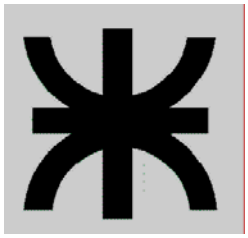

UNIVERSIDAD TECNOLOGICA NACIONAL
FACULTAD REGIONAL GENERAL PACHECO

TÉCNICO SUPERIOR EN PROGRAMACIÓN

PROGRAMACIÓN III
APUNTE TEORICO – UNIDAD 1
INTRODUCCION A .NET FRAMEWORK

ING. CLAUDIO FERNANDEZ
LIC. MARIANA BERTELLO

UNIDAD 1
INTRODUCCION A .NET FRAMEWORK

Introducción

.Net Framework es una plataforma creada por Microsoft para desarrollar aplicaciones. Estas aplicaciones pueden ejecutar sobre el entorno Windows pero también pueden ejecutarse en otros entornos No-Windows.

Como ejemplo podemos indicar que existe el .Net Compact Framework, un subconjunto del .Net Framework, que puede ser utilizado en asistentes personales digitales (PDA) y en algunos teléfonos inteligentes.

Uno de los objetivos de .Net Framework es permitir la integración de diferentes sistemas operativos.

.Net Framework permite el desarrollo de distintos tipos de aplicaciones entre ellas podemos enumerar:

- Aplicaciones de consola
- Aplicaciones de Windows Form
- Aplicaciones Web
- Aplicaciones de Servicio Web
- Otros tipos de aplicación que iremos descubriendo

Que hay dentro del .Net Framework

El .Net Framework es un gran librería de código que se utiliza desde alguno de los lenguajes clientes siendo el C# y el Visual Basic son los más conocidos.
Para poder utilizar o consumir esas librerías desde nuestras aplicaciones se utilizaran en forma casi excluyente técnicas de Programación Orientada a Objetos o POO.

Esta gran librería está dividida o categorizada en diferentes módulos y se utilizaran algunos de estos módulos dependiendo del objetivo final de nuestra aplicación.
Por ejemplo un módulo contiene los bloques para desarrollar aplicaciones Windows, otro módulo para aplicaciones de red, otro módulo para aplicaciones Web. Algunos módulos están compuestos por submódulos, como en el caso de las aplicaciones de Servicio Web, que forman parte del módulo de aplicaciones Web.

Parte de la librería .Net Framework define algunos *tipos de datos básicos*. Un tipo de dato es la representación de un dato y su especificación facilita la interoperabilidad entre los lenguajes del .Net Framework. A esto se lo llama *Common Type System* (CTS).

También forma parte del .Net Framework el llamado .Net *Common Language Runtime* (CLR) que es el responsable de permitir la ejecución de todas las aplicaciones desarrolladas utilizando .Net.

Desarrollando aplicaciones usando Microsoft Visual Studio 2008

Desarrollar una aplicación usando VS 2008 significa utilizar un poderoso entorno de desarrollo integrado (IDE) que soporta entre otros lenguajes al C#. La ventaja de este entorno es la facilidad con la cual las características de .Net pueden ser integradas a nuestro desarrollo.

Para que nuestro código C# ejecute, debe ser convertido a un lenguaje que nuestro Sistema Operativo comprenda, el llamado *Código Nativo*.
Esta conversión se llama *Compilación* y es ejecutado por un compilador. Sin embargo dentro de .Net Framework este proceso se lleva a cabo en dos pasos llamados MSIL y JIT, por sus siglas en inglés.

MSIL y JIT

Al compilar código dentro de .Net Framework no se crea inmediatamente código nativo sino que el código se compila a Microsoft Intermediate Language (MSIL). Este código no es específico para un Sistema Operativo y su generación no depende del lenguaje en el cual hayamos desarrollado nuestra aplicación. Esto significa que ya sea que programemos nuestra aplicación en C# o en Visual Basic, la primera etapa de compilación generará MSIL.

En una segunda etapa este código MSIL será compilado por el JIT para convertirlo en Código Nativo específico para el Sistema Operativo y arquitectura del equipo donde será ejecutado.

Esta novedad evita lo que sucede en otros entornos donde es necesaria la compilación del mismo código en distintas aplicaciones, cada una de ellas destinadas a un SO y CPU diferente.
El código MSIL es independiente de la arquitectura, SO y CPU. Existen distintos compiladores JIT, cada uno de ellos destinados a distintas arquitecturas, y se utilizará aquel apropiado para el generar el Código Nativo requerido.

Assemblies (Ensamblados)

Al compilar una aplicación el código MSIL creado se almacena en un *Assembly.* Este incluye archivos de aplicación ejecutables (.exe) y librerías (dll) que pueden ser utilizadas por otras aplicaciones.

