

# CnPack 多语组件帮助文档

作者: 刘啸(liuxiao@cnpack.org)  
部门: CnPack 开发组 不可视组件开发组 管理员  
类别: 帮助文档  
版本: V0.8.2.0  
创建: 2006.09.28  
修改: 2007.11.11

## 一、 引 言

### 1、 组件包概述

CnPack 多语组件是 CnPack 组件包中的一个重要组成部分, 主要用来用来实现多语言界面。它们虽未组成可安装的独立组件包, 但它们功能相对独立, 在组件板上独占一页, 和其他组件关联不大。本文档提供了 CnPack 多语组件包中的几个组件的详细帮助信息。

一个完整的多语工程所应该具有的多语支持翻译能力应该包括三部分: **多个语言条目的存储能力、运行期切换语言的能力和设计期支持的能力**。为工程提供这三种功能的组件分别是多语存储组件、多语管理器和多语翻译器。

### 2、 术语定义

**多语包/CnMultiLang:** CnPack 组件包的一个相对独立的子部分, 由几个实现多语功能的组件组成, 可单独安装, 也可以随 CnPack 组件包一起安装。

**多语条目/语言条目:** 对应着不同语种的字符串, 供多语包管理检索, 并能被加载并显示到软件界面上去。

**多语存储文件:** 保存某语言条目的文件, 在我们的多语包中, 一个文件保存且只保存一种语言的所有条目。

**翻译能力:** 能对其他对象进行属性字符串轮询并查找替换的能力, 并非指人工智能中的自动将一种语言条目翻译成另外一种语言条目。

**多语化/本地化:** 对应用程序的代码进行处理, 以使它能方便地切换到另外一种语言, 或使它具有这种切换语言的能力。

### 3、 最简单的多语组件的使用步骤

1. 建立一空工程, 拖放一些可视化组件在上面。保存。
2. 拖一 TCnHashLangFileStorage, 添加一语言条目。
3. 拖一 TCnLangMgr, LanguageStorage 设置为刚才的 CnHashLangFileStorage1。
4. 拖一 TCnLangTranslator, 双击打开翻译管理器, 生成语言条目并翻译、保存。
5. 运行, 代码中写 CnLanguageManager.CurrentLanguageIndex := 0; 即可看到翻译后

的界面。其中 0 为第一个语言条目的索引号。

## 二、 TCnLangMgr 组件

### 1、 组件功能概述

TCnLangMgr 是注册到组件板的多语管理器组件，主要用来提供翻译窗体界面等的功能。它能连接到一多语存储组件。设置多语管理器的 `LanguageIndex` 可触发当前语言的变化，多语管理器可将当前语言改变的事件传入多语存储组件，待多语存储组件装载当前语言条目成功后，从多语存储组件中读取当前语言的条目并进行窗体界面等的翻译。同时多语管理器还提供一些属性、方法和事件用来进行其他的翻译控制。

### 2、 所在文件

CnLangMgr.pas, 多语管理器基础类单元。

### 3、 属性说明

**property** LanguageStorage: TCnCustomLangStorage;

多语存储属性，类型为 `TCnCustomLangStorage`，指向一具体的多语存储组件，可读写。

**property** CurrentLanguageIndex: Integer;

当前语言号索引，影响到整个程序的语言设置。语言号含义由存储组件的条目内容决定，可读写。

**property** AutoTranslate: Boolean;

布尔属性，控制是否在当前语言号改变后自动翻译已经存在的窗体和其他内容。可读写。

**property** TranslationMode: TCnTranslationMode;

`TCnTranslationMode = (tmByComponents, tmByStrings);`

翻译模式，控制根据窗体和控件等遍历还是根据翻译字符串内容遍历，前者是递归搜索界面字符串并去多语存储组件中寻找翻译条目，后者是遍历多语组件中的字符串条目并定位到界面进行翻译。默认为前者。可读写。

