# CnWizards框架详细设计说明书

作者：周劲羽(zjy@cnpack.org)

部门：CnPack开发组 设计组 管理员

类别：包设计

版本：V0.8.0.3

创建：2002.10.09

修改：2007.05.01

## 一 引 言

1. **编写目的**

CnWizards是CnPack开发包的IDE专家包部分，本文档介绍了专家包框架的详细设计内容，描述了专家包框架的设计、实现及使用方法，以及开发专家时应注意的东西，CnPack专家组开发成员必读。

1. **背景**

这份文档也是属于先编码后补文档的那种：）

编写时框架代码已基本完成，同时已开发出两个简单的专家。虽然如此，我仍然郑重地将文档补上，并借此对整个专家包设计做一个总结。

1. **定义**

IOTAWizard接口，是Open Tools API定义的IDE专家必须实现的接口。

Wizard或Expert，专家。根据官方定义，专家必须是实现了IOTAWizard接口并在IDE中注册的对象。而在本文档中专家可能不是这个含义，而是指CnWizards中独立的工具，有的工具虽然实现了专家接口，但实际上并没有在IDE中注册，而通过专家管理器来管理和调用。

RepositoryWizard，资源模板向导。这种向导表现为在IDE中选择“New…”时出现的“New Items”对话框中的一个模板项，通过它可以快速创建单元、窗体和工程的框架。

KeyboardBind，键盘绑定。Delphi4/BCB4以后的IDE中通过键盘绑定来管理快捷键，专家包也采用了这种方式来向IDE中注册快捷键。

1. **参考资料**

1）编写本文件用到的参考资料：

《CnPack开发草案》

《CnWizards专家包概要设计说明书》

2）阅读本文件时需要对照查阅的参考资料：

[CnPack 项目代码包](http://download.cnpack.org/CnWizards_Src.zip)（包含有本文档提到的其它文档、源代码和专家库）

GExperts1.12 源代码。

## 二 框架结构

1. **框架功能描述**

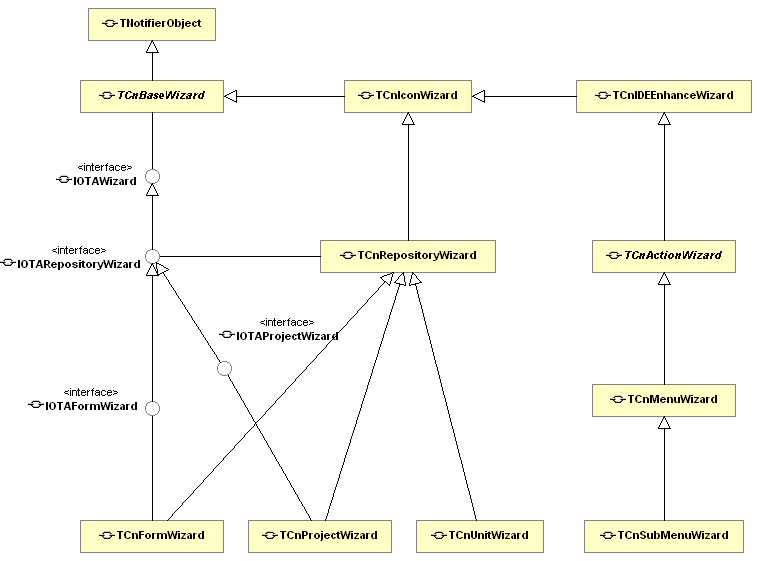
CnWizards框架指不涉及具体专家功能，但为所有专家服务，实现公共功能的部分。

框架要完成在IDE主菜单中插入CnPack菜单、配置和管理快捷键、专家子菜单、提供专家设置界面、实现关于窗口，以及定义专家基础类等功能。同时，框架还要实现专家管理器，负责专家列表的维护、注册、释放等操作。