Los assemblies contienen también *Metadata* es decir información referente al contenido del assembly y *Resources*, tales como archivos de sonido o imágenes.
Esto permite a los assemblies ser auto-descriptivos por lo tanto no se necesita información del registro, lo cual generaba muchos problemas en plataformas anteriores, situación conocida como DLL Hell, o infierno de las DLL, ya que al crear una aplicación la misma alteraba el registro y podía dejar inoperante a otra aplicación.

Distribuir una aplicación .Net es tan simple como copiar y pegar los archivos y carpetas en otra computadora. Al ser los assemblies unidades independientes, se puede ejecutar el archivo .exe directamente, asumiendo que el .Net CLR está instalado.

GAC

En el caso que se necesite escribir código que ejecuta tareas requeridas por múltiples aplicaciones es útil colocar ese código reusable en un lugar accesible para todas las aplicaciones. En el Framework a este lugar se lo llama *Global Assembly Cache* (GAC) y allí se coloca el assembly que contiene el código en el directorio que contiene a este cache.

Codigo Manejado o Administrado (Managed Code)

El código escrito que es compilado a MSIL y compilado por JIT es *manejado* cuando es ejecutado en la etapa de runtime.
Esto significa que el CLR controla las aplicaciones manejando la memoria, la seguridad, etc. Las aplicaciones que no ejecutan bajo el control del CLR se dice que son *unmanaged*, y algunos lenguajes como C++ pueden ser utilizados para crear esas aplicaciones, que pueden acceder a funciones de bajo nivel del SO.
En C# solamente se puede escribir código manejado.

Garbage Collection (Colector de basura)

Una de las características más importantes del código manejado es el concepto de Garbage Collection. Esta es la herramienta de .Net que permite asegurar que la memoria usada por una aplicación es liberada completamente cuando la misma finalice.
Previo a .Net esta tarea era responsabilidad del programador, y algún simple error de código podía resultar en grandes bloques de memoria ocupados innecesariamente lo que generaba una lenta y progresiva baja en el rendimiento de la aplicación.
El Garbage Collection trabaja inspeccionando la memoria en forma automática, esto significa que no es necesario código para su ejecución.

Linking

El código C# que es compilado a MSIL no necesita estar contenido en un único archivo físico y puede estar contenido en varios que luego son compilados en un único assembly, a este proceso se lo conoce como *linking*.
Es más fácil trabajar con varios archivos pequeños que con uno enorme. Se puede separar la aplicación en varios archivos individuales para ser trabajados en forma independiente por distintos programadores. Esto permite que se puedan controlar partes de código por separado sin poner en riesgo las partes que funcionan correctamente con aquellas que aún contienen errores.

Librerías de clases

La plataforma .Net contiene más de 5000 clases para ayudar a los programadores en el desarrollo de todo tipo de programas. Estas clases se agrupan en librerías y todas ellas están incorporadas en el entorno de ejecución de .Net.

Estas librerías son comunes para todos los lenguajes soportados por la plataforma, es decir, no hay una versión de la librería para C# y otra para VB. Net, sino que el mismo conjunto de clases puede ser utilizado por cualquiera de los lenguajes. Cada una de las librerías de la plataforma se encuentra compilada en un ensamblado de tipo .DLL.

Todas las librerías proporcionadas por .Net se pueden clasificar en dos grandes grupos:

- Librerías de clases base
- Librerías de entorno gráfico

Librerías de clases base

Las clases incluidas en las librerías de este grupo proporcionan las funcionalidades básicas que puede necesitar toda aplicación.
Algunas de estas librerías son:

- IO. Contiene las clases para las operaciones de entrada y salida de datos en un programa, por ejemplo para transferir información entre la aplicación y un archivo de disco.
- Collection. Esta librería incluye clases para el tratamiento de colecciones, las cuales representan una estructura de programación muy utilizada en las aplicaciones .Net.
- ADO. Net. Es una de las librerías más importantes de .Net. Sus clases dan el soporte para acceder a diferentes orígenes de datos.
- Base Class Library (BCL). Es la librería que contiene clases de uso general para cualquier aplicación, como la clase String para el manejo de cadenas de caracteres o Math para las operaciones matemáticas.

Librerías de entorno gráfico

Las clases incluidas en las librerías de este grupo proporcionan las funcionalidades para la creación de aplicaciones basadas en entorno gráfico. Las dos librerías más importantes que forman este grupo son:

- Windows.Forms. Incluye las clases para la creación de aplicaciones Windows. Proporciona los controles Windows clásicos.
- Asp .Net. Incluye todas las clases para la creación de aplicaciones de entorno Web.