**property** AutoTransOptions: TCnAutoTransOption;

`TCnAutoTransOption = (atApplication, atForms, atDataModules);`

`TCnAutoTransOptions = set of TCnAutoTransOption;`

自动翻译时的选项，当自动翻译属性 `AutoTranslate` 为 `True` 时，控制是否翻译现存的窗体、数据模块和 `Application` 实例，默认三者都参与翻译。可读写。

**property** TranslateListItem: Boolean;

布尔属性，控制翻译过程中是否翻译 `ListView` 中的 `ListItem`，默认为 `True`。可读写。

**property** TranslateTreeNode: Boolean;

布尔属性，控制翻译过程中是否翻译 **TreeView** 中的 **TreeNode**，默认为 **True**。可读写。

**property** UseDefaultFont: Boolean;

布尔属性，控制是否翻译完窗体后使用 **DefaultFont** 属性来设置窗体字体。默认为 **True**。可读写。

**property** TranslateOtherFont: Boolean;

布尔属性，控制翻译过程中是否将除 **TControl.Font** 外的其他的 **Font** 属性翻译成字符串。默认为 **False**。可读写。

**property** IgnoreAction: Boolean;

布尔属性，控制翻译过程中是否翻译 **Action** 属性不为空的控件的 **Caption** 和 **Hint** 属性。因为连接了 **Action** 的组件自身的 **Caption** 和 **Hint** 等属性，通常都由对应的 **Action** 控制，不需要单独设置，所以此属性默认为 **True**。可读写。

## 4、 方法说明

**function** Translate(Src: string): string;

功能：根据当前语言和字符串标识从多语存储组件中获得翻译的字符串。

返回值：翻译的字符串值，无多语组件或当前语言条目中无此标识则返回空。

参数：

6. Src: string; 待翻译的标识字符串，如 “TForm1.Caption”

**function** TranslateString(Src: string): string;

功能：根据当前语言和字符串标识从多语存储组件中获得翻译的字符串。

返回值：翻译的字符串值，无多语组件或当前语言条目中无此标识则返回空。

参数：

1. Src: string; 待翻译的标识字符串，如 “TForm1.Caption”

**function** TranslateStrFmt(Src: string; Args: array of const): string;

功能：根据当前语言和字符串标识从多语存储组件中获得翻译字符串，然后将此字符串作为格式字符串，将后续参数填入后返回。

返回值：翻译并格式化后的字符串值，无多语组件或当前语言条目中无此标识则返回空。

参数：

1. Src: string; 待翻译的标识字符串，如 “SCnErrorFmt”

2. Args: array of const; 作为 **Format** 参数的常量数组。可参考 **Format** 函数。

**procedure** TranslateForm(AForm: TCustomForm);

功能：递归翻译一个 **Form** 及其子对象和属性。

参数：

1. AForm: TCustomForm; 待翻译的窗体。

```
procedure TranslateComponent (AComponent: TComponent;  
    const BaseName: string = '');
```

功能: 翻译一个组件及其子对象和子属性。

参数:

1. AComponent: TComponent; 待翻译的组件。
2. **const** BaseName: **string**; 翻译前导字符串, 默认为空。

```
procedure AddChangeNotifier (Notify: TNotifyEvent);
```

```
TNotifyEvent = procedure (Sender: TObject) of object;
```

功能: 增加一个语言改变后的事件通知。语言改变后, 该事件会被调用。

参数:

1. Notify: TNotifyEvent; 待添加的事件处理函数, 也就是方法名。

```
procedure RemoveChangeNotifier (Notify: TNotifyEvent);
```

```
TNotifyEvent = procedure (Sender: TObject) of object;
```

功能: 删除已经增加过的语言改变时后的事件通知。删除该事件后, 语言改变后此事件不再被调用。

参数:

1. Notify: TNotifyEvent; 待删除的事件处理函数, 也就是方法名。

## 5、 事件说明

```
property OnStorageChanged: TNotifyEvent;
```

```
TNotifyEvent = procedure (Sender: TObject) of object;
```

