Fun with Enumerators

Primož Gabrijelčič
About me

• Primož Gabrijelčič
  • http://primoz.gabrijelcic.org
  • programmer, MVP, writer, blogger, consultant, speaker
  • Blog http://thedelphigEEK.com
  • Twitter @thedelphigEEK
  • Skype gabr42
  • LinkedIn gabr42
  • GitHub gabr42
  • SO gabr
Fun with enumerators

Boy, was this an interesting trip.

For the last six days I was writing a series of articles on Delphi enumerators, one day each. In some way, this was very similar to something else I like to do - writing magazine articles on computer-related topics. So similar that I planned this series exactly as I'm planning an article. In some other way, it was also very different. Later posts I adapted based on feedback from earlier ones. For example, Part 6 was not included in the original article outline. This topic came to my mind while I was reading reader comments. In a way, it was like working with a very eager editor who is checking every chapter immediately I'm finished with it. Or, if you want, it was similar to pair programming.

In a way, writing this series was more like writing a book. If that's so, I have something more to write - a table of contents. It will help new readers to read whole series or just find the part they are interested in. So without further ado, here is the

Table of Contents

Part 1 - Introduction

Contains a short introduction on Delphi iterators (for, in statement) and describes Delphi support for iterator extensibility.

Part 2 - Additional enumerators

Shows how to add an additional enumerator to a class that already contains one.

Part 3 - Parameterized enumerators

This chapter takes Part 2 topic one level further by introducing enumerator parameters.

Part 4 - External enumerators

In this chapter you'll learn how to create enumerators without changing the class they are enumerating.

Part 5 - Class helper enumerators

Shows how to create additional enumerators using class helpers and how to use same technique to add enumerators to classes that don't have one.

Part 6 - Generators
Books

- Parallel Programming with OmniThreadLibrary
  - [http://tiny.cc/pg-ppotl](http://tiny.cc/pg-ppotl)

- Delphi High Performance
  - [http://tiny.cc/pg-dhp](http://tiny.cc/pg-dhp)

- Hands-On Design Patterns with Delphi
  - [http://tiny.cc/pg-dpd](http://tiny.cc/pg-dpd)
• Delphi Programming Projects by William Duarte
• The Complete Guide to RAD Server by David Intersimone
• Delphi Cookbook - Third Edition by Daniele Spinetti and Daniele Teti
• Delphi Memory Management For Classic And ARC Compilers by Dalija Prasnikar
• Expert Delphi by Paweł Głowacki
• Coding in Delphi, More Coding in Delphi, Dependency Injection in Delphi by Nick Hodges
• Delphi XE2 Foundations by Chris Rolliston
• Delphi Succinctly by Marco Breveglieri

• http://glooscapsoftware.blogspot.com
ENUMERATORS
For-in

• **for [var]** `element in collection do`
• *Iterator pattern*

• `collection = set`
  
  string
  
  array
  
  “collection”

• `element = readonly!`
Collection enumeration

- Class/interface/record: T
  - public function GetEnumerator(): E

- E: class/interface/record
  - public function MoveNext(): boolean
  - public property Current: V, readonly
    - function GetCurrent: V

var collection: T;
for var element: E in collection do
  DoSomething(element);
var collection: T;

for var element: E in collection do
  DoSomething(element);

var collection: T;
var element: E;
var enum := T.GetEnumerator;

while enum.MoveNext do
  DoSomething(enum.Current);

enum.Free; // if required
- System.Classes.TList
- System.Classes.TCollection
- System.Classes.TStrings
- System.Classes.TInterfaceList
- System.Classes.TComponent
- Vcl.Menus.TMenuItem
- Vcl.ActnList.TCustomActionList
- Vcl.ComCtrls.TListItems
- Vcl.ComCtrls.TTreeNodes
- Vcl.ComCtrls.TToolBar
- Data.DB.TFields
- Data.DB.TDataSet
Access to private data

• Enumerator needs access to private data!

• Possible solutions
  • Enumerator “knows” about internal implementation 😞
  • Enumerator = internal class/interface/record 😊
  • Enumerator = collection itself 😊
    • Interfaces/records only!
Multiple iterators

- X = class/record
  - GetEnumerator(): XEnumerator
  - AnotherEnumerator(): AnotherFactory

- AnotherFactory = record
  - GetEnumerator(): AnotherEnumerator

- TDictionary<K,V>
  - .Keys
  - .Values
Reusing enumerators

• GetEnumerator returns existing enumerator
Creative use

• Chaining enumerators
  • Spring4D
  • .Skip(3).Take(10).Reverse

• Enumerating external entities
  • Files
  • Network interfaces
  • ...

• Enumerating without data
  • Enumerator as a factory
Enumerators “on the budget”

• For..in works on arrays, so...
• ...just return TArray<T>
  • Slower, but simpler