1. **框架类层次结构**

在CnWizards中，专家分为两大体系：基于菜单、快捷键或在后台运行的专家，以及用来创建新单元、窗体、工程的资源模板向导。

如图一所示，整个专家基础类由以下内容组成：

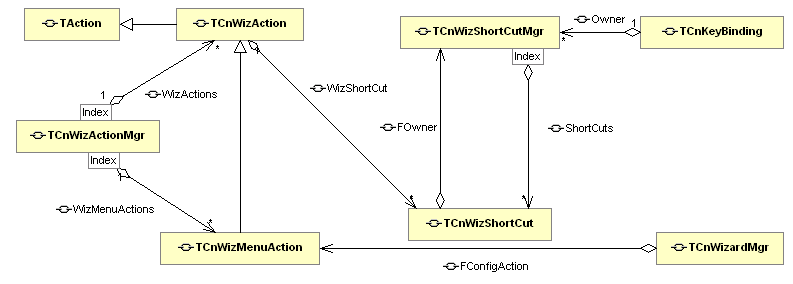


图一 专家包基础类派生关系

* TCnBaseWizard专家抽象基础类，该类实现IOTAWizard接口，并实现一些公共的基础方法。
* TCnIconWizard，带图标的抽象基类，允许专家拥有一个专用图标。
* TCnIDEEnhanceWizard，扩展IDE功能的抽象基类。
* TCnActionWizard带快捷键的抽象类，实现IDE中的Action注册、快捷键绑定等功能，从该类派生出的子类可以用快捷键调用，并可加入到工具栏中。
* TCnMenuWizard带菜单的基础类，通常IDE扩展专家从该类派生，可以通过主菜单来调用。
* TCnRepositoryWizard模板向导基础类，实现IOTARepositoryWizrd接口，其子类放在IDE的New模板仓库中，通常并不直接用它来派生子类。
* TCnUnitWizard单元模板向导，从该类派生出的向导用于创建一个单元。
* TCnFormWizard窗体模板向导，从该类派生出的向导用于创建一个窗体和单元。
* TCnProjectWizard工程模板向导，从该类派生出的向导用于创建一个工程。

1. **框架模块及功能定义**

除了专家基础类以外，框架还要实现其它一些辅助功能，如图二所示：



图二 专家包框架管理类

* 快捷键注册和管理部分，由TCnWizShortCut、TCnWizShortCutMgr、TCnKeyBinding组成，负责向IDE中注册快捷键绑定、维护快捷键列表。TCnWizShortCut为简单的快捷键、响应事件封装类，TCnWizShortCutMgr为快捷键列表管理器，维护一个TCnWizShortCut对象列表。TCnKeyBinding为实现了IOTAKeyboardBinding接口的类，TCnWizShortCutMgr通过使用它来向IDE中注册快捷键。
* Action、菜单注册和管理部分，由TCnWizAction、TCnWizMenuAction、TCnWizActionMgr组成，负责创建和维护注册到IDE中的Action、菜单项列表。TCnWizAction 为TAction子类，通过创建TCnWizShortCut对象实现在IDE中注册快捷键的功能。TCnWizMenuAction在TCnWizAction的基础上增加了菜单项，对应于专家基础类的TCnMenuWizard。TCnWizActionMgr管理器用于维护Action列表。
* TCnWizardMgr专家管理器，用于维护专家列表，并完成一些公共的功能。

## 三 TCnBaseWizard专家抽象基类设计说明

1. **模块描述**

TCnBaseWizard是所有CnWizards专家类的抽象基类，同时定义了一些专家公共属性，该类的实现在CnWizClasses.pas单元中。

1. **功能**

作为专家类的抽象基类，该类定义了与CnPack专家包特定条件相关的一些公共属性，并提供了IOTAWizard接口的统一实现。

1. **继承关系**

类派生关系：

TCnBaseWizard <- TNotifierObject <- TInterfacedObject <- TObject

实现的接口：

IOTAWizard <- IOTANotifier <- IUnknown

注：该类实现了IOTAWizard接口，但这并不表示它的子类都将作为Add In型专家注册到IDE中。事实上，在最初的设计中，该类是直接从TObject派生而来。CnWizards中的所有专家都由专家管理器统一维护，并由管理器创建实例、分配快捷键、创建菜单等来进行调用。之所以要实现IOTAWizard，是因为它的子类TCnRepositoryWizard（见后）只有注册到IDE中才能正常使用，为了统一管理而设计。

