alpha Software®

Casting the .NET

What, Why and How of .NET in Alpha Anywhere

Kurt Rayner
Vice President Research & Development
Alpha Software

Agenda

- History and a Working Definition of .NET
- Concepts and Terms - XBasic terms and syntax where possible
- Using .NET in XBasic
  - How Alpha Anywhere Integrates with .NET
  - Helper Classes
  - Creating References and Instances
  - Mapping Types
  - Collections and Enumerators
- Examples
- Gotchas

What is .NET?

History

Decades of Technical Learning and Application - Experience

- **Program Translation** - Static compilation versus interpreters
- **Module Integration** - DLL/COM/OLE/IBM DOM (not browser DOM)
- **Platform Independence** - Virtual Machines
  - Java run-time (language, byte code/pseudo code)
  - .NET (C#, VB.NET, C++/CLI, IronLisp, IronPython, IronRuby, IronScheme, Jscript.NET, J#, F#,P#, L#, PowerBuilder, NetCOBOL, Eiffel, Windows PowerShell)
- **Just-in-time** code generation (**JIT**)
- **Dynamism** – expression trees, dynamic language runtimes
- **True Parallelism** – asynchronous libraries, fork and join semantics
- **Competition** - Java ⇔ C#

Ref: https://en.wikipedia.org/wiki/List_of_CLI_languages

What is .NET?

Well…it's not…

- **A language?**

  Supports multiple languages…
- **An interpreter?**

  Well…not really…JIT…
- **A development environment?**

  Visual Studio is separate from .NET…
- **A Windows component?**

  Not just…Mono… .Net Core headed for Linux and Mac™
- **The answer** to the question of the meaning of life the universe and everything?

  Okay…seriously…

What is .NET?

A Framework

Common Language Runtime (CLR)

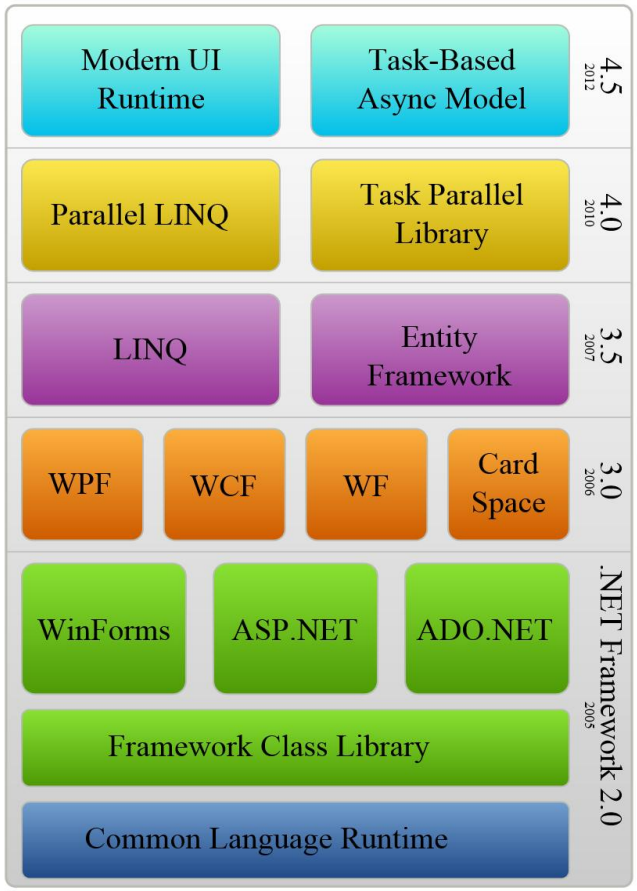
- A "player" (VM) for machine independent code
  - Security, memory management, exception handling
- A set of "containers" in which modules run

Framework Class Library (FCL)

- **Base Class Library (BCL)**
  - Types, basic file access, collections, attributes (custom, security), I/O streams, string manipulation.
- Classes for developers (written in C#)
  - ADO.NET, ASP.NET, Language Integrated Query (LINQ), Windows Presentation Foundation (WPF), Windows Communication Foundation (WCF)
  - Rich tooling interfaces (Reflection, CodeDOM)