存储组件属性 LanguageStorage 改变时触发。

参数:

2. Sender: TObject; 组件管理器实例。

```
property OnLanguageChanged: TNotifyEvent;
```

```
TNotifyEvent = procedure (Sender: TObject) of object;
```

当前语言索引号改变后触发。

参数:

1. Sender: TObject; 组件管理器实例。

```
property OnTranslateObject: TCnTranslateObjectEvent;
```

```
TCnTranslateObjectEvent = procedure (AObject: TObject; var Translate:  
Boolean) of object;
```

开始翻译一对象时触发, 用户可控制是否翻译此对象。注意此事件仅当翻译模式是以控件为根据 (tmByComponents) 时触发。

参数:

1. AObject: TObject; 待翻译的对象。
2. **var** Translate: Boolean; 给此变量赋值可控制是否翻译此对象。

```
property OnTranslateObjectProperty: TCnTranslateObjectPropertyEvent;  
TCnTranslateObjectPropertyEvent = procedure (AObject: TObject; const  
PropName: string; var Translate: Boolean) of object;
```

开始翻译一对象的某个属性时触发，用户可控制是否翻译此对象的此属性。

参数：

1. AObject: TObject; 待翻译的对象。
2. **const** PropName: **string**; 待翻译的对象的属性名。
3. **var** Translate: Boolean; 给此变量赋值可控制是否翻译此对象。

## 6、 公用函数说明

```
function CnLanguageManager: TCnCustomLangManager;
```

功能：全局函数，用于返回多语言管理器的实例。

返回值：第一个创建的多语管理器实例，如无实例，则返回 **nil**。

```
procedure CreateLanguageManager(AOwner: TComponent = nil);
```

功能：创建多语言管理器，用于非可视化或手工创建多语言管理器的场合。

参数：

1. AOwner: TComponent; 待创建的多语管理器实例的 Owner。

```
procedure TranslateStr(var SrcStr: string; const IDStr: string);
```

功能：翻译某个字符串，如无翻译管理器或不存在翻译后的条目，则 **SrcStr** 保持不变。

参数：

1. **var** SrcStr: **string**; 待翻译的字符串，如果翻译成功，则被赋值为翻译后的值。
2. **const** IDStr: **string**; 字符串标识。

```
procedure TranslateStrArray(var StrArray: array of string; const IDStr:  
string);
```

功能：循环翻译某个字符串数组的内容，标识字符串为 **IDStr** 加数组下标的形式。

参数：

1. **var** StrArray: **array of string**; 字符串数组。
2. **const** IDStr: **string**; 字符串标识。

## 7、 属性编辑器说明

```
property CurrentLanguageIndex: Integer;
```

此属性的属性编辑器能下拉当前存储组件的语言条目 **ID** 和名称等供直观选择，如无多语存储组件，则无供选择条目。

# 三、 TCnLangTranslator 组件

## 1、 组件功能概述

TCnLangTranslator 是注册到组件板上的多语翻译器组件，本身无具体功能，仅仅用于设计期

使用其组件编辑器来进行窗体标识字符串的收集生成，并提供手工翻译界面。运行期的多语功能并不依赖于此组件。

## 2、 所在文件

CnLangTranslator.pas 多语翻译器的实现单元。  
 CnTransEditor.pas 其设计期组件编辑器的实现单元。

## 3、 属性说明

无。

## 4、 方法说明

无。

## 5、 事件说明

无。

## 6、 组件编辑器（翻译管理器）说明

双击设计期窗体上的多语翻译器组件，弹出翻译管理器窗口，如下图：



窗体翻译管理器上部是工具栏。左边的树状结构显示了当前窗体上的多语存储组件及其语言条

目(黑体的多语组件表示它是多语管理器所连接的多语存储组件,黑体的语言条目表示是当前语言)。选中某语言后,该语言内的所有字符串会显示在右边翻译栏目内供编辑。用户也可以根据当前窗体等生成语言条目。

