期间识别它们。例如，假设 Form 上有十个按钮，它们有一个共同的事件句柄，为了知道到底是哪个按钮被按下，可以事先给这十个按钮编号，然后用 Tag 特性判断是哪个按钮被按下，程序示例如下：

```
Procedure TForm1.ButtonClick(Sender: TObject);
Begin
    If (Sender as TButton).Tag = 1 Then (Sender as TButton).Caption := 'My tag is 1';
End;
```

Create 构造

声明： constructor Create(AOwner: TComponent); Virtual;

Create 做了这么几件事情：创建和初始化元件的实例，指定元件的拥有者(由 AOwner 参数指定)，把 ComponentStyle 特性设为 csInheritable 表示此元件是可继承的。Create 是虚拟的，TComponent 的派生类可以重载。

要注意的是，在 Delphi 中一般不直接调用 Create 来创建元件的实例，因为，当从元件选项板上选择一个元件放到 Form 上时，Delphi 自动创建这个元件的实例。同样，也不需要直接调用 Destroy 来删除元件的实例，因为，当应用程序终止时，会自动删除元件的实例。

如果要在运行期间动态地引入元件，就要显式地调用 Create，并指定元件的拥有者。

Destroy 析构

声明： destructor Destroy; virtual;

Destroy 的作用与 Create 正好相反，用于删除元件的实例，包括该元件所拥有的元件。不过最好调用 Free 来删除元件的实例，因为 Free 会检查对象是不是 NIL。

正如 Create 一样，一般不需要调用 Destroy 或 Free 来删除元件的实例，因为当应用程序终止时，会自动删除应用程序(Application)所拥有的 Form，而 Form 又会删除 Form 上的元件。当然，如果元件是在运行期间动态引入的，需要相应地调用 Destroy 或 Free 来删除元件的实例。另外，如果程序中还创建了其它不属于元件的对象，当这些对象不再需要用到时，就要及时地调用 Destroy 或 Free 删除它们，否则，它们所占的内存就不会释放，除非整个应用程序终止。对于 Form 来说。最好调用 Release 来释放它的实例。

注意：千万不要在元件的事件句柄中删除元件自身或元件的拥有者，否则会导致意想不到的结果。

DestroyComponents 元件

声明： Procedure DestroyComponents;

这个过程删除所有拥有的元件，但不删除自身。注意：一般不需要直接调用 DestroyComponents，因为调用 Destroy 时会自动调用 DestroyComponents。

Destroying 过程

声明： Procedure Destroying;

此过程把元件及拥有的元件的 ComponentState 特性设为 csDestroying，表示元件将要被删除。注意：一般不需要直接调用 Destroying，因为调用 Destroy 时会自动调用 Destroying。

FindComponent 函数

声明：Function FindComponent(const AName: string): TComponent;

这个函数从元件所拥有的元件(实际上就是从 Components 数组)中查找一个元件，其名称是 AName 参数（不区分大小写）。如找到，就返回这个元件。

FreeNotification 过程

声明：Procedure FreeNotification(AComponent: TComponent);

这个过程通知 AComponent 参数指定的元件：本元件将要被删除了。例如，元件 A 被赋值给元件 B 的某特性，当元件 A 将要被删除时，就要通知元件 B。不过，如果元件 A 和元件 B 位于同一个 Form 内，通知是自动进行的。如果两个元件不在同一个 Form 内，并且存在依赖关系，才需要调用 FreeNotification。

FreeOnRelease 过程

声明：Procedure FreeOnRelease;

这个过程一般是 Delphi 内部使用的，用于释放 COM 对象的接口。

GetParentComponent 函数

声明：Function GetParentComponent: TComponent; dynamic;

这是个动态方法，派生类重载它是为了返回一个特定的父类。

HasParent 函数

声明：Function HasParent: Boolean; dynamic;

这个函数判断元件是否有“父”，“父”通常是元件所在的容器如 TForm、TPanel 等。

InsertComponent 过程

声明：Procedure InsertComponent(AComponent: TComponent);

这个过程向元件的 Components 数组中增加一个 AComponent 参数指定的元件，这个元件要么没有名字，要么与 Components 数组中现有的元件相异。

RemoveComponent 过程

声明：Procedure RemoveComponent(AComponent: TComponent);

这个过程从元件的 Components 数组中删除 AComponent 参数指定的元件。注意：在设计期，无论把一个元件加到 Form 上，还是从 Form 上移走一个元件，InsertComponent 或 RemoveComponent 都是自动被调用的。

28.5　TControl 类

TControl 类是从 TComponent 类继承下来的，作为 Delphi 中所有控制的基类。所谓控制，是指那些在运行期间可见的元件，例如，TButton 元件在运行期间就是个按钮，TEdit 元件在运行期间就是一个编辑框。对于控制来说，可以设置它的外观，包括尺寸、位置、