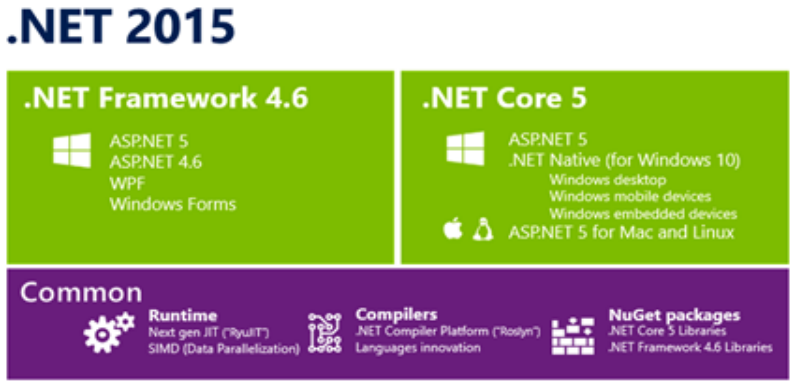
Ref: http://stackoverflow.com/questions/807880/bcl-base-class-library-vs-fcl-framework-class-library

The State of .NET



4.6 - July 20, 2015

- New high performance JIT
  - RyuJIT
  - 64-bit systems
- NSA "Suite B"
  - AES, SHA-2, elliptic curve Diffie-Hellman and elliptic curve DSA



Ref: https://en.wikipedia.org/wiki/.NET_Framework_version_history

Concepts

.NET Assemblies

- **Contain code and/or resources**
  (Classes, interfaces, enumerated types, static resources, generics)
- **Self describing**
  (Contain definitions and byte codes for CLR)
- **Loaded** into a **.NET application "domain"**
  (Can not be unloaded until the domain is terminated)
- **Can be generated at run time**
  (On disk or in memory)
- **DLL** extension on disk
  (An assembly can include more than one DLL.)

Concepts

Types and Namespaces

Primitive Types (string, blob, number, logical, date/time, GUID)

Objects

Classes - Define properties and methods (static/instance)

Instances – Created using a class/definition, have state

Interfaces – Look like an instance at runtime (classes implement)

Namespaces

Group and organize types (classes, interfaces, generics, enumerations) – prevent “namespace pollution”

Concepts

Types and Namespaces

Syntax for a Fully Qualified Type

[::] [<namespace [:: <namespace> …]] <class or type>

Examples

System::Text::StringBuilder
DotNet::Services
SQL::Connection
SQL::Query::JoinType

