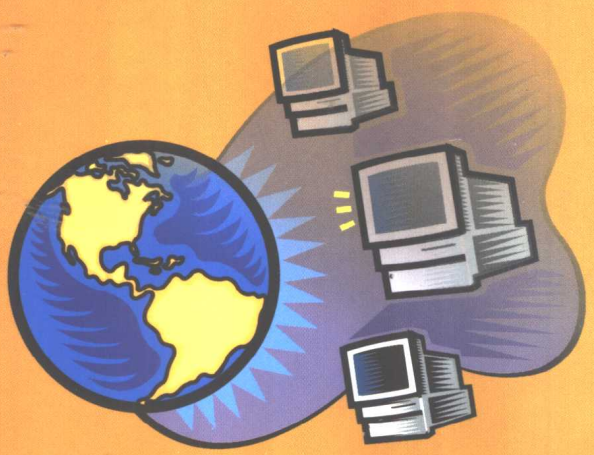
Delphi 分布式多层应用程序开发

陈 锐 编著



清华大学出版社
http://www.tup.tsinghua.edu.cn

Delphi 分布式多层应用程序开发

陈　锐　编著

清华大学出版社

(京)新登字 158 号

内 容 简 介

随着电子商务以及互联网的发展，企业规模的扩大，开发多层分布式应用和基于 Web 的应用已经成为软件开发的发展方向。本书就是针对如何利用 Delphi 开发以上的应用而写的。本书的重点有 3 个：Windows 平台下的 COM 控件开发；多层应用的开发；基于 Web 应用的开发。对于每一部分的内容，本书都有详细的介绍以及详尽的范例。

本书面向 Delphi 程序员，特别是对 Delphi 以及 Delphi 数据库编程比较熟悉但是不了解 COM/ DCOM/ MTS，MIDAS 等多层应用开发的程序员，借助本书可快速了解以上原理及基本开发方法。

图书在版编目(CIP)数据

Delphi 分布式多层应用程序开发/陈锐编著. - 北京：清华大学出版社，2002
ISBN 7-302-05129-1

Ⅰ. D... Ⅱ. 陈... Ⅲ. DELPHI 语言 - 程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字(2001)第 094227 号

出 版 者：清华大学出版社(北京清华大学学研大厦，邮编：100084)
http://www.tup.tsinghua.edu.cn
责任编辑：许振伍
印 刷 者：北京鑫丰华彩印有限公司
发 行 者：新华书店总店北京发行所
开　　本：787×1092　1/16　**印张**：17.75　**字数**：428 千字
版　　次：2002 年 3 月第 1 版　2002 年 3 月第 1 次印刷
书　　号：ISBN 7-302-05129-1/TP·3006
印　　数：0001～5000
定　　价：25.00 元

前　言

本书共分为6章，从COM开始逐步进行深入探讨，在每一章中对于所涉及的内容都会有一个详细的开发范例帮助读者掌握。

COM可以说是Windows下多层应用的开发基础。所以本书从COM基础知识开始，在第1章中讨论了COM、接口、GUID等基本术语，并且结合实例介绍了如何在Delphi中编程实现COM控件以及在程序中调用COM控件提供的服务。

第2章介绍的是开发ActiveX控件和如何通过Delphi提供的向导实现利用Web页面发布ActiveX控件，以及控件安全性和ActiveX控件版本升级问题的解决方法。

第3章开始涉及到Windows平台下的分布式开发，介绍的是DCOM/ MTS的开发。本章分为两个部分：第1部分重点在于介绍如何建立DCOM服务器程序以及如何对服务器端的DCOM控件进行配置；第2部分首先讨论MTS控件的一些特性，例如事务处理、无状态对象、基于角色的安全性等，然后给出一个开发基于MTS的Delphi应用程序的范例。

第4章介绍基于MIDAS的多层开发。MIDAS是Borland开发的多层应用开发引擎。本章从MIDAS原理开始，比较详细地介绍了开发基于MIDAS引擎的多层应用的方法。

第5章介绍了WebBroker开发。阐述了基于WebBroker的Web应用程序建立以及发布方法，并且提供了一个开发论坛程序的详细范例。读者还可以通过这一章了解Web应用程序的原理以及开发技巧。