右边的翻译后文本编辑框,选中后单击可进入编辑状态。但对多行字符的支持比较不好。这种情况下用户可能需要保存后手工编辑多语组件的存储内容。

**汇总:** 自动搜索当前工程的所有窗体,生成所有窗体的需要翻译的字符串置入当前语言中。当前语言现有的内容将被清空。

**生成:** 自动搜索当前窗体,生成当前窗体的需要翻译的字符串置入当前语言中。当前语言现有的内容将被清空。

**复制:** 生成的字符串只包括待翻译的条目和原文的字符串值。点击此按钮可将“原文”全部复制到“翻译后文本”中。注意“原文”是不保存到多语存储组件中的。

**更新:** 当当前窗体的语言条目生成完毕保存了,但窗体的组件又发生了变化后,可通过此按钮更新当前语言的条目。已经翻译的条目不会丢失。

**保存:** 将当前翻译栏目中的字符串保存到多语存储组件中。切换左边语言前请注意是否保存当前语言条目。

**加行:** 在翻译栏目中增加一空行,用户可手工输入翻译条目。

**删行:** 在翻译栏目中删除选中的一行。

**删空:** 自动生成的翻译条目可能包括很多字符串值为空的条目,点击此按钮可删除当前语言中的这些空值条目。

**清空:** 删除当前语言的所有翻译字符串条目。

**关闭:** 关闭此窗口,不提示保存。

## 四、 TCnCustomLangStorage 基类

### 1、 基类功能概述

TCnCustomLangStorage 是有多语存储组件的基础类,其自身并未注册到组件板上。它提供了一些基础的属性方法等供多语管理器操纵,包括存储、加载多语条目,改变当前语言等,同时还提供设计期的语言条目编辑功能。