Concepts

Enumerated Types

Implemented as Classes In .NET and Constants in XBasic

SQL::Query::JoinType

System::IO::FileMode

Define Set of Unique Numeric Value and a Name Pairs

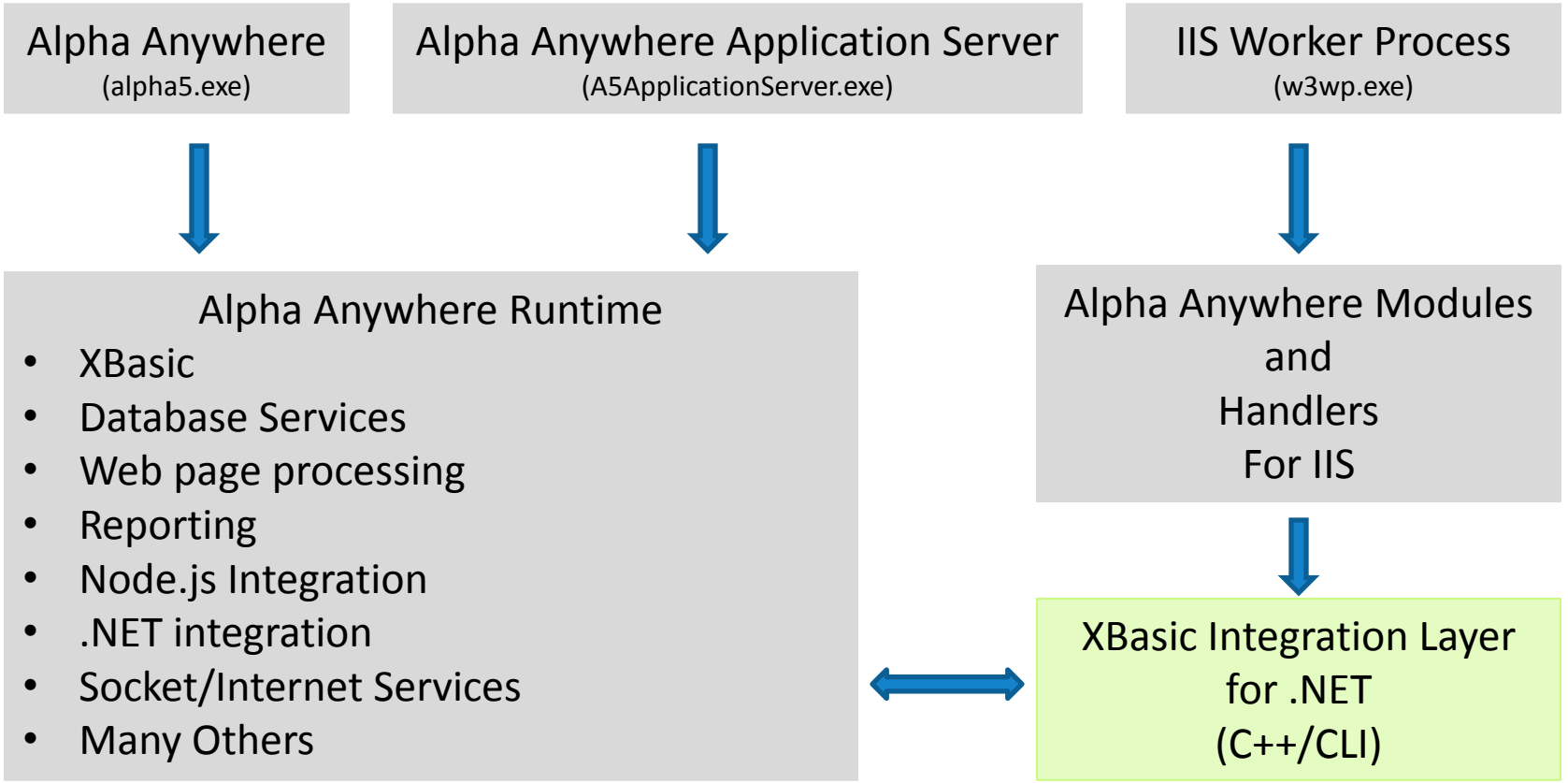
1	**CreateNew**
2	**Create**
3	**Open**
4	**OpenOrCreate**
5	**Truncate**
6	**Append**

Can be Referenced By Name Or Number

```
DIM JoinType as SQL::Query::JoinType
JoinType = SQL::Query::JoinType::Inner
?JoinType
= 0
JoinType = 1
```

Using .NET In XBasic

Integration



Using .NET In XBasic

Integration

- **Alpha Anywhere starts the .NET runtime**
  - Alpha Anywhere runs in the “default” application domain
  - Alpha Anywhere Application Server does too
  - Alpha Anywhere Application Server for IIS applications are run from the executable w3wp.exe and IIS creates multiple application domains

Using .NET In XBasic

Assembly References

Assemblies and their types **must be added to the XBasic type system** once per application domain.

- **Alpha Anywhere pre-loads Microsoft assemblies** for many of the more commonly used .NET types
  - **System** and **Microsoft** namespaces in the XBasic type system.
  - ***Never add ANY namespaces or types to either of these namespaces!***
- **Alpha Anywhere pre-loads it's own assemblies**
- **You register other types using classes in the DotNet namespace**
  - DotNet::AssemblyReference
  - DotNet::Services
  - Assemblies may be referenced **by name** if they are installed in the Global Assembly Cache (GAC) or **by file name**.

Using .NET In XBasic

DotNet::AssemblyReference

DIM Assy as DotNet::AssemblyReference

Set by File Name

Assy.FileName – “C:\temp\MyAssembly.dll”

Set Properties

Assy.Name = "System.Data"

Assy.Version = "1.0.3300.0"

Assy.Culture = "neutral"

Assy.PublicKeyToken = "b77a5c561934e089"

?Assy.FullName

= System.Data, Version=1.0.3300.0, Culture="neutral", PublicKeyToken=b77a5c561934e089

Using .NET In XBasic

DotNet::Services

Register an Assembly, a Namespace or a Class

L **RegisterAssembly**(C ParentNamespace, P Assembly)
L **RegisterClass**(C ParentNamespace, C AssignedClassName, C SourceClassName [, P Assembly = null_value])
L **RegisterNamespace**(C ParentNamespace, C SourceNameSpace, P Assembly)

Example

```
DIM Sv as DotNet::Services
DIM Assy as DotNet::Assembly Reference
Assy.FileName = "c:\Program Files (x86)\MyProject\MyAssy.dll"
?Sv.RegisterAssembly("::MyProject", Assy)

DIM Instance as ::MyProject::MyAssy::MyClass

?Instance.
```

Using .NET In XBasic

DotNet::Services

Static Method and Property Access (Alternate Methods)

- L CallStaticFunction(C FullyQualifiedFunctionName [, P Assembly = null_value], A Args)
- L GetStaticProperty(C FullyQualifiedPropertyName [, P Assembly = null_value], A Args)
- L SetStaticProperty(C FullyQualifiedPropertyName, A Value [, P Assembly = null_value], A Args)

Functions to Create Objects and Access Generics

- C ConstructGenericTypeName(C ClassName, C GenericTypes)
- L CreateObject(P Instance, C ClassName [, P Assembly = null_value], A Args)

Web Service Functions

- L GenerateWCFWebServiceClientFromURL(C URL, C AssemblyFileName)
- L GenerateWCFWebServiceClientFromWSDL(C WSDL, C AssemblyFileName)
- L GenerateWebServiceClientFromURL(C URL, C AssemblyFileName [, C Protocol = "Soap"])
- L GenerateWebServiceClientFromWSDL(C WSDL, C AssemblyFileName [, C Protocol = "Soap"])

Experimental Functions

L GenerateEventObject(C Language, C Script [, C Headers [, C References]])

Using .NET In XBasic
Creating Instances

Default Constructor

```
DIM Builder as System::Text::StringBuilder
```

Overloaded Constructor

```
DIM Builder as System::Text::StringBuilder = new System::Text::StringBuilder(10000)
```

Property Class Variable

```
DIM Builder as P
Builder = new System::Text::StringBuilder()
```

Note: Depending on the object, you may not have a default constructor.

Using .NET In XBasic

Type Conversions

How Types are Mapped

XBasic Type	.NET Type
C – String	System::String, System::Char
B – Blob	System::Byte[]
D – Date	System::DateTime
N – Numeric	System.Sbyte, System.Byte, System.Int16, System.Int32, System.Int64, System.UInt16, System.UInt32, System.UInt64, System.Double, System.Single, System.Decimal
P – PropertyClass	*A5DotNetReentry::XBasicObject*
T – Time	System::DateTime – Default for mapping to XBasic
U – Guid	System::Guid
Y – ShortTime	System::DateTime

Notes:

Arrays are converted to arrays of the corresponding type.

Underlined types are the mapping from XBasic to .NET

Function arguments are converted to match to the target type in the target function prototype.

Using .NET In XBasic

Type Conversions

It Just Works

```
DIM Builder as System::Text::StringBuilder
Builder.AppendLine("This is a new line.")
Builder.AppendLine("This is another new line.")
?Builder.ToString()
= This is a new line.
This is another new line.
?typeof(Builder.ToString())
= "C“

? System::Double::Parse("21") * 10
= 210

?System::Boolean::Parse("true")
= .T.
```

Using .NET In XBasic

Static Methods and Properties

Don’t Require an Instance

```
Text = <<%txt%
#Hello
#Mother
#Hello
#Father
#%txt%

System::IO::File::WriteAllText("C:\temp\MyFile.txt", Text)
?System::IO::File::ReadAllText("c:\temp\MyFile.txt")
= Hello
Mother
Hello
Father

?System::DateTime::UtcNow
= 09/09/2015 09:40:08 77 pm
```

Using .NET In XBasic

Fun With Enumerated Types

DIM Share as System::IO::FileShare

Set By Name

```
Share = System::IO::FileShare::Write
?Share.ToString()
= "Write“
```

Set By Number

```
Share.value__ = 1
?Share.ToString()
= "Read"
```

Parse a String (Can have Multiple Values)

```
Share = System::IO::FileShare::Parse(Share.GetType(), "Read, Write")
?Share.ToString()
= "ReadWrite"
```

Using .NET In XBasic

Enumerating Lists and Collections

XBasic's ForEach Operator Knows About Enumerators

```
Dim List as System::Collections::Specialized::StringDictionary
List.Add("FirstName", "Fred")
List.Add("LastName", "Jones")
for each k in List.Keys;   ? k + "= " + List[""" + k] + crlf();  next
firstname= Fred
lastname= Jones
```