1. **类设计**

TCnBaseWizard = class(TNotifierObject, IOTAWizard)   
  private   
    FActive: Boolean;   
    FWizardIndex: Integer;   
  protected   
    procedure SetActive(Value: Boolean); virtual;   
    Active 属性写方法，子类重载该方法处理 Active 属性变更事件   
    function GetHasConfig: Boolean; virtual;   
    HasConfig 属性读方法，子类重载该方法返回是否存在可配置内容   
    function GetIcon: TIcon; virtual; abstract;   
 Icon 属性读方法，子类重载该方法返回返回专家图标，用户专家通常不需要自己处理

    // IOTAWizard methods   
    function GetIDString: string;   
    function GetName: string;   
  public   
    constructor Create; virtual;   
    类构造器   
    destructor Destroy; override;   
    类析构器   
    function WizardName: string;   
    取专家名称，可以是支持本地化的字符串   
  
    // IOTAWizard methods   
    function GetState: TWizardState; virtual;   
    返回专家状态，IOTAWizard 方法，子类重载该方法返回专家状态   
    procedure Execute; virtual; abstract;   
    专家执行体方法，IOTAWizard 抽象方法，子类必须实现。当用户执行一个专家时，将调用该方法。   
  
    procedure GetWizardInfo(var Name, Author, Email, Comment: string); virtual;   
      abstract;   
    取专家信息，用于提供专家的说明和版权信息。抽象方法，子类必须实现。   
 var AName: string - 专家名称，可以是支持本地化的字符串   
 var Author: string - 专家作者，如果有多个作者，用分号分隔   
 var Email: string - 专家作者邮箱，如果有多个作者，用分号分隔   
 var Comment: string - 专家说明，可以是支持本地化带换行符的字符串   
    procedure Config; virtual;   
    专家配置方法，由专家管理器在专家配置界面中调用，当 HasConfig 为真时有效   
    procedure LoadSettings(Ini: TCustomIniFile); virtual;   
    装载专家设置方法，子类重载此方法从 INI 对象中读取专家参数   
    procedure SaveSettings(Ini: TCustomIniFile); virtual;   
    保存专家设置方法，子类重载些方法将专家参数保存到 INI 对象中   
  
    function GetIDStr: string;   
    返回专家唯一标识符，供管理器使用   
    function CreateIniFile: TCustomIniFile;   
    返回一个用于存取专家设置参数的 INI 对象，用户使用后须自己释放   
  
    property Active: Boolean read FActive write SetActive;   
    活跃属性，表明专家当前是否可用   
    property HasConfig: Boolean read GetHasConfig;   
    表示专家是否存在配置界面的属性   
    property WizardIndex: Integer read FWizardIndex write FWizardIndex;   
    专家注册后由 IDE 返回的索引号，在释放专家时使用，请不要修改该值   
    property Icon: TIcon read GetIcon;   
    专家图标属性   
  end;

1. **接口**

所以专家子类都由专家管理器CnWizardsMgr来创建。

专家基类单元维护一个专家类引用列表，管理器通过该列表来创建和注册所有的专家，专家框架提供了以下几个全局函数供专家和管理器使用：

procedure RegisterCnWizard(const AClass: TCnWizardClass);   
注册一个CnBaseWizard专家类引用，每个专家实现单元应在该单元的 initialization   
节调用该过程注册专家类   
function GetCnWizardClass(const ClassName: string): TCnWizardClass;   
根据专家类名取指定的专家类引用   
function GetCnWizardClassCount: Integer;   
返回已注册的专家类总数   
function GetCnWizardClassByIndex(const Index: Integer): TCnWizardClass;   
根据索引号取指定的专家类引用