所有多语存储组件的具体实现类都应该继承于此基类。

### 2、 所在文件

CnLangStorage.pas 多语存储基类的实现单元。

CnLangEditors.pas 其设计期部分组件编辑器的实现单元。

### 3、 属性说明

**property** CurrentLanguage: TCnLanguageItem;  
当前语言条目对象,只读。受当前语言索引号的控制。

**property** CurrentLanguageIndex: Integer;  
当前语言索引号,供多语管理器设置,可读写。不推荐在脱离多语管理器的控制下而直接改变

它。

**property** DefaultFont: TFont;

当前语言的默认字体，只读。受当前语言条目中字体部分的控制，如当前语言无默认字体信息，则受操作系统语言的控制。

**property** DefaultLanguageID: Integer;

默认语言的 ID，只读。受操作系统语言的控制。

**property** FontInited: Boolean;

说明当前语言装载后，字体是否已经初始化完毕，只读。

**property** LanguageCount: Integer;

该多语存储组件中的语言条目数，只读。

**property** Languages: TCnLanguageCollection;

该多语存储组件中的所有语言列表，为一 TCollection 子类。

## 4、 方法说明

**procedure** AddLanguage (ALanguageID: LongWord);

功能：增加一空语言条目，该条目的语言 ID 为传入的 ALanguageID 参数。

参数：

1. ALanguageID: LongWord; 待添加的语言条目的语言 ID。

**function** GetString (Name: string; var Value: string): Boolean; **virtual;**  
**abstract;**

功能：抽象方法，根据一字符串标识获得翻译后的字串，子类必须重载以实现翻译。这个方法是整个翻译功能的基础。

返回值：翻译是否成功，成功为 True。

参数：

1. Name: String; 供查找的翻译字符串标识，比如“TForm1.Caption”。
2. var Value: string; 翻译后通过此参数返回翻译后的值。

**procedure** GetNamesList (List: TStrings); **virtual;** **abstract;**

功能：抽象方法，获得当前语言的所有翻译条目名称列表。子类必须重载实现。

参数：

1. List: TStrings; 将当前语言的所有翻译字符串标识加入此 Strings 中，Strings 中原有的内容会被清空。

**procedure** ClearCurrentLanguage; **virtual;** **abstract;**

功能：抽象方法，删除当前语言的所有翻译条目列表。子类必须重载实现。

**function** LoadCurrentLanguage: Boolean; **virtual;** **abstract;**

功能：抽象方法，可以从存储介质中载入当前语言条目，为翻译字符串做准备。子类可视需要重载实现。

返回值：Load 是否成功，成功返回 True。

**procedure** SaveCurrentLanguage; **virtual; abstract;**

功能：抽象方法，可以是保存当前语言条目到存储介质中。子类可视需要重载实现。

**procedure** SetString(Name, Value: string); **virtual; abstract;**

功能：抽象方法，在当前语言中设置一翻译字符串条目。如此条目存在则覆盖，不存在则新增。

参数：

1. Name: string; 待设置的翻译字符串标识，比如“TForm1.Caption”。
2. Value: string; 翻译后的字符串值，比如“窗体标题 1”。

**function** CreateIterator: ICnLangStringIterator; **virtual;**

功能：抽象方法，获得一条目的遍历器接口实例，如果子类不支持按存储条目遍历，则必须返回 nil。

返回值：遍历器接口实例。

## 5、 事件说明

**property** OnLanguageChanged: TLanguageChangeEvent;

TLanguageChangeEvent = **procedure** (Sender: TObject; ALanguageIndex: Integer)  
**of object;**

事件：当前语言号改变后触发。

参数：

1. Sender: TObject; 多语存储组件本身。
2. ALanguageIndex: Integer; 改变后的语言号。

**property** OnLanguageChanging: TLanguageChangingEvent;

TLanguageChangingEvent = **procedure** (Sender: TObject; ALanguageIndex: Integer;  
var AllowChange: Boolean) **of object;**

事件：当前语言号改变前触发，可控制是否允许改变。

1. Sender: TObject; 多语存储组件本身。
2. ALanguageIndex: Integer; 欲改变成的语言号。
3. var AllowChange: Boolean; 是否允许改变，如事件处理函数中将其赋值为 False，则此次改变被禁止，当前语言号保持不变。

## 6、 属性/组件编辑器说明

**property** Languages: TCnLanguageCollection;

双击多语存储组件，可弹出 Languages 属性的 Collection 编辑器供编辑语言条目。

## 五、 TCnCustomLangFileStorage 基类

### 1、 组件功能概述

TCnCustomLangFileStorage 是所有基于文件存储方式的多语存储组件的基础类，其自身并未注册到组件板上。它提供了一些基于文件/目录等的属性方法等供多语管理器操纵，实现了两种多语文件存储模式（文件方式、目录方式）。其实现规则如下：

- 每一语言条目对应一存储文件，里头可存储多个待翻译条目。所有存储文件具有同一扩展名。
- 存储模式为文件模式时，所有语言条目的语言文件都存放于同一目录下，扩展名相同，文件名默认使用该语言的三字母缩写。
- 存储模式为文件模式时，每一语言条目的语言文件可存放于同一目录下的“语言 ID”子目录下，比如英语的存可于 1033 子目录下。不同语言的语言存储文件的文件名相同，但所在目录不同。文件名默认使用可执行文件名。

所有基于文件的、符合上述文件存储规则的多语存储组件的具体实现类都应该继承于此基类。

### 2、 所在文件

CnLangStorage.pas 多语翻译器的实现单元。

CnLangEditors.pas 其设计期部分组件编辑器的实现单元。

### 3、 属性说明

```
property StorageMode: TCnStorageMode;  
TCnStorageMode = (smByFile, smByDirectory);
```

该多语存储组件的文件存储类型，按同一目录下多文件存储还是不同目录下的同一文件名存储。可读写。

```
property LanguagePath: string;
```

