Microsoft .NET y el Lenguaje C#

Clase I – El Framework .NET

El lenguaje C# no puede verse por separado del framework .NET, puesto que el compilador de C# compila específicamente para dicho framework. Por lo tanto es importante entender cómo está compuesto, al menos básicamente, el framework .NET.
Pero qué es .Net?

.NET es un framework de Microsoft que hace un énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones. Basado en ella, la empresa intenta desarrollar una estrategia horizontal que integre todos sus productos, desde el sistema operativo hasta las herramientas de mercado.

Common Language Runtime (CLR).

El núcleo del framework .NET es su entorno de ejecución llamado, en inglés, Common Language Runtime (CLR).
No obstante antes de ser ejecutado, todo código fuente debe ser compilado, (en C# o en cualquier otro lenguaje). La compilación de un programa .NET ocurre en dos instancias.

1 – Se compila el código fuente a codigo Microsoft Intermediate Language (MSIL).
2 – MSIL es compilado a código máquina específico de la plataforma.

Justamente tener un programa en código intermedio, (JAVA también produce código intermedio con Bytes Codes), permite que el mismo sea multiplataforma.

El código MSIL es compilado a código nativo de una forma particular. En lugar de compilar toda la aplicación de una, el compilador simplemente compila cada porción de código a medida que es invocado, de ahí que se conoce a estos compiladores como JITTERS o JIT, (“Just in Time”). El código nativo resultante es guardado hasta que la aplicación termine, así no necesita ser recompilado cada vez que es llamado.

Interoperabilidad entre lenguajes.

Otras de las características notables de usar un lenguaje intermedio es la posibilidad de escribir una aplicación en múltiples lenguajes, ya que todos generan el mismo código intermedio MSIL. Microsoft ha rediseñado y/o extendido varios lenguajes de programación clásicos para que puedan crear código MSIL, o sea, que usen el framework .NET.

Además de C# existen.

- Visual Basic .NET
- Visual C++/CLI (Common Language Infrastructure).
- Visual J#.
- ASP.NET

Por otro lado otras empresas también han adaptado sus lenguajes para poder generar código MSIL, como es el caso de Borland con Delphi y C++. Incluso existe una versión de Cobol .NET.

Common System Type (CST).

Teniendo presente la cualidad de interoperabilidad entre lenguajes .NET suponga que un método de una clase escrita en Visual Basic.NET retorna un valor *integer,* el cual es uno de los tipos de datos estándares en dicho lenguaje. C# no tiene un tipo de datos con ese nombre, obviamente uno podría derivar una nueva clase de ésta si el compilador de C# conociera como “mapear” el tipo de datos *integer* de Visual Basic.NET a alguno de los usados en C#.
Este problema es solucionado en .NET a través del uso del Common Type System (CTS). El CTS define los datos predefinidos que están disponibles en MSIL, o sea, todos los lenguajes que usen el framework .NET producirán código MSIL que estarán basados en esos tipos. No importa que en Visual Basic se llame integer y en C# int, siempre producirán el mismo tipo de datos en MSIL.

CTS Data Type	VB Keyword	C# Keyword	C++/CLI Keyword
System.Byte	Byte	byte	unsigned char
System.SByte	SByte	sbyte	signed char
System.Int16	Short	short	short
System.Int32	Integer	int	int or long
System.Int64	Long	long	__int64
System.UInt16	UShort	ushort	unsigned short
System.UInt32	UInteger	uint	unsigned int or unsigned long
System.UInt64	ULong	ulong	unsigned __int64
System.Single	Single	float	float
System.Double	Double	double	double
System.Object	Object	object	object^
System.Char	Char	char	wchar_t
System.String	String	string	String^
System.Decimal	Decimal	decimal	Decimal
System.Boolean	Boolean	bool	bool

De la misma forma una declaración en diferentes lenguajes .NET es sólo cuestión de formalidad:

```
// Definiendo unos enteros en C#.
int i = 0;
```

```
System.Int32 j = 0;

' Definiendo unos enteros en VB.NET.
Dim i As Integer = 0
Dim j As System.Int32 = 0
```

Common Lenguaje Specification (CLS).

El CLS trabaja junto con el CTS para asegurar la interoperabilidad.
Es un conjunto mínimo de estándares a los que todos los compiladores que pretendan generar código .NET deben ajustarse.

El CLS trabaja de dos maneras. Primero, aceptando que se puede escribir un compilador .NET sin necesidad de tener presente todas las características de .NET, y segundo, garantizando que si ese compilador restringe sus clases solamente a exponer la funcionalidad expuesta por el CLS, el código escrito en otro lenguaje .NET también podrá usar su clase.

Lo particular de esto es que las restricciones a usar sólo características aceptadas por el CLS, se aplica a miembros públicos y protegidos de las clases. En lo que respecta a la implementación privada de sus clases, se puede escribir código fuera del CLS porque de todas maneras nadie podrá acceder a ese código.

En resumen: el CLS son un conjunto de reglas que describen en detalle un mínimo y completo conjunto de características de un compilador .NET de manera tal que pueda producir código compilado (MSIL) entendible para el CLR y a la vez accesible de la misma forma para múltiples lenguajes de programación.

Recolector de basura (Garbage Collector).

El recolector de basura sería algo así como el paraíso de un programador de C/C++, ya que se encarga de administrar la memoria usada en tiempo de ejecución

Es un programa cuyo único propósito es limpiar la memoria. La idea es que toda memoria pedida dinámicamente es ubicada en el *montón* (*heap*), lo cual también se aplica a todos los lenguajes aunque no sean .NET, pero en .NET el CLR mantiene su propio *heap* a el cual le aplica el recolector de basura cada vez que un objeto sale de ámbito.

Actividades:

- Microsoft ofrece una gratuita: Visual Studio Community Edition.
  Descárguelo desde el siguiente link, instálelo y explore de forma intuitiva la interface.
  En las siguientes clases haremos uso intensivo de la herramienta.

https://visualstudio.microsoft.com/es/vs/community/

Si ya tiene instalada una versión de Visual Studio que incluye C#, no necesita bajar esta versión, puede usar la que ya tiene.