1. **存储分配**

该类中定义了Icon属性，但并未提供实现，其两大子类分别用不同的方式来实现。

1. **注释设计**

注释规范见开发包注释规范（后面内容同此）。

1. **测试计划**

由于不同编译器版本的差异，该基类的定义要求在所有计划支持的编译器上通过，目前有：Delphi5/6/7、C++Builder5/6以及BDS 2005/2006、Delphi 2007。

详见《CnWizards专家包框架测试计划》（后面内容同此）。

1. **尚未解决的问题**

无。

## 四 TCnActionWizard设计说明

1. **模块描述**

TCnActionWizard是所有支持Action和快捷键的专家抽象基类，它主要派生出TCnMenuWizard，该类的实现在CnWizClasses.pas单元中。

1. **功能**

在TCnBaseWizard的基础上，它增加了Action属性及ImageIndex等相关属性，并定义了一些相关的虚拟方法。通常，只有极少数不需要通过菜单调用的专家才从它派生。

1. **继承关系**

类派生关系：

TCnActionWizard <- TCnBaseWizard <- TNotifierObject <- TInterfacedObject <- TObject

实现的接口同TCnBaseWizard。

1. **类设计**

  TCnActionWizard = class(TCnBaseWizard)   
  private   
    FAction: TCnWizAction;   
    procedure Click(Sender: TObject);   
    function GetImageIndex: Integer;   
  protected   
    function GetIcon: TIcon; override;   
    procedure OnActionUpdate(Sender: TObject); virtual;   
    function CreateAction: TCnWizAction; virtual;   
    function GetCaption: string; virtual; abstract;   
     返回专家的标题   
    function GetHint: string; virtual;   
     返回专家的Hint提示   
    function GetDefShortCut: TShortCut; virtual;   
     返回专家的默认快捷键，实际使用时专家的快捷键会可能由管理器来设定，这里只需要返回默认的就行了。   
  public   
    constructor Create; override;   
     类构造器   
    destructor Destroy; override;   
     类析构器   
  
    property ImageIndex: Integer read GetImageIndex;   
     专家图标在 IDE 的主 ImageList 中的索引号   
    property Action: TCnWizAction read FAction;   
     专家 Action 属性   
    function HasShortCut: Boolean; virtual;   
     返回专家是否可以用快捷键调用   
  end;

1. **接口**

类定义中用到的TCnActionWizard，是专家框架模块的一部分，为封装了IDE快捷键绑定的Action。同时该类定义的虚拟方法CreateAction允许子类重载以返回扩展的Action对象。

1. **尚未解决的问题**

无。

## 五 TCnMenuWizard设计说明

1. **模块描述**

TCnMenuWizard是所有通过菜单调用的专家基类，绝大多数的专家工具由它派生。该类的实现在CnWizClasses.pas单元中。

1. **功能**

在TCnActionWizard的基础上，它增加了菜单属性，并定义了一些相关的虚拟方法。

1. **继承关系**

类派生关系：

TCnMenuWizard <- TCnActionWizard <- TCnBaseWizard <- TNotifierObject <- TInterfacedObject <- TObject

实现的接口同TCnBaseWizard。

1. **类设计**

  TCnMenuWizard = class(TCnActionWizard)   
  private   
    function GetAction: TCnWizMenuAction;   
    function GetMenu: TMenuItem;   
  protected   
    function CreateAction: TCnWizAction; override;   
  public   
    function HasShortCut: Boolean; override;   
    重载该方法，当菜单包含子菜单时返回为假，以防止为它分配快捷键   
    property Menu: TMenuItem read GetMenu;   
     专家的菜单属性   
    property Action: TCnWizMenuAction read GetAction;   
     专家 Action 属性   
  end;

1. **接口**

该类重新声明了Action属性，返回一个带菜单的TCnWizMenuAction，该类也是专家框架模块的一部分，在TCnWizAction的基础上增加了菜单属性。

1. **尚未解决的问题**

无。

## 六 TCnRepositoryWizard设计说明

1. **模块描述**

TCnRepositoryWizard模板向导抽象基类，是所有资源模板向导的基类。它的实现在CnBaseWizard.pas单元中。TCnRepositoryWizard以及它的子类所实现的内容包括如何在IDE中注册一Repository向导，但这里的设计并未包括生成窗体单元代码的内容。关于如何生成代码请参考设计文档《CnOTACreators设计说明书》。

1. **功能**

该类主要实现了IOTARepositoryWizard接口，并封装了一些公共功能。通常我们也不直接使用它，而从的三个子类来派生专家。

1. **继承关系**

类派生关系：

TCnRepositoryWizard <- TCnBaseWizard <- TNotifierObject <- TInterfacedObject <- TObject

实现的接口：

IOTARepositoryWizard <- IOTAWizard <- IOTANotifier <- IUnknown

1. **类设计**

  TCnRepositoryWizard = class(TCnBaseWizard, IOTARepositoryWizard)   
  private   
    FIcon: TIcon;   
  protected   
    function GetIcon: TIcon; override;   
  public   
    constructor Create; override;   
     类构造器   
    destructor Destroy; override;   
     类析构器   
  
    // IOTARepositoryWizard methods   
    function GetAuthor: string;   
    function GetComment: string;   
    function GetPage: string;   
    {$IFDEF COMPILER6\_UP   
    function GetGlyph: Cardinal;   
    {$ELSE   
    function GetGlyph: HICON;   
    {$ENDIF   
  end;

1. **尚未解决的问题**

无。

## 七 TCnUnitWizard设计说明

1. **模块描述**

TCnUnitWizard单元模板向导基类，是用来创建单元的模板向导的基类。它的实现在CnWizClasses.pas单元中。

1. **功能**

该类用来实现创建纯Unit的向导，在实现上它必须实现 IOTAFormWizard 才能在 New 对话框中出现, BDS2006 则要求 IOTAProjectWizard。

1. **继承关系**

类派生关系：

TCnUnitWizard <- TCnRepositoryWizard <- TCnBaseWizard <- TNotifierObject <- TInterfacedObject <- TObject

实现的接口除同TCnRepositoryWizard外，还实现IOTAProjectWizard（DELPHI10 / BDS2006下）或IOTAFormWizard。

1. **类设计**

TCnUnitWizard = **class**(TCnRepositoryWizard, {$IFDEF DELPHI10}IOTAProjectWizard{$ELSE}IOTAFormWizard{$ENDIF});

1. **尚未解决的问题**

无。

## 八 TCnFormWizard设计说明

1. **模块描述**

TCnFormWizard窗体模板向导基类，是用来创建窗体和单元代码的模板向导的基类。它的实现在CnWizClasses.pas单元中。

1. **功能**

该类实现IOTAFormWizard，该接口为空，无需要实现的方法。

1. **继承关系**

类派生关系：

TCnFormWizard<- TCnRepositoryWizard <- TCnBaseWizard <- TNotifierObject <- TInterfacedObject <- TObject

实现的接口：

IOTAFormWizard <- IOTARepositoryWizard <- IOTAWizard <- IOTANotifier <- IUnknown

1. **类设计**

TCnFormWizard = **class**(TCnRepositoryWizard, IOTAFormWizard);

1. **尚未解决的问题**

无。

## 九 TCnProjectWizard设计说明

1. **模块描述**

TCnProjectWizard工程模板向导基类，是用来创建一个工程框架的模板向导的基类。它的实现在CnWizClasses.pas单元中。

1. **功能**

该类实现IOTAProjectWizard，该接口为空，无需要实现的方法。

1. **继承关系**

类派生关系：

TCnProjectWizard<- TCnRepositoryWizard <- TCnBaseWizard <- TNotifierObject <- TInterfacedObject <- TObject

实现的接口：

IOTAProjectWizard<- IOTARepositoryWizard <- IOTAWizard <- IOTANotifier <- IUnknown

1. **类设计**

TCnProjectWizard = **class**(TCnRepositoryWizard, IOTAProjectWizard);

1. **尚未解决的问题**

无。