# CnOTACreators设计说明书

作者：刘啸(liuxiao@cnpack.org)

部门：CnPack开发组 专家组 设计员（管理员、开发员）

类别：类设计

版本：V0.1.0.1

创建：2004.06.24

修改：2007.11.11

## 引 言

### 编写目的

本文档描述了CnWizards IDE专家包中，利用Open Tools Api中的IOTACreators接口而设计的CnOTACreators体系所实现的主要功能和框架结构。CnOTACreators体系主要用在Repository类型的专家里头。

预期读者为CnPack开发组参与CnWizards开发的成员及对专家开发感兴趣的成员，要求具有一定的VCL底层知识和Open Tools API的基本知识。

### 背景

CnPack开发当前处于进阶阶段，CnPack IDE 专家包已经初步成型，已发布0.6.9内部测试版，在CnWizards中Repository类型的专家开发完成后，补充CnOTACreators的设计文档。

### 定义

CnWizards：CnPack开发包中IDE专家工具包部分，简称专家包。

Repository专家：用来在IDE中产生新文件（包括源码、窗体、工程等类型）的向导式专家或称模板专家。

### 参考资料

#### 编写本文件用到的参考资料：

《CnWizards专家包概要设计说明书》（周劲羽）

《CnWizards专家包框架详细设计说明书》（周劲羽）

#### 阅读本文件时需要对照查阅的参考资料：

OpenTools Api帮助文档。

## CnOTACreators设计预备知识

OpenTools Api中使用了IOTACreators等接口提供Repository的功能以达到在IDE中生成源码、窗体、工程等文件的目的。一般的Repository专家都需要按不同的具体环境和设置以生成不同内容的文件，为了增加灵活性，CnPack IDE 专家包中设计了一套使用TCnTemplateParser类来根据模板生成文件的类体系，称之为CnOTACreators，下面简要加以说明。

### IOTACreators和相关运行机制

IOTACreators中的文件内容创建机制可以这样描述：首先通过BorlandIDEServices来获得一个IOTAModuleServices接口实例，然后调用这个接口的CreateModule函数就可以完成文件内容的创建。而CreateModule函数需要一个IOTACreator类型的接口实例作为参数。所以，一个Repository专家要做的事情大多数就是写一个实现IOTACreator接口的类，在这个类的各个实现方法中定义好返回的内容，然后创建这个类的实例传递给CreateModule函数就能完成实际的创建。

CreateModule函数会调用我们传递的实例的各个接口成员函数，其中最重要的是NewImplSource等几个需要返回IOTAFile接口的函数。IDE利用定义好的IOTAFile接口，就能从中直接获得文件内容。IOTAFile的实现在后文有详述。

IOTACreator接口和其子接口的一系列方法都是被IDE调用的，这些方法里头要求实现这个接口的对象返回待创建文件的具体信息，如是否生成头文件、是否生成窗体文件、是否命名等。具体信息可参考Open Tools Api中有关IOTACreator、IOTAModuleCreator、IOTAProjectCreator、IOTAProjectGroupCreator等内容的相关帮助信息。

### TCnTemplateParser的模板和标签替换机制

TCnTemplateParser类类似于用在网页开发中的TPageProducer类，用于根据模板进行不同的标签替换，从而产生不同的输出内容。TCnTemplateParser的LoadTemplateFile方法可以加载模板文件。模板文件是普通文本文件，当调用TCnTemplateParser的Content属性要求它产生页面时，它会从头到尾扫描模板文件，当遇到<#Name>之类的标签的时候，就会触发OnTag事件供调用者自定义标签替换内容，从而达到灵活设置其内容的目的。

OnTag的事件处理器声明如下：

TCnReplaceTagEvent = procedure (Sender: TObject; const TagString: string;

TagParams: TStrings; var ReplaceText: string) of object;

假设TCnTemplateParser在扫描模板文件的过程中碰到了形如<#Name>的Tag，那么就会触发OnTag事件，参数中的TagString即是#号后的字符串，在这里是“Name”，如果给ReplaceText赋值，则ReplaceText的内容会替代“<#Name>”，成为实际生成的内容。

## CnOTACreators详细说明

CnOTACreators类体系使用了TCnTemplateParser类作为其文件生成的工具，模板文件使用自定义的单元文件和窗体文件等。CnOTACreators体系的作用是用TCnTemplateParser的Tag替换机制，自动实现IOTACreator及其子接口的多个复杂函数，而将真正的创建设置集中控制在类的几个替换处理函数中，以简化专家子类的开发。

CnOTACreators类体系包括以下几个类：

* TCnBaseCreator：它是所有Creator的基类，实现了IOTACreator接口。TCnBaseCreator是一个抽象的文件创建接口实现类，不能被直接使用，具体的不同文件类型的创建需要落实到其子类。它内部使用了TCnTemplateParser来实现基本的Tag替换机制。
* TCnTemplateModuleCreator：它是创建单个单元或包括窗体的单元的类，实现了IOTAModuleCreator接口，能够根据模板文件类型设置其生成文件的类型和内容。
* TCnTemplateProjectCreator：它是创建工程的类，继承于TCnBaseCreator，实现了IOTAProjectCreator接口，能够创建一工程。
* TCnTemplateDataModuleCreator：它是创建数据模块的类，继承于TCnTemplateModuleCreator，简单重载了IOTAModuleCreator接口中的GetAncestorName返回“DataModule”标识，和TCnTemplateModuleCreator大同小异。
* TCnOTAFile：它是一个简单的IOTAFile接口的实现类，能够接受和表示被创建的文件内容。Creators需要用到它来向IDE提供IOTAFile接口。

另外还应该有TCnTemplateProjectGroupCreator，实现IOTAProjectGroupCreator，用来创建工程组。因应用较少，所以暂未设计。

利用CnOTACreators类体系进行Repository专家开发也不太麻烦，一般来说，只需根据需求，选择基类继承一子类，然后重载其模板文件获取和标签替换处理的函数，写上自己的返回内容。执行时用类似于以下的方法便可完成新建文件（假设TMyModuleCreator是已经写好的TCnBaseCreator的子类）：

MyModuleCreator := TMyModuleCreator.Create;

(BorlandIDEServices **as** IOTAModuleServices).CreateModule(MyModuleCreator);

## CnOTACreators类层次设计

### CnOTACreators类层次图

CnOTACreators的类设计图如下图所示。

new diagram

该类图中，矩形代表单个类，空心单箭头表示继承关系，斜体字母代表抽象类或抽象方法，#表示protected域成员，+表示public域成员。

TCnBaseCreator中定义了抽象虚函数GetTemplateFile，子类必须实现此函数以返回合适的模板文件名。

OnReplaceTagsSource是TCnTemplateParser的OnTag事件处理函数，在TCnBaseCreator中是空内容，在TCnTemplateModuleCreator等子类中加入了基本的模块名等的替换处理，并调用虚函数DoReplaceTagsSource进行自定义标签替换。具体实现文件创建功能的子类主要是重载GetTemplateFile和DoReplaceTagsSource，如果需要进行其他控制，还可以重载IOTACreators及其子接口的一些方法。

### IOTAFile接口的实现

在IDE最终生成文件内容的时候，需要我们提供一个实现了IOTAFile接口的具体类的实例给它，由它调用此接口以获得文件的具体内容。我们使用了一个通用的实现IOTAFile接口的类来实现此功能，这个类很简单，创建的时候指定其内容字符串，然后在需要的时候，将对象实例返回即可。

  TCnOTAFile = **class**(TInterfacedObject, IOTAFile)   
  *{\* 实现 IOTAFile接口的简单文件类 }*   
  **private**   
    FSource: **string**;   
  **public**   
    **constructor** Create(**const** Source: **string**);   
    **function** GetSource: **string**;   
    **function** GetAge: TDateTime; **virtual**;   
  **end**;

这个机制在TCnBaseCreator.GetIOTAFileByTemplate中应用，GetIOTAFileByTemplate由IDE间接调用，内部实现了获取模板文件、进行TCnTemplateParser的替换、返回IOTAFile接口等功能，可以说是文件产生部分的核心内容。主要起作用的代码如下：

Producer.HTMLFile := TemplateFile;

Producer.OnHTMLTag := OnReplaceTagsSource;

Result := TCnOTAFile.Create(Producer.Content);

TCnOTAFile.Create(Producer.Content)便是根据Producer（一TCnTemplateParser实例）替换出来的文件内容生成一个TCnOTAFile的实例，该实例实现了IOTAFile接口，因此可以被返回给IDE让它调用以产生文件内容。至于这个对象啥时候被释放就不用担心了，接口引用计数能保证它在被IDE使用完毕后被自动释放。

## 其它设计

### 关键技术说明

关键技术包括TCnTemplateParser的使用和Open Tools Api中的相关内容。

### 性能

Repository专家没有明显的性能要求，一般在1秒内能完成替换，1到5秒内能完成新文件在IDE中的生成。

### 接口

CnOTACreators使用了Open Tools Api中的IOTACreator、IOTAModuleCreator、IOTAProjectCreator、IOTAProjectGroupCreator和IOTAFile等接口，不对外提供接口。

### 存储分配

无。

### 注释设计

参考CnPack发布的《Delphi单元规范格式.pas》。

### 限制条件

受IDE的限制。

### 开发平台

Delphi 5 ＋Windows 2000

### 测试计划

在现有的“DUnit测试用例生成向导”和“INI读写单元生成向导”中测试。

### 尚未解决的问题

TcnTemplateParser目前的局限性主要在于不支持嵌套标签替换，也就是说，标签的替换只能整体进行。不能在替换某个标签的时候在内部进行又一次替换，这让CnOTACreators的应用有了不少局限。如果要解决这个问题，则需要进行进一步的功能扩展和升级。