Differences between VO and X#

The major differences between VO and X#

Wolfgang Riedmann
wolfgang@riedmann.it

Meeting Italiano 2018 a Bolzano

Riedmann

Runtime

A X# application is executed under the control of the .NET runtime and not anymore under the control of a proprietary runtime. This changes a few things:

- Stabler runtime, more performant garbage collector
- Available on all .NET platforms: x86, x64, AnyCPU, Linux
- Some features of the VO language are not available anymore. For example: access/assign cannot have the same name as a method, a class cannot have the same name as the enclosing binary
- The repository with its incremental compiling does not exists anymore, the code is located in prg files, but can be organized in folders and subfolders
- The ErrorBlock is not available anymore!!!
- The largest code library is available at no cost: the .NET Framework with thousands of classes
- Faster codeblocks, but slower in the compilation (for now!)
- Function() does not exists anymore, is emulated by static methods of the class Functions. You can continue to write and use functions.
- Access/Assign does not exists anymore, is emulated with property. An assign cannot return any value!

Riedmann

Runtime? What is it?

As runtime we specify the libraries/DLLs than come with :

- **The real runtime: data types, base functions, macrocompiler.**
- VO 2.8: VO28RUN.DLL
- Vulcan: VulcanRT.DLL, VulcanRTFuncs.DLL, VulcanMacroCompiler.DLL
- X#: Xsharp.Core.DLL, Xsharp.VO.DLL, Xsharp.MacroCompiler.DLL

- **Data access functionality: RDD (Replaceable database drivers)**:
- VO 2.8: CavoDBF.RDD, DBFCDX.RDD, DBFNTX.RDD, _DBFCDX.RDD
- Vulcan: VulcanDBFCDX.DLL, VulcanDBFFPT.DLL
- X#: currently in development, beta planned for August 2018

- **Class Libraries (System, GUI, RDD, SQL, OLE, Windows API, Internet)**:
- VO 2.8: VO28SYS.DLL, VO28GUI.DLL, VO28RDD.DLL, VO28SQL.DLL, VO28OLE.DLL
- Vulcan: VulcanVOSystemClasses.DLL, VulcanVOGUIClasses.DLL, VulcanVORDDClasses.DLL, VulcanVOSQLClasses.DLL
- X#: will be created by a tool from your own VO source code. GUI classes based on WinForms and SQL classes based on ADO.NET arriving too!

Everything is an object

Every data type is an object, even basic data types like "string", "int", "logic"

This means that even these datatypes can have methods and properties:

MyString:ToUpper()
MyString:Length
nNumber:ToString()

Riedmann

Unicode !!!

In .NET all strings are Unicode.

- No problems with national character sets anymore
- A character is NOT a byte! A string of 10 characters is 20 bytes long (normally)
- String != array of char != array of byte
- Attention to the conversions!
- You cannot more use easily memo fields for binary content (compressed or crypted data, etc.). The content would be destroyed by the automatich conversions.

Riedmann

String, Char, Byte

To read from a file on disk:
File.ReadAllBytes(cFileName) -> aContent as byte[]
File.ReadAllText(cFileName, Encoding.ASCII) -> cContent as string

To convert:
System.Text.Encoding.Unicode.GetString(aBytes) -> aBytes as byte[] -> cString as string
System.Text.Encoding.Unicode:GetBytes(cString) -> cString as string -> aBytes as byte[]

between string and array of char:
cString := String{ aChars } -> aChars as char[]
aChars := cString:ToCharArray() -> aChars as char[]

Riedmann

Literals, Suffix, Prefix

A string in X# è is delimited with the double apostrophe: " but in the VO dialect also the single apostrophe can be used. The parentheses [] cannot be used anymore.

A single character delimited with single apostrophe is not a string, but a char!

Attention to the dialects: Vulcan does not permits the single apostroph as string delimiters

Escaped string: e"Hi guys,\nwelcome to Bolzano, the capital of \"Südtirol\""
Interpolated string i"Hi {cNome}"
Prefix to specify a char in VO dialect: c'A'

Suffixes for numbers:

d – double	10d
s – single	10s
m – decimal	10m
b – binary	10b (=2)

Namespaces and Assemblies

The namespaces are helping to organize your own classes, and to avoid conflicts.

It is possible to have two different classes with the same name in two different namespaces.

The „Using“ statement declares the used namespace, but it is optional, then you need to specify the fully qualified class name.
Separator between class and namespace : the dot "."

System.Collections.Generic.List{}

Attention: Assembly and namespace are two different things!!!! Microsoft gives a bad example because they mix namespaces and assemblies (more different namespaces in one assembly, and namespaces sparsed over multiple assemblies)

Riedmann

Dot or colon?