第6章介绍InternetExpress编程。InternetExpress相比WebBroker，增加了XML处理功能。将InternetExpress和WebBroker结合起来可以开发专业的Web应用程序。本章介绍了InternetExpress控件的特性。在本章的最后是一个将二者结合起来开发在线书店程序的实现范例。

本书中的程序是在Windows 2000 Professional中文版 + Internet Explorer 5.5中文版 + Delphi 5环境下完成的。其中第5、6章中Web应用程序所使用的Internet服务器为Windows 2000 Professional中文版 + IIS 5.0。部分读者应该已经或者准备使用Delphi 6开发应用程序了。书中的部分范例作者移植到Delphi 6下并可以成功运行，因此对于使用Delphi 6的程序员同样有指导作用。

本书针对的是Delphi程序员，特别是对Delphi以及Delphi数据库编程比较熟悉但是不了解COM/DCOM/MTS、MIDAS等多层应用开发的程序员。借助本书可以使读者能够快速了解以上原理和基本开发方法。

由于作者水平有限，本书中难免有错误和疏漏。如果读者对这本书有任何意见、建议或者批评，欢迎到作者的主页 http://www.applevb.com 同作者探讨。

作　者

敬告读者：欢迎您对清华版图书提出批评和建议，此外，如果您还须要了解其他流行软件类的清华版计算机重点图书，可以访问 http://www.tupwq.com。

清华大学出版社流行软件编辑室

目　录

第 1 章　COM 基础

在本章中，将学习一些 COM 的基础知识和概念，例如，COM 对象、接口、COM 服务器、COM 服务器线程、GUID 等。然后还会介绍两个 COM 实例。

简单的说，COM(Component Object Model)是一项通过接口透明地传递封装数据的技术。这种接口可以是独立的模块、线程、进程，甚至机器。COM 对象是独立于语言和操作平台的，也就是说，使用 Delphi 编写的 COM 对象可以在 Windows 9x/NT 等平台上发布(有消息说，Microsoft 将把 COM 技术扩展到 Unix 平台上，但是现在 COM/DCOM/COM + 技术只能应用于 Windows 平台)。而且该控件可以使用 VC、VB、Visual Foxpro 等编程语言实例化。建立 COM 对象同建立 Delphi 中的类有一些相似并密切相关，但是也有很大的区别。在后面我们要接触的 ActiveX、DCOM、MTS、MIDAS 等技术都与 COM 有关。现在几乎所有的 Microsoft 的软件都是基于 COM 技术的，所以在掌握 Windows 下的分布式程序开发前须要首先掌握 COM 的一些基本知识。

1.1　COM 与 Object Pascal

1. 接口(Interface)

一个 COM 对象是实现一个或者若干个接口的对象，或者说 COM 对象借助接口来输出它所提供的服务。接口可以使调用 COM 对象的程序和 COM 对象的函数之间进行通信。下面是一个接口的定义。

```
type
  IDSort = interface
   Function fSort:Integer;
End;
```

了解接口的比较简单的方法是它多少等同于 Object Pascal 中的抽象类。接口被声明为 interface 类型，一般的惯例是：接口的名称从字母 I 开始。一个接口实例是不能够被直接创建的，如下所示。

```
var
   mID: IDSort;
begin
   mID: = IDSort.Create;
end;
```

但是接口并不是一个类，而是一个预先定义的协议。直接创建接口是非法的，接口中

定义的方法必须在类对象中被实现，如下所示。

```
type
  TMySort = class(TInterfacedObject,IDSort)
   Function fSort:Integer;
  end;
```

上面的代码从 TInterfacedObject 派生一个实现 IDSort 的类 TMySort。然后在程序的 implementatio部分实现 TMySort 的 fSort 方法，如下所示。

```
function TMySort.fSort:Integer;
    begin
    Result:=1;
end;
```

最后再通过 Create 建立 TMySort 类，如下所示。

```
Var
    FIn: TMySort
Begin
    FIn:=TMySort.Create;
    FIn.fSort;
    FIn.Free;
End;
```

使用接口要了解以下 4 点。

- **接口不是类**：通过类可以生成对象，而接口不能实例化，因为没有实现方法。接口中定义的方法是在类中实现的。
- **接口不是对象**：接口只是对象与调用者之间的协议。客户端访问对象时，只有一个接口指针用于访问接口中的内容。指针是不透明的，通过指针用户无法看到任何对象的内部细节，例如对象的状态信息。客户端只能调用接口函数，但这并不是一个限制，因为这个特性使得 COM 对象提供了地址透明性以及有效的二进制标准。
- **接口是惟一的**：每一个接口都有自己的 GUID，保证不会与其他接口发生冲突。
- **接口是不可变的**：接口没有版本，接口添加、删除或者修改功能之后成为了全新的接口，并指定新的接口标识符，因此，新接口和旧接口并不产生冲突。如果须要增强功能，可以通过派生的方法来实现新接口。

