



Lo que siempre quiso saber y nunca se atrevió a preguntar

¿Qué es .NET?

.Net es la nueva plataforma de desarrollo que ha lanzado al mercado MicroSoft, y en la que ha estado trabajado durante los últimos años. Sin duda alguna va a ser uno de los entornos de desarrollo que reinen durante los próximos años.

¿Entonces .NET es un lenguaje de programación?, No. Es mucho más que eso, Microsoft .NET es un conjunto de tecnologías de software, compuesto de varios lenguajes de programación que se ejecutan bajo el .NET Framework. Es además un entorno completamente orientado a objetos y que es capaz de ejecutarse bajo cualquier plataforma.

Vamos por partes. Podríamos dividir el entorno .NET en las siguientes partes:

- .NET Framework, que es el entorno de trabajo de la plataforma .NET y que la engloba completamente. Toda la plataforma .NET forma parte de .NET framework.
- Lenguajes .NET. Destacan C# y VB.NET, y recientemente J#, aunque existen más lenguajes (unos 49)
- El Common Runtime Language CRL, que es el motor de ejecución común a todos los lenguajes .NET.
- MSIL, Microsoft Intermedial language, es el lenguaje intermedio al que compilan las aplicaciones (Asemblies) .NET. Este lenguaje intermedio es interpretado por el CRL en tiempo de ejecución.
- CLS, common Language Specification, que engloban las pautas que deben cumplir los lenguajes .NET. Es esta característica la que va a permitir a otras compañías producir lenguajes compatibles con .NET.
- ADO.NET, es la nueva interfaz de bases de datos. No se trata de una evolución de ADO, sino que se trata de una interfaz completamente nueva.
- ASP.NET, es la nueva tecnología para páginas web dinámicas completamente integrada dentro del entorno .NET. Representa una autentica revolución en el desarrollo Web (Internet e Intranet).
- Biblioteca de clases .NET, es el conjunto de clases que componen el .NET framework y que nos permiten realizar casi cualquier tarea de una manera facil y rápida.

En la actualidad existen varias versiones del FrameWork .NET, estas versiones han sido mejoras sustanciales en cada una de ellas pero manteniendo la arquitectura, actualmente estamos en la versión 3.0, aunque recientemente se liberará la versión 3.5

MSIL, CRL y el código controlado.

Cuando escribimos un programa lo hacemos en un determinado lenguaje que podríamos llamar "humano" (aunque algunos se empeñen en decir que los programadores no somos humanos). Es decir, podemos leer y entender un programa (o al menos intentarlo) a través de un editor de texto, ya que este programa esta escrito en lenguaje "humano" (utilizando nuestra letras y esas cosas).

El único problema es que un ordenador no es capaz de entender nuestro programa, así que hay que traducirlo a su idioma. A este proceso se le conoce como compilación. Como resultado

del proceso de compilación obtenemos el programa ejecutable en código máquina, que entiende el ordenador pero no el "humano"(si alguien lo entiende no es humano, aquí sí).

Según la arquitectura del procesador, el sistema operativo, etc. este código es diferente y un programa que se ejecuta correctamente en un entorno **Windows** no funciona en **Macintosh** o **UNIX**. Es decir el programa sólo funciona para la plataforma para la que fue diseñado.

Este no es el proceso que ocurre en .NET. Cuando compilamos un programa escrito en cualquiera de los lenguajes .Net no se compila hacia código máquina nativo, sino que se hace hacia MSIL (Microsoft Intermediate Language), este MSIL es un lenguaje intermedio y universal. Cuando compilo un programa escrito en C# o en VB.Net ambos generan MSIL, con ciertas diferencias pero MSIL. Este código será interpretado posteriormente por un intérprete, el CLR. De este modo conseguimos que un programa escrito en .NET funcione en cualquier plataforma existente, incluso en plataformas futuras, sólo necesitamos construir el intérprete apropiado.

El MSIL es independiente del procesador, de la plataforma de desarrollo y de la plataforma de ejecución. El MSIL es parte del .Net Framework.

Llegados a este punto tenemos nuestro programa compilado a MSIL, pero el programa no funciona, ya que el procesador sólo entiende su propio código máquina nativo, y MSIL no lo es.

Es entonces cuando aparece el CLR (Common Language Runtime), o motor de ejecución común, que lo que hace es servir de traductor entre el MSIL y el código máquina nativo. Cuando ejecutamos un programa el CLR se encarga de compilar a código nativo dicho programa y ejecutarlo. A este tipo de compiladores se les conoce como compiladores JIT (Just In Time).

Si alguno de vosotros conoce **Java** habrá gritado ¡plagio, esto es el ByteCode!. No exactamente... cambian... los nombres. Teóricamente el CLR interpreta MSIL mucho mejor de lo que lo hace Java con el ByteCode y existen ciertas diferencias en la arquitectura interna, el CLR únicamente compila a código nativo la parte necesaria en cada momento durante la ejecución mientras que Java compila el programa completo, pero ...

De este modo podemos ejecutar nuestro programa sobre cualquier máquina, siempre y cuando exista una versión del .Net Framework y del CLR apropiada. Al código que se ejecuta bajo la batuta del CLR se le conoce como código manejado.

Assemblies

Un proyecto .NET no genera un ejecutable tal y como lo conocemos. Un proyecto .NET genera Assemblies. Un assembly es la unidad ejecutable de cualquier programa .NET, pero no se limita al código compilado sino que también incluye lo que se ha dado en llamar el manifiesto.

El manifiesto es un listado de las librerías (dll) y los controles (ocx) que son necesarios para que la aplicación funcione. Este listado contiene el número de versión necesario para que la aplicación funcione correctamente, impidiendo que la instalación posterior de un programa afecte a nuestro ejecutable.