所有语言文件存储的统一目录名。可读写。注意这是全路径名，如“C:\Lang”，如果要使用相对路径，请使用“.\Lang”的形式。

```
property FileName: string;
```

多语文件按目录存储时具有的统一文件名。可读写。

```
property AutoDetect: Boolean;
```

LanguagePath 改变时是否自动搜索检测此目录下的语言文件并生成语言条目。可读写。

### 4、 方法说明

```
function GetCurrentLanguageFileName: string; virtual;
```

功能：获得当前语言的语言文件名，包括扩展名。

返回值:

返回当前语言的语言存储文件的完整路径名。

```
class function GetLanguageFileExt: string; virtual;
```

功能: 类方法, 获得所有语言文件的统一扩展名。

返回值: 返回此多语存储组件的语言存储文件的统一扩展名, 子类如不重载则返回默认的 **txt**。

```
function IsLanguageFile(const FileName: string): Boolean; virtual;
```

**abstract**;

功能: 抽象方法, 判断某一文件是否是合法的语言文件。用于自动检测文件时。子类必须实现。

参数:

1. **const** FileName: **string**; 待检测的文件名。

## 5、 事件说明

无。

## 6、 组件编辑器说明

```
property LanguagePath: string;
```

此属性的属性编辑器可弹出一目录浏览对话框供选择目录。

# 六、 TCnHashLangFileStorage 组件

## 1、 组件功能概述

TCnHashLangFileStorage 是注册到组件板上的、基于 TXT 文件和 Hash 方式管理多语条目的多语存储组件, 它内部使用了 **HashMap** 来管理和搜索多语条目, 因此具有较快的速度。

其祖先类的属性方法事件等请参考上文, 这里只列出它自身的属性。

## 2、 所在文件

CnHashLangStorage.pas Hash 多语存储组件的实现单元。

## 3、 属性说明

```
property ListLength: Integer;
```

内存 **HashMap** 的初始列表大小, 可读写。默认值为 **1024**, 一般不需要改动。

```
property IncSize: Integer;
```

重新分配时增加的大小。可读写。默认值为 **2**, 一般不需要改动。

## 4、 方法说明

无。

## 5、 事件说明

无。

# 七、 TCnIniLangFileStorage 组件

## 1、 组件功能概述

TCnIniLangFileStorage 是注册到组件板上的、基于 INI 文件的多语存储组件，它继承自 TCnHashLangFileStorage 组件，内部也同样使用 HashMap 来管理条目，而外部接口使用的是 VCL 中封装的 INI 读写过程，因此在工程稍大时读写效率稍许低下一些。

TCnIniLangFileStorage 的 INI 格式规定如下：以感叹号!开头的内部条目存储于[!Global]节中，普通字符串翻译条目存储于[!Strings]节中，其余每个窗体或每个 DataModule 内的所有字符串条目自成一节，节名为窗体类名或 DataModule 类名。

其祖先类的属性方法事件等请参考上文，其自身无额外的属性方法和事件。

## 2、 所在文件

CnIniLangFileStorage.pas，INI 多语存储组件的实现单元。

## 3、 属性说明

参见 TCnHashLangFileStorage。

## 4、 方法说明

参见 TCnHashLangFileStorage。

## 5、 事件说明

参见 TCnHashLangFileStorage。

# 八、 TCnLanguageCollection/TCnLanguageItem 类

## 1、 功能概述

TCnLanguageCollection/TCnLanguageItem 用于在多语存储组件中管理语言条目，为从 TOwnedCollection 和 TCollectionItem 所继承的类。多语存储组件的 Languages 属性中便包括此两类的实例。前者的属性方法事件等和普通 Collection 类似，因此此处不赘述，只列出 TCnLanguageItem 类的各项信息。