2. IUnknown 接口

所有的 COM 接口都从 IUnknown 直接或者间接继承。IUnknown 接口在 System. pas 单元中声明，声明代码如下。

```
IUnknown = interface
    ['{00000000-0000-0000-C000-000000000046}']
    function QueryInterface(const IID: TGUID; out Obj): HResult; stdcall;
    function _AddRef: Integer; stdcall;
    function _Release: Integer; stdcall;
end;
```

下面来了解一下 IUnknown 接口。在 IUnknown 接口中定义了 3 个函数：QueryInterface、_AddRef 和_Release。

◆ QueryInterface

QueryInterface 是请求指向一个接口指针的函数。如果接口是由 Obj 对象实现的，返回 Obj 参数中的接口并且返回值 S_OK；如果不是，则返回 Microsoft 定义的常量 E_NOINTERFACE。这些值在 Delphi 中有定义。

◆ _AddRef 和_Release

这两个函数分别用于增加和减少引用计数器的值。当客户须要使用接口时调用_AddRef 函数；不再使用接口时调用_Release。不过很幸运的一点是，Delphi 提供了实现 IUnknown 接口的类，从而使我们在编程时不太须要考虑增加和减少引用计数器的问题(这在 Visual C ++ 中都没有实现，而须要自己编写引用记数的代码)。在下面的章节中我们将看到这一点。

3. GUID

从前面的声明中用户可能注意到了第 2 行的一长串数字，这就是 IUnknown 接口的全局统一标识符(Globally Unique Identifier，缩写为 GUID)。所有的 COM 接口需要一个惟一的 GUID 才能运行。

GUID 是一个 16 字节的数字。创建 GUID 的算法十分复杂，如果用户的计算机中安装了网卡的话，算法会根据网卡的 MAC 地址作为生成种子之一，从而生成一个惟一的、不会在其他机器上出现的 GUID。即使没有，由于 GUID 的位数很大，也基本上不可能产生重复的 GUID。如果 GUID 用在接口中的话，也称作 IID。

在 Delphi 中，GUID 数据通过 TGUID 记录来表示，该记录在 System 单元中定义如下。

```
PGUID = ^TGUID;
    TGUID = packed record
    D1 : LongWord;
    D2 : Word;
    D3 : Word;
    D4 : array[0..7] of Byte;
end;
```

在 Dephi 中如果要生成一个 GUID，只须要把光标定位在要插入 GUID 的地方，并按下 Ctrl + Shift + G 键就可以了。在 Delphi 中通过 COM Object Wizard 生成一个 COM 对象时，Delphi 会自动生成 GUID。

提示与技巧：

GUID 命名规则

代表 COM 对象的 GUID 称为 CLSID(Class ID)。除了 CLSID 以外，由于一个 COM 对象还拥有接口，因此在 COM 控件模型中也适用 GUID 来表示接口，称为 IID(Interface ID)。除此以外，在 COM 控件模型中还有其他的 GUID，介绍如下。

CLSID(Class ID)：标示一个 COM 对象。

IID(Interface ID)：标示一个 COM 对象的接口，对于 COM 对象中的多个接口，每一个接口都有一个 IID。

APPID(Application Identifier)：应用程序 ID。

CATID(Category Identifier)：COM 控件实现的控件类型。

LIBID(Library Identifier)：COM 对象实现的 Type Library 代表的 ID。

虽然说只要实现了 IUnknown 接口便可以称为是 COM 对象，但是我们之所以要编写 COM 对象，是须要它提供特定的服务以便让其他的应用程序可以调用，以使用该控件提供的服务。因此一个 COM 对象通常会提供其他的客户化接口以便提供特定的服务，并且在控件中实现这些接口所定义的方法、属性以及事件等，如图 1-1 所示。