Examples

Speech Synthesis

Register the Class

```
dim Sv as DotNet::Services
dim Assy as DotNet::AssemblyReference
Assy.filename = "C:\Program Files" +  if(Is64BitWindows(), " (x86)", "") + \
                                        "\Reference Assemblies\Microsoft\Framework\v3.0\System.Speech.dll"
Sv.RegisterClass("Speech", "Synth", "System.Speech.Synthesis.SpeechSynthesizer", Assy)
```

Synthesize Some Speech

```
dim TestTalker as Speech::Synth
dim Text as C = "Alpha Anywhere is the coolest product you have ever seen! Richard and Selwyn have made it easy to build web applications!!"
TestTalker.SpeakAsync(Text)
```

Examples

Microsoft Word Interop Library

Register the Class

```
dim Sv as DotNet::Services
dim Assy as DotNet::AssemblyReference
Assy.filename = ""C:\Program Files (x86)\Microsoft Visual Studio 10.0” + \
                                        ”\Visual Studio Tools for Office\PIA\Office12\Microsoft.Office.Interop.Word.dll"
Sv.RegisterAssembly("::", Assy)
```

Open a Microsoft Word Document

```
dim App as Microsoft::Office::Interop::Word::ApplicationClass
App.Documents.Open("c:\temp\Hello.docx")
```

Examples

File Streams with .NET

Open a Binary File the Hard Way

```
dim Stream as System::IO::FileStream = new System::IO::FileStream( \
                                                "C:\temp\Hello.txt", \
                                                System::IO::FileMode::Open, \
                                                System::IO::FileAccess::Read,\
                                                System::IO::FileShare::Read)
dim Bytes as B
dim Reader as System::IO::BinaryReader = new System::IO::BinaryReader(Stream)
Bytes = Reader.ReadBytes(1000)
showvar(Bytes.size())
showvar(base64encode(Bytes))
```

Gotcha's
"You can't get there from here!"

Try this in the interactive window:

Dim MyDate as System::DateTime = new System.DateTime()
ERROR: Not found
System not found.

dim MyDate as System::DateTime = now()
ERROR: Variable type mismatch: variable of type 'P' cannot be initialized to value of type 'T'.

Why not?

- Well known XBasic types are automatically mapped
- System::DateTime, System::String, System::Guid are all "immutable" – can't be changed in place

You CAN use static methods and properties like:

?**MyDate.IsLeapYear(2016)**
= .T.
?**System::DateTime::IsLeapYear(2016)**
= .T.
?**System::DateTime::Now**
= 09/09/2015 02:06:34 18 pm

Gotcha's
It's not as generic as you thought…

Generics don't play well with .NET Reflection and XBasic doesn't do generics

```
dim MyCollection as System::Collections::Generic::Dictionary`2
ERROR: Type 'System::Collections::Generic::Dictionary' not found dimming variable 'MyCollection'.
delete MyCollection
```

Use DotNet::Services to create instances of generic classes

```
DIM MyCollection as P
dim Sv as DotNet::Services
Name = Sv.ConstructGenericTypeName("System::Collections::Generic::Dictionary", "System.String", "System.DateTime")
?Name
= "System.Collections.Generic.Dictionary`2[System.String,System.DateTime]"
Sv.CreateObject(MyCollection, Name)
MyCollection.Add("Hello", now())
?MyCollection["Hello"]
= 09/09/2015 02:24:29 83 pm
```

Note: Generic methods (have a type in the name) can not be called through System::Reflection, so they are filtered out of the function list.

Gotchas

Memory Management a Slight Tangent

Variable Storage

Value

- Holds the data in it's own allocation
- Numbers, logical, character, date/time, structures, enumerations

Reference

- Points to another memory location
- Strings, arrays, class types, delegates

Memory Management

- Common Language Runtime (CLR) manages referenced storage
- Unreferenced memory is freed using ***garbage collection*** when memory "pressure" is high
- CLR can move referenced objects around to compact the managed heaps

Gotchas
Performance/Best Behavior

Garbage Collection

- Is complex and most programmers don't understand it (generations, large object heap, server mode, clean up thread).

  In most cases you are best to leave it alone.

- System::GC::Collect() can be used to force collection
- There are options to tune garbage collection (generations, large objects)

Gotchas
Performance/Best Behavior

Dispose

- Unmanaged resources (file and socket handles, GDI resources, database connections) can be explicitly freed to avoid waiting for garbage collection.
- A ”pattern” copied from Java
  - C# has a keyword for this - “using”
- XBasic can’t make assumptions and won’t call it.
- Some classes handle this in a Close() function

Questions?