In VO we use the colon to access the methods and assign/access of classes, and the dot for the members of structures.
In X# we need delimiters on more occasions:

- Method/Assign/Access/Property: colon
- Static method: dot
- Separatore di namespace: dot
- Structure: dot

Attention: some uses have requested that colon and dot are treated the same, and in some places this should work! (C# uses only the dot, so the Roslyn compiler make no distinction)

Riedmann

Try – catch – end try

Since the application does not run under the VO runtime control anymore, we have to adjust to the .NET Framework way.
Attention: without error handling the application could terminate without error message or with a message that gives no meaningful message!

```
try

System.IO.File.WriteAllText( "c:\meeting_2018\test.txt", "Ciao ragazzi" )

catch oEx as ArgumentNullException
    // path is null or contents is empty.
    MessageBox.Show( self, "cartella non specificata", "Errore" )
catch oEx as ArgumentException
    // path is a zero-length string, contains only white space, or contains one or more invalid characters as defined by InvalidPathChars.
    MessageBox.Show( self, "cartella non valida", "Errore" )
catch oEx as PathTooLongException
    // The specified path, file name, or both exceed the system-defined maximum length. for example, on Windows-based platforms,
    // paths must be less than 248 characters, and file names must be less than 260 characters.
    MessageBox.Show( self, "nome cartella troppo lungo", "Errore" )
catch oEx as DirectoryNotFoundException
    // The specified path is invalid (for example, it is on an unmapped drive).
    MessageBox.Show( self, "cartella non trovata", "Errore" )
catch oEx as IOException
    // An I/O error occurred while opening the file.
    MessageBox.Show( self, "errore in scrittura del file", "Errore" )
catch oEx as UnauthorizedAccessException
    // path specified a file that is read-only.
    // or: This operation is not supported on the current platform.
    // or: path specified a directory.
    // or: The caller does not have the required permission.
    MessageBox.Show( self, "nessun diritto a scrivere il file", "Errore" )
catch oEx as NotSupportedException
    // path is in an invalid format.
    MessageBox.Show( self, "percorso in formato non ammesso", "Errore" )
catch oEx as SecurityException
    // The caller does not have the required permission.
    MessageBox.Show( self, "errore di sicurezza", "Errore" )
catch oEx as Exception
    MessageBox.Show( self, oEx:Message + CRLF + CRLF + oEx:StackTrace, "Exception" )

end try
```

Global Error Handlers

There is not more the possibility to set a global error handler. Error handling in .NET is local, no more global like in VO!
But we can (and should) set error handlers for errors occurring in WinForms, WPF, etc.

AppDomain.UnhandledException: handles otherwise not handled errors
Application.ThreadException: handles errors occurring in WinForms
Application.DispatcherUnhandledException: handles errors in WPF code
Please see: https://docs.xsharp.it/doku.php?id=exceptions

Please pay attention to "begin sequence – end sequence" in applications migrated from VO!
Often in these blocks the errors are not handled.

Most important properties of the Exception class:
Exception:Message
Exception:Stacktrace
Exception: InnerException

Riedmann

Interface

Interface is an entirely new possibility to define the interface of a class.

Every class that implements an interface must have all methods and properties that are defined in the interface

Variables can be defined "as IName"

In VO for such purposes classes needed to have the same father class, using interfaces the classes don‘t need to have any relation. Please see the documentation for the interface IDisposable

Convention: every interface definition name starts with an uppercase i "I"

Riedmann

Problems with float? Decimal!

Float in Clipper and VO, and therefore also in Vulcan and X# is a „floating point“ data type, with a floating comma and a exponent.

A number „123456789.1“ internally could be „123456789.0999999999“ .
This can lead to rounding problems since
„123456789.1 == nVariable“ could return false. In fact, with the float datatype a number with a decimal part cannot represented correctly.

In the .NET Framework there is a datatype „decimal“ with high precision, where these sort of errors does not occur anymore. Unique problem: it is slower.

Very good notice: in the X# runtime the „decimal“is supported directly (or better: used internally)!
You can store a decimal in a usual, and it will be a decimal(27).

Riedmann

Performance

The performance of VO e X# is very difficult to compare.

X# has been developed with performance in mind. But code executed in the .NET runtime is slower than native code like VO.

Optimizations:

- Interal all arrays are typed
- Macros are slower in compilation, but faster in execution because is effectively compiled code
- Garbage collector enhanced, stable and fast, in AnyCPU all the memory can be used, not only 2 GB

Riedmann

End of session

Thank you for your attention!

https://www.xsharp.info
https://docs.xsharp.it
https://www.xsharp.info/help/index.html