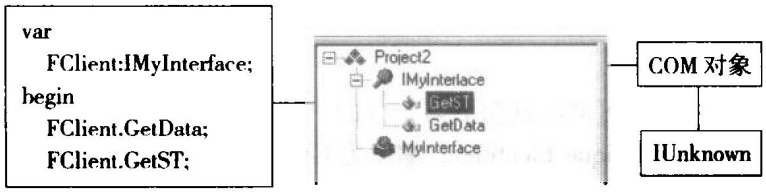


图 1-1 COM 通过接口来提供服务

1.2 COM 对象的实现实例

前面讲到过，COM 是实现一个或者几个接口的对象。一个 COM 对象位于 EXE 文件或者 DLL 文件中，位于 DLL 中的 COM 对象称为进程内服务器；位于 EXE 中的称为进程外服务器。后面将简要地讨论进程外服务器和进程内服务器，在本章中主要接触进程内服务器。

如果要编写可以与任何语言兼容的 COM 对象，则必须从 TComObject 中派生类。TComObject的声明在 comobj. pas 中。该结构的定义很复杂，但是绝大多数我们不须要处理，因为 Delphi 已经为我们提供了实现，我们所要做的只是从 TComObject 类中派生 COM 对象。

1. 类工厂

COM 对象不是由应用程序直接建立实例的。相反，COM 使用类工厂来创建对象。类工厂是一个用于创建其他对象的对象。每一个 COM 对象都有一个相关的类工厂对象。在通过 COM Object Wizard 建立 COM 对象时，Delphi 自动生成代码在 COM 服务器初始化时来创建类工厂并通过类工厂创建 COM 对象，在 Delphi 中注册 TCOMObject 类的类工厂类是 TComObjectFactory 类。该类在 comobj. pas 中有如下定义。

```
TComObjectFactory = class(TObject, IUnknown, IClassFactory, IClassFactory2)
```

TComObjectFactory 类从 TObject 中继承并实现 IUnknown、IClassFactory 和 IClassFactory2接口。

2. COM 对象建立实例

下面我们首先来建立一个简单的进程内的 COM 服务器——这个服务器只包含一个 TMySort 类的 COM 控件，该类从 TComObject 类中继承并且实现预先定义的 IMySort 接口。然后将程序编译并注册，最后再建立一个客户程序访问服务器中的 COM 对象。

首先打开 Delphi，选择 File|New 菜单项，在 New Items(在 Delphi 中的正式称呼为 Object Repository，本文称之为"窗口标题")中选择 ActiveX 选项卡，再选择其中的 ActiveX Library，然后单击 OK 按钮。这样就建立了一个进程内服务器的工程文件框架。将工程文件保存为 SortServ. dpr，文件代码如下。

```
library SortServ;
uses
    ComServ;

exports
    DllGetClassObject,
    DllCanUnloadNow,
    DllRegisterServer,
    DllUnregisterServer;

{$R *.RES}

begin
end.
```

在上面代码中生成了4个输出函数，它们的默认实现已经由 Delphi 中的 comserv. pas 实现了。在这里介绍一下它们的用途。

- DllRegisterServer：在 COM 对象被注册时调用，这个调用可能来自 Delphi 中的 Register ActiveX Server 菜单项，也可能来自 Windows 命令行应用程序 Regsvr32. Exe。无论通过何种方式调用，DllRegisterServer 都将通过修改 Windows 注册表来注册 COM 对象。
- DllUnregisterServer：DllRegisterServer 过程的逆过程，它将 DllRegisterServer 过程在注册表中建立的项删除。
- DllGetClassObject：DllGetClassObject 负责提供给 COM 一个类工厂，该类工厂用于创建一个 COM 对象。每个 COM 服务器将实现它输出的每个 COM 对象的一个类工厂。
- DllCanUnloadNow：COM 负责调用 DllCanUnloadNow，来看是否可以从内存中卸载 COM 服务器。如果在此服务器中任何应用程序都有针对每个 COM 对象的引用，DllCanUnloadNow 返回 S_FALSE；如果此服务器中对于任何 COM 对象都没有打

开的引用，那么 DllCanUnloadNow 返回 S_TRUE，并从内存中移走 COM 服务器。

1.2.1 建立 COM 服务器

下面建立 COM 服务器对象，选择 File | New 菜单项，在 New Items 对话框中选择 ActiveX选项卡，选取其中的 COM Object 项后单击 OK 按钮启动 COM Object Wizard，如图 1-2 所示。