一 TCnLanguageItem 对应着一种语言及其语言翻译条目的描述，包括语言 ID、语言名称、翻译此语言的作者信息等等。

## 2、 所在文件

CnLangCollection.pas 语言条目的实现单元。

## 3、 属性说明

**property** Abbreviation: **string**;

该语言的三字母缩写，在设置 LanguageID 时会自动设置。可读写。

**property** Author: **string**;

该语言的翻译条目的作者名。可读写。

**property** AuthorEmail: **string**;

该语言的翻译条目的作者的电子邮件地址。可读写。

**property** LanguageID: LongWord;

该语言的 ID 号。可读写。设置它会影响其余属性。

**property** LanguageName: **string**;

该语言的名称，在设置 LanguageID 时会自动设置。可读写。

**property** LanguageFileName: **string**;

可用的保存文件名供多语言存储组件使用，以文件方式存储时其结果有效。可读写。在设置 LanguageID 时会被自动设置为三字母缩写。

**property** LanguageDirName: **string**;

可用的保存多语的目录名，以目录方式存储时其结果有效。可读写。在设置 LanguageID 时会被自动设置为该语言的 LanguageID。

**property** DefaultFont: TFont;

该语言的默认字体，内部使用 FontStr 字符串存储。可读写。

## 4、 方法说明

**function** IsValidLanguageID(ALanguageID: LongWord): Boolean;

功能：判断一 ID 是否是合法的语言 ID。

返回值：是合法的 ID 则返回 True，否则返回 False。

参数：

1. ALanguageID: LongWord; 待判断的 LanguageID。

## 5、 事件说明

**property** OnLanguageIDChanged: TNotifyEvent;

事件，当语言 ID 发生改变时触发。

## 6、 组件编辑器说明

**property** LanguageID: LongWord;

该属性的属性编辑器能下拉显示当操作系统支持的所有语言 ID 和名称供选择。选择某语言 ID 后，对应的缩写、文件名和目录名等属性都会被自动设置。

## 九、 ICnLangStringIterator 接口

### 1、 接口功能概述

ICnLangStringIterator 是 TCnCustomLangStorage 及其子类可以提供的，用来遍历多语存储组件内部某一语言的所有字符串条目的遍历器接口实例。它主要用于多语管理器按字符串条目来遍历应用程序进行翻译的场合。比如 CnHashLangStorage 中的 TCnHashStringIterator 便是 TCnHashLangFileStorage 用来实现遍历接口的类。

### 2、 所在文件

CnLangStorage.pas 多语存储组件的基类单元。

### 3、 属性说明

**property** Eof: Boolean;

此接口的此属性返回当前是否已经到所有条目的结尾。只读

**property** Bof: Boolean;

此接口的此属性返回当前是否已经到所有条目的开始。只读。

### 4、 方法说明

**procedure** StartIterate(**const** FrontPattern: **string** = '');

功能：根据提供的首字符串过滤条件开始一次新的遍历。

参数：

1. **const** FrontPattern: **string**; 匹配的字符串，默认为空，表示匹配所有语言标识字符串。如不为空，则获得的字符串标识必须以 FrontPattern 开头。

**procedure** Previous;

功能：跳至前一符合过滤条件的语言条目。

**procedure** Next;

功能：跳至后一符合过滤条件的语言条目。

**procedure** EndIterate;

功能：结束遍历，做清理工作。

**procedure** GetCurrentKeyValue(**var** Key: **string**; **var** Value: **string**);

功能: 返回当前的条目字符串中的字符串标识和字符串值。

参数:

1. **var Key: string;** 返回当前语言条目的字符串标识, 比如 “TForm1.Caption”。
2. **var Value: string;** 返回当前语言条目的字符串值, 比如 “标题”。

**function** GetCurrentString: **string;**

功能: 返回当前的条目字符串, 为 “字符串标识=字符串值” 的形式。比如 “TForm1.Caption=标题”。

**function** GetEof: Boolean;

功能: 返回 Eof 的属性值, 标识当前是否已经到所有条目的结尾。

**function** GetBof: Boolean;

功能: 返回 Bof 属性的值, 标识当前是否已经到所有条目的开始。

## 5、 事件说明

无。