Pointers.NET

Autor Ricardo Lejovitzky
Modificado el martes, 05 de agosto de 2008

Página 1 de 3Soporte para cÃ³digo no administrado en .NET Framework

La plataforma .NET es actualmente el entorno de desarrollo para aplicaciones administrativas y de redes mÃ¡s popular y avanzado existente para Windows y, mediante el uso de Mono Framework, Linux y otros sistemas operativos.

El rÃ¡pido crecimiento de .NET es explicable, principalmente, por razones de mercado. Sin duda que el impulso de una corporaciÃ³n dominante como Microsoft (y, en menor medida, de Novell al apoyar el desarrollo de Mono) no es algo que deba ignorarse en este anÃ¡lisis, pero, mÃ¡s allÃ¡ de estas consideraciones, hay otros fundamentos que hacen a .NET merecedor de esta progresiva hegemonÃ-a:

Â

- Fue diseÃ±ado desde el comienzo como un entorno independiente del lenguaje de programaciÃ³n. A diferencia de Java, sobre .NET se pueden encarar desarrollos en distintos lenguajes (C# y VB .NET son los mas populares), incluso en un mismo proyecto. Por ejemplo, mientras unos programadores pueden desarrollar la interface de usuario usando VB .NET, otros pueden abocarse a la lÃ³gica de negocio creando una librerÃ-a en C# o Delphi.NET.

- Cuenta con una Biblioteca de Clases (BCL) robusta, muy completa y con una organizaciÃ³n interna coherente, facilitando de asÃ- el desarrollo.

- El cÃ³digo compilado es, al igual que Java, independiente de plataforma (ByteCode). Gran parte de los ensamblados generados con .NET para Windows pueden ejecutarse, sin necesidad de recompilar, sobre Linux mediante Mono.

- Otra caracterÃ-stica que lo hace similar a Java es la facilidad en la depuraciÃ³n y la capacidad del compilador en tiempo real (JIT) de detectar errores no encontrados en la fase de compilaciÃ³n. TambiÃ©n se puede ajustar la seguridad del ensamblado final, haciendo que este solo funcione en un entorno apto y con los privilegios adecuados.

- Por ultimo, .NET supera a Java (plataforma en la que se inspira) al brindar un mecanismo de cache de ensamblados (ya compilados en cÃ³digo de maquina) y a la capacidad de sobrepasar los chequeos de seguridad mencionados anteriormente usando los servicios que brinda el sistema operativo en forma directa.

Entonces, un programa en .NET desarrollado en un lenguaje que soporte esta funcionalidad (C#, por ejemplo) puede acceder a direcciones de memoria (Punteros) sin intermediarios trabajando igual que el cÃ³digo compilado en lenguajes clÃ¡sicos como C o C++.

El uso de esta caracterÃ-stica de .NET, llamada cÃ³digo no administrado, no es algo que sea necesario ni recomendable para el desarrollo de aplicaciones habituales de tipo administrativo o gestiÃ³n tal como se advierte en la documentaciÃ³n oficial de MSDN:

"Para mantener la seguridad de tipos y la seguridad, C# de manera predeterminada no admite la aritmÃ©tica con punteros. Sin embargo, utilizando la palabra clave unsafe, es posible definir un contexto no seguro en el que se pueden utilizar punterosâ€¦

En el Common Language Runtime (CLR), se hace referencia al cÃ³digo no seguro como cÃ³digo no comprobable. El cÃ³digo no seguro en C# no es necesariamente peligroso; simplemente es cÃ³digo cuya seguridad no puede ser comprobada por el CLR. Por consiguiente, el CLR sÃ³lo ejecutarÃ¡ cÃ³digo no seguro si se encuentra en un ensamblado de plena confianza. Si utiliza el cÃ³digo no seguro, es su responsabilidad garantizar que su cÃ³digo no introduce riesgos

de seguridad o errores de puntero."

Entonces, Â¿En que situaciones es conveniente (o incluso necesario) usar CÃ³digo no administrado?

Â

- Cuando se trabaja con grandes volÃºmenes de datos frecuentemente, donde el cÃ³digo administrado puede afectar negativamente el rendimiento global de la aplicaciÃ³n. Esto puede ocurrir, por ejemplo, en rutinas de compresiÃ³n / descompresiÃ³n, de codificaciÃ³n / decodificaciÃ³n, de generaciÃ³n / verificaciÃ³n de checksums, algoritmos de ordenamiento complejos, etc.

- Cuando el tiempo de respuesta de la aplicaciÃ³n y la interacciÃ³n con el usuario es critico. Por ejemplo, cuando se usan rutinas de Video o 3D.

- Cuando se debe interactuar directamente con el hardware o utilizar funciones y librerÃ-as del sistema operativo (API).

- Cuando se desea economizar al mÃ¡ximo el uso de memoria de la aplicaciÃ³n y evitar copias de memoria redundantes.

Sintetizando, programando con punteros podemos aproximarnos a la economÃ-a, flexibilidad y potencia que nos brindan los compiladores clÃ¡sicos como C / C++ pero usando nuestro lenguaje preferido y sin salirnos de .NET; aunque, como contrapartida, tambiÃ©n nos aproximamos a los riesgos que justamente trae aparejado el uso de estos compiladores.

Es por estos riesgos que no todos los lenguajes soportan el cÃ³digo no administrado y por lo cual, para mantener una interface comÃºn y equitativa a todos los lenguajes, la BCL no brinda al programador facilidades para su uso.

Entonces, las dificultades para el programador se multiplican dado que ademÃ¡s de prestar la atenciÃ³n extra que requiere el desarrollo de este cÃ³digo, tambiÃ©n debe trabajar doblemente para desarrollar y depurar rutinas bÃ¡sicas como copiar memoria, ordenar, crear colecciones, etc. Es por eso, que esta caracterÃ-stica diferenciadora de Java y extensamente utilizada por Microsoft al desarrollar internamente la BCL, es muy poco usada por los programadores que desarrollan sus proyectos de .NET.

Pointers.NET es un proyecto de cÃ³digo abierto bajo licencia GPL.

Su principal ensamblado (enpalermo.dll) es una extensa librerÃ-a (de casi 100.000 lÃ-neas de cÃ³digo) que brinda este apoyo en las clases y funciones bÃ¡sicas que el programador requiere para poder abocarse exclusivamente al desarrollo de sus propias rutinas.

Anterior - Siguiente >>