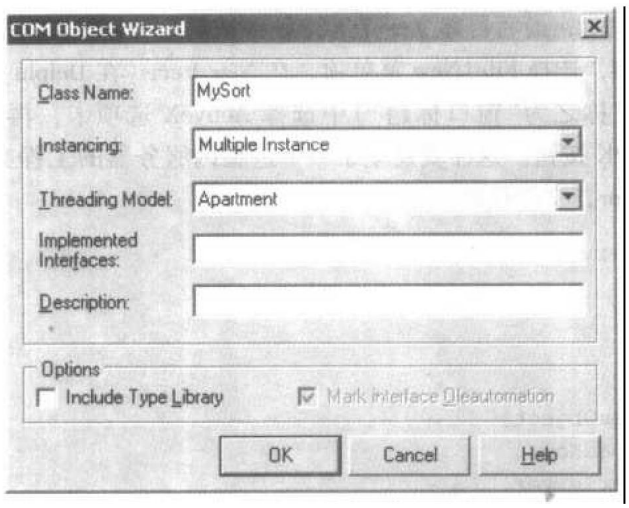


图 1-2 COM Object Wizard

其中，Class Name 用于指定新建立的类的名称，在这里我们填入 MySort；Instancing 指定 COM 对象的实例类型，它的选择以及定义如下。

- Internal：只能建立内部的对象实例。外部程序不能够直接建立对象的实例。
- Single Instance：每一个应用程序只能够建立一个 COM 对象。如果要建立多个 COM 对象实例，必须执行多个应用程序。
- Multiple Instance：该定义可以让多个应用程序同时与 COM 对象建立链接。默认的实例类型是 Multiple Instance。

Threading Model 用于指定 COM 对象使用的线程支持模式，这个选择对于控件的性能有比较重要的影响。线程支持只支持进程内服务器，不支持进程外服务器。进程内服务器可以使用几个线程模式中的一个。进程内服务器的线程模式保存在 Windows 注册表中。可支持的线程模式如下。

- Single(单线程)：单线程模式实际上并没有线程支持。在同一时间只能有一个客户线程访问服务器。在该模式下不存在数据保护的问题。
- Apartment(单元线程)：单元线程(有时称为公寓线程)，这种线程模式允许服务器同时拥有多个相同的 Apartment 在不同的服务器线程中执行，这样可以同时服务多个客户端。但是每一个 Apartment 中只有一个单一的线程可以执行这个 Apartment 中的 COM 对象。在该模式下，局部变量是安全的，但是由于服务器中

可能有多个线程同时执行，所以每一个 COM 对象必须保护全局变量的存取。

- Free(自由线程)：该模式允许客户端应用同时存取 COM 服务器中的任何 COM 对象。在这种模式下，COM 对象必须对局部数据和全局数据都做出保护。
- Both(同时支持单元以及自由线程)：同时支持 Apartment 线程模式以及 Free 线程模式。

现在的应用程序支持的几乎都是 Apartment 模式。在这里我们也选择 Apartment 模式。

下一步将 COM Object Wizard 中 Options 选项组中的 Include Type Library 和 Mark interface Oleautomation 复选框中的选中标记都去掉后单击 OK 按钮。这样 Delphi 就建立了一个新的 Unit，代码如下。

```
unit Unit1;

interface

uses
    Windows, ActiveX, Classes, ComObj;
type
    TMySort = class(TComObject)
    protected
    end;

const
    Class_MySort: TGUID = '{DC29E6B7-F51D-4BB7-A5DC-02F291266C9E}';

implementation

uses ComServ;

initialization
    //初始化时通过类工厂建立 COM 对象实例
    TComObjectFactory.Create(ComServer, TMySort, Class_MySort,
      'MySort', '', ciMultiInstance, tmApartment);
end.
```

将文件以 Unit1.pas 保存。我们可以看到其中包括一个新的类 TMySort，这是通过向导产生的。Class_MySort 是本 COM 服务器的 GUID。在单元的初始化部分是一个建立类工厂 TComObjectFactory 类的 Create 方法。TObjectFactory 的 Create 方法定义如下。

```
constructor Create(ComServer: TComServerObject; ComClass: TComClass;
    const ClassID: TGUID; const ClassName, Description: string;
    Instancing: TClassInstancing; ThreadingModel: TThreadingModel = tmSin-
gle);
```

