**Проект утилит Utils для TurboX**

Оглавление

[Класс SetClass (множество) и его энумератор 2](#_Toc16502260)

[Класс MapClass (мэп) и его энумератор 4](#_Toc16502261)

[Класс RunBaseClass 6](#_Toc16502262)

[Класс IEnum и его инфраструктура 7](#_Toc16502263)

## Класс SetClass (множество) и его энумератор

Класс Utils.SetClass.SetClass (в Utils.Def определен как AxSet) является аналогом аксаптовского класса множества Set.

Класс является сортированным хранилищем уникальных однотипных элементов.

Экземпляр класс создается с помощью статического метода Construct(aType: integer), где параметр – это элемент перечисления типов, которые возвращает функция VarType (полный список доступен в стандартном файле помощи по адресу kernel.Программирование.Классы.System.Прочие.ТипыДанныхVarTypes), например varInteger или varString.

**Основные свойства и методы класса:**

SetType: integer – тип данных (varType) элементов множества.

Length: integer – количество элементов в множестве.

IsEmpty: Boolean – признак пустоты множества.

Add(aValue: variant): boolean – добавление элемента в множество (если элемент уже существует – метод возвращает False).

Exists(aValue: variant): boolean – проверка существования элемента в множестве.

Remove(aValue: variant): boolean – удаление элемента из множества (если такого элемента не было – возвращает False).

Equal(aSet: AxSet): boolean – проверка равенства другому множеству.

GetEnumerator: SetEnumerator – создает энумератор для текущего множества.

**Статические методы:**

Construct(aType: integer): AxSet

Difference(aSet1: AxSet; aSet2: AxSet): AxSet – возвращает разницу двух множеств.

Intersection(aSet1: AxSet; aSet2: AxSet): AxSet – возвращает пересечение двух множеств.

Union(aSet1: AxSet; aSet2: AxSet): AxSet – возвращает объединение двух множеств.

Энумератор (класс Utils.SetClass.SetClassEnumerator, определен в Utils.Def как SetEnumerator) служит для перебора множества.

**Содержит следующие основные методы**:

Статический Construct(aSet: axSet): SetEnumerator – создает энумератор для указанного множества.

Current: variant – возвращает текущий элемент множества (на который указывает энумератор).

MoveNext: Boolean – перейти на следующий элемент множества (если элементов больше нет – возвращает FALSE)

Reset – сбросить позицию энумератора на ноль.

**Пример использования класса:**

var set1: AxSet;

var se: SetEnumerator;

set1 = AxSet.Construct(varInteger);

set1.Add(1);

set1.Add(2);

se = set1.getEnumerator;

while (se.MoveNext) do

Trace(str(se.Current));

end;

## Класс MapClass (мэп) и его энумератор

Класс Utils.MapClass.MapClass (в Utils.Def определен как AxMap) является аналогом аксаптовского класса Map.

Класс является словарем, в котором хранятся однотипные значения с однотипными ключами, отсортированными по значению ключа. Типы значений и ключей могут быть различными.

Экземпляр класс создается с помощью статического метода Construct(aKeyType: integer; aValueType: integer), где параметры – это элементы перечисления типов, которые возвращает функция VarType (полный список доступен в стандартном файле помощи по адресу kernel.Программирование.Классы.System.Прочие.ТипыДанныхVarTypes), например varInteger или varString.

**Основные свойства и методы класса:**

KeyType: integer – тип данных (varType) ключей мэпы.

ValueType: integer – тип данных (varType) значений мэпы.

Length: integer – количество элементов в мэпе.

IsEmpty: boolean – проверяет, является ли мэп пустым.

GetEnumerator: MapEnumerator – создает энумератор для текущей мэпы.

Lookup(aKey: variant): variant – возвращает значение элемента по заданному ключу; генерирует ошибку, если такого ключа нет.

LookupIfExists(aKey: variant): variant – возвращает значение элемента по заданному ключу; не генерирует ошибку, если такого ключа нет.

Remove(aKey: variant): Boolean – удаляет элемент по заданному ключу. Если такого элемента нет – возвращает FALSE.

Exists(aKey: variant): boolean – проверяет наличие ключа в мэпе.

Insert(aKey: variant; aValue: variant) – вставляет в мэп заданное значение с заданным ключом, заменяя старое значение, если ключ уже существовал.

Энумератор (класс Utils.MapClass.MapClassEnumerator, определен в Utils.Def как MapEnumerator) служит для перебора множества.

**Содержит следующие основные методы**:

Статический конструктор Construct(aMap: axMap): MapEnumerator – создает энумератор для указанного мэпа.

Reset – сбросить позицию энумератора на ноль.

MoveNext: boolean – перейти на следующий элемент (если элементов больше нет – возвращает FALSE).

CurrentKey: variant – возвращает ключ текущего элемента мэпа (на который указывает энумератор).

