

Back to basics: Enumerators

Primož Gabrijelčič

@thedelphigeek

BACK TO BASICS: ENUMERATORS



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](#)*

Monday, September 30, 2019

CompareValue for booleans

CompareValue function is incredibly practical when you are writing comparers (functions that determine how some data structure is ordered). *System.Math* and *System.StrUtils* define a bunch of functions that can be used to compare integers, doubles, strings ... There's, however, no *CompareValue* for booleans.

A *CompareValue* function compares two parameters, traditionally named *left* and *right*, and returns *0* if they are the same, *-1* if *left* is smaller and *1* if *right* is smaller.

If we use the usual ordering of *false* < *true*, we can write the missing function as follows:

```
function CompareValue(left, right: boolean): integer; overload;
begin
  if left < right then
    Result := -1
  else if left > right then
    Result := 1
  else
    Result := 0;
end;
```

Your task for today – if you choose to accept it – is: Write this function without any *if* statements.

[Read more »](#)

Posted by [gabr42](#) at [20:22](#) 9 comments: [Links to this post](#) 

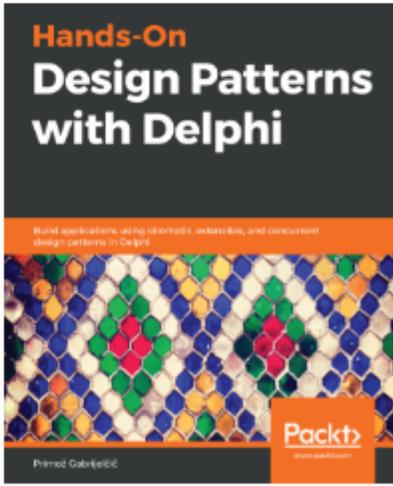
Labels: [Delphi](#), [programming](#), [tips](#)



Embarcadero
MVP

Pages

[Presentations](#)

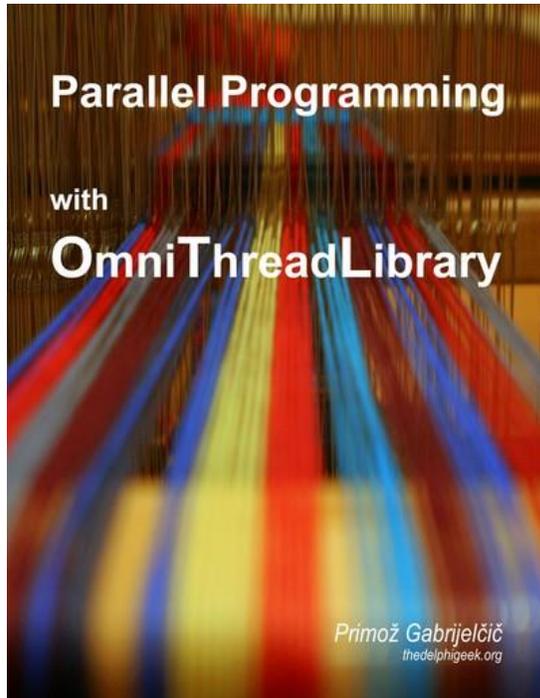


Hands-On
Design Patterns
with Delphi

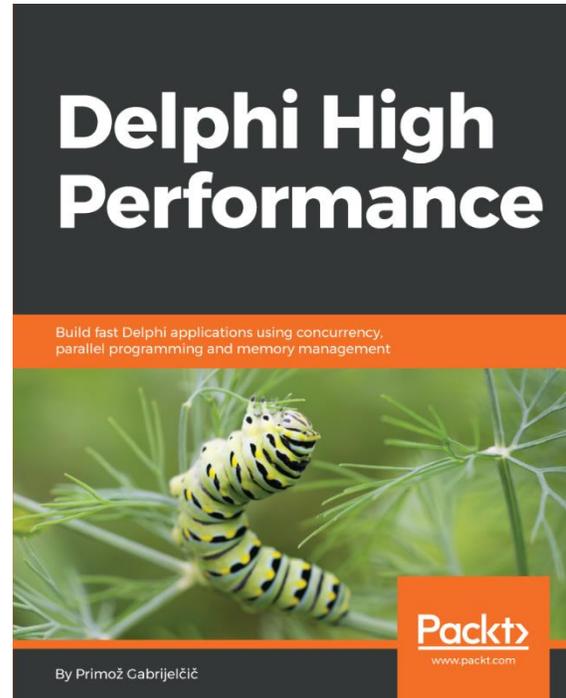
Build applications using observables, actors, and concurrent design patterns in Delphi

Packt

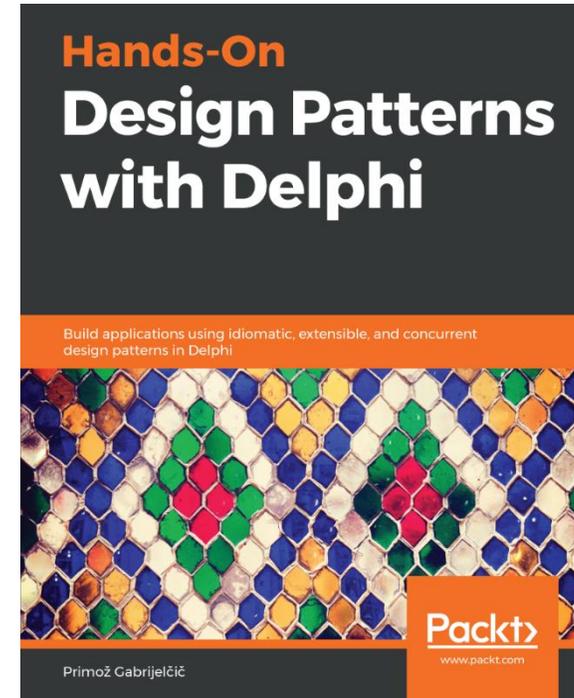
Primo Gabriel



[http://tiny.cc/
pg-ppotl](http://tiny.cc/pg-ppotl)



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



[http://tiny.cc/
pg-dpd](http://tiny.cc/pg-dpd)

BACK TO BASICS: 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);
```



Hidden implementation

```
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.TTreeNode](#),
[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!
 - Only one enumerator at the time!



Multiple iterators

- **for** *element in collection*.*AnotherEnumerator* **do**
- X = class/record
 - GetEnumerator(): XEnumerator
 - AnotherEnumerator(): AnotherFactory
- AnotherFactory = record/interface
 - GetEnumerator(): AnotherEnumerator
- TDictionary<K,V>
 - .Keys
 - .Values



Reusing enumerators

- GetEnumerator returns existing enumerator
- Useful when class wrap another collection (TList<T> ...)

Creative use

- Chaining enumerators
 - Spring4D
 - `collection.Skip(3).Take(10).Reverse`
- Enumerating external entities
 - Files
 - Network interfaces
 - ...
- Enumerating without data
 - Enumerator as a factory

SAYING ALL THAT ...



Enumerators “on the budget”

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

Q&A