其中，参数 ComServer 定义了 COM 服务器对象，在程序中一般都要设定为ComServer；参数 ComClass 用于指定要建立的类，在上面的程序中我们指定为 TMySort；参数 ClassID 用于指定类的 GUID，在程序中我们指定为类 TMySort 的 GUID Class_MySort；参数 ClassName 用于指定类的名称；参数 Description 用于指定对类的描述信息；参数 Instancing 用于指定实例类型，在程序中我们指定为 ciMultiInstance；参数 ThreadingModel 指定线程模型，

在定义中默认是 tmSingle(单线程)，在程序中我们指定为 tmApartment。

下面的程序清单 1-1 到 1-3 是 SortServ 的全部源程序。

程序清单 1-1：SortServ. dpr

```
library SortServ;

uses
    ComServ,
    Unit1 in 'Unit1.pas',
    SortInterface in 'SortInterface.pas',
    SortServ_TLB in 'SortServ_TLB.pas';

exports
    DllGetClassObject,
    DllCanUnloadNow,
    DllRegisterServer,
    DllUnregisterServer;

{$R *.TLB}

{$R *.RES}

begin
end.
```

上面程序段中的引用文件 SortServ_TLB. pas 是 Delphi 自动建立的，其中包含 COM 服务器 SortServ 的相关信息。

程序清单 1-2：Unit1. Pas

```
unit Unit1;

interface

uses
  Windows, ActiveX, Classes, ComObj,SortInterface;

type
  TMySort = class(TComObject,IMySort)
  protected
   FItems:array[0..19]of integer;
   FPoint:Integer;

  Public
   constructor Create;
   procedure SetValue(iVal:Integer);
   function GetSize:Integer;
   function GetVal(iInd:Integer):Integer;
   procedure BeginSort;
```

```
    procedure ClearStack;
  end;

implementation

uses ComServ;

constructor TMySort.Create;
   begin
  //建立类时将 FPoint 设置为 0
  FPoint:=0;
end;

procedure TMySort.ClearStack;
begin
  FPoint:=0;
end;

procedure TMySort.SetValue(iVal:Integer);
begin
  //将新数值添加到数组中
  if FPoint<20 then begin
   FItems[FPoint]:=iVal;
   FPoint:=FPoint+1;
  end;
end;

function TMySort.GetSize:Integer;
begin
  //返回数组中数值的个数
  Result:=FPoint;
end;

function TMySort.GetVal(iInd:Integer):Integer;
begin
  //返回数组中各个数值
  if ((iInd>=0) and (iInd<=FPoint)) then
   Result:=FItems[iInd]
  else
   Result:=0;
end;

procedure TMySort.BeginSort;
var
  i,j,Temp:Integer;
begin
  //利用简单的冒泡法对数组进行排序
  for i:=0 to FPoint do
   for j:=FPoint downto i+1 do begin
```

```
        if FItems[j] < FItems[j - 1] then begin
          Temp: = FItems[j];
          FItems[j]: = FItems[j - 1];
          FItems[j - 1]: = Temp;
        end;
      end;
end;

initialization
  TComObjectFactory.Create(ComServer, TMySort, Class_IMySort,
    'MySort', '', ciMultiInstance, tmApartment);
end.
```

程序清单 1-3：SortInterface. pas

```
unit SortInterface;

interface

type
  IMySort = interface
    ['{BB99371A - A959 - 40EB - ACA2 - 5734F3F5B471}']
    procedure SetValue(iVal:Integer);
    function GetSize:Integer;
    function GetVal(iInd:Integer):Integer;
    procedure BeginSort;
    procedure ClearStack;
end;

const
  Class_IMySort:TGUID = '{BB99371A - A959 - 40EB - ACA2 - 5734F3F5B471}';
implementation

end.
```

将 IMySort 接口的定义和 GUID 放到一个单独的、名称为 SortInterface. pas 的文件中使它们更容易在客户代码中引用。

> **提示与技巧：**
>
> SortInterface. pas 中的 GUID 是作者生成的，不要直接抄到你的代码中，而要通过快捷键 Ctrl + Shif + G 生成一个新的。也不要直接从其他代码中拷贝 GUID 而要自己生成，这是一个好习惯。

下面来简单讨论一下代码，主要是 Unit1. pas 中 TMySort 实现 IMySort 的部分。在 TMySort 中包含以下两个 protected 定义。

```
FItems:array[0..19]of integer;
FPoint:Integer;
```