CurrentValue: variant – возвращает значение текущего элемента мэпа (на который указывает энумератор).

**Пример использования класса:**

var map: AxMap;

var me: MapEnumerator;

map = AxMap.Construct(varString, varInteger);

map.Insert('a', 1);

map.Insert('b', 2);

map.Insert('c', 3);

me = map.getEnumerator;

while (me.MoveNext) do

Trace(str(me.CurrentKey) + ‘ -> ‘ + str(me.CurrentValue));

end;

## Класс RunBaseClass

## Класс IEnum и его инфраструктура

Класс Utils.Enum.IEnum является оберткой класса Com.Fld.IField для упрощения работы с перечислениями. Также служит для разграничения отдельных перечислений по собственным классам для удобочитаемости, позволяет уменьшить дублирование кода при инициализации контролов на формах.

В инфраструктуру входят:

**Класс Utils.Enum.EnumElement**

Класс является структурой из двух полей – целочисленного значения Value и строки с именем значения Name. Также в классе перекрыт метод Equal, для сравнения (и обеспечения уникальности) элементов энума по числовому значению.

**Расширение класса Com.Frm.ICardForm.**

Расширение добавляет метод AddFieldEnum\_Col, аналогичный стандартному методу Com.Frm.ICardForm.AddFieldCol (используется для добавления столбца на TAB-форму). У метода есть нестандартный необязательный параметр aSkipValues: integer[], содержащий те элементы перечисления, которые вы хотите исключить из показа в контроле.

**Расширение класса Com.Ctrl.IControlsLayout.**

Расширение добавляет метод AddFieldEnum\_Edit, аналогичный стандартному методу Com.Ctrl.IControlsLayout.AddFieldEdit (используется для добавления на FRM-форму эдит-контрола). У метода есть нестандартный необязательный параметр aSkipValues: integer[], содержащий те элементы перечисления, которые вы хотите исключить из показа в контроле.

**Для использования IEnum необходимо сделать следующее:**

1. Создать наследника класса. Сам тип перечисления для удобства лучше всего создавать именно в этом наследнике.

class inherited Utils.Enum.IEnum;

import Utils classes Def;

inclass public

type SimpleEnum = UtilsTest.Enum.SimpleEnum;

type Values = (

zeroValue = 0,

firstValue = 1,

secondValue = 2,

fourthValue = 4,

seventhValue = 7,

tenthValue = 10

);

1. Перекрыть в нем метод Init для инициализации набора значений энума и имен этих значений.

inobject public

proc Init;

inherited Init;

Self.AddEnumValue(EnumElement.CreateEx(zeroValue, "value 0"));

Self.AddEnumValue(EnumElement.CreateEx(firstValue, "value 1"));

Self.AddEnumValue(EnumElement.CreateEx(secondValue, "value 2"));

Self.AddEnumValue(EnumElement.CreateEx(tenthValue, "value 10"));

Self.AddEnumValue(EnumElement.CreateEx(fourthValue, "value 4"));

Self.AddEnumValue(EnumElement.CreateEx(seventhValue,"value 7"));

end;

1. В классе REC для вашей записи вместо определения переменной типа IField для вашего поля надо определить переменную типа UtilsTest.Enum.SimpleEnum. Соответственно, в методе Init надо инициализировать эту переменную специальным образом.

class inherited Com.Rec.IRecord;

inclass public

var RecordClass: Class Record := UtilsTest.EnumTest;

inobject public

var enumField: UtilsTest.Enum.SimpleEnum;

proc Init;

inherited Init;

enumFIeld = UtilsTest.Enum.SimpleEnum.CreateEx(AddField("enumField"));

end;

1. В ТАБ-классе использовать метод AddFieldEnum\_Col вместо метода AddFieldCol

class inherited Com.Frm.ICardForm;

import Com classes Def;

inclass public

var RecordClass: Class Com.Rec.IRecord := Rec.EnumTest;

inobject private

var Current: Rec.EnumTest;

proc Init;

inherited Init;

AddFieldEnum\_Col('Enum field', 100, Current.enumField);

end;

1. В FRM-классе использовать метод AddFieldEnum\_Edit вместо метода AddFieldEdit

class inherited Com.Frm.IEditorWithFrames;

import Com classes Def;

inclass public

var RecordClass: Class Com.Rec.IRecord := Rec.EnumTest;

inobject private

Var FRecord: Rec.EnumTest;

proc Init;

inherited Init;

with FieldsFrame do

StartGroupEx(3, 100, , ,[ -consts.StdLabelWidth, 1/3, -consts.StdLabelWidth, 1/3, -consts.StdLabelWidth, 1/3 ]);

AddFieldEnum\_Edit('Enum', FRecord.EnumField, , , , [UtilsTest.Enum.SimpleEnum.firstValue, UtilsTest.Enum.SimpleEnum.seventhValue]);

end;

end;

end