



Pep Lluís Baño
Octavio Hernández



entrevista a

Lisa Feigenbaum

En esta ocasión, tenemos la suerte de contar con Lisa Feigenbaum, Community Program Manager del equipo de lenguajes de Visual Studio, una personalidad muy conocida y que goza de un gran prestigio entre la comunidad mundial de desarrolladores Microsoft.

Pep Lluís Baño es MVP Visual Developer (VB). Analista/programador desde 1979 y arquitecto de soluciones desde 1992. Actualmente, Pep es miembro activo de la comunidad de desarrolladores en España, siendo SpainNet UG Leader e INETA-Latam Líder del Comité de Oradores.

Octavio Hernández es MVP de C# desde 2004, MCSD y MCT. Autor de los libros "C# 3.0 y LINQ" y "ADO.NET Entity Framework: Aplicaciones y servicios centrados en datos" (con **Unai Zorrilla** y **Eduardo Quintás**).

[Octavio] Tengo entendido que comenzaste a trabajar en el equipo de Visual Basic. Me gustaría saber tu implicación actual dentro de ese equipo y para con la comunidad de desarrolladores de VB.

Comencé a trabajar en el equipo de VB.NET en 2004, como Program Manager para el IDE. En ese rol lo pasé genial diseñando diversas características del editor y el depurador de Visual Studio. Mi trabajo para el entorno integrado abarca desde VS 2005, pasando por VS 2008 y los primeros tiempos de VS 2010. Algunas de las características en las que he trabajado son IntelliSense, los *code snippets*,

los comentarios XML, corrección de errores, *edit-and-continue* y LINQ. En 2008, cambié de actividad y me convertí en Community Program Manager. Como parte de mi trabajo, creo y organizo contenidos para el equipo de lenguajes de Visual Studio. Esto incluye los sitios Web del Developer Center, blogs, fóruns, artículos, vídeos, *podcasts*, y también el programa MVP al que ambos pertenecéis. También me encargo de la planificación de diferentes conferencias de Microsoft. Finalmente, otra parte de mi trabajo consiste en interactuar con los clientes en las diversas ocasiones que se presentan, y recoger su *feedback* para utilizarlo diri-

giendo la estrategia y las futuras orientaciones de nuestro producto.

[Pep Lluís] Si los desarrolladores de C# —como Octavio— se interesan por tu rol e implicación para con VB, no me queda otro remedio que preguntarte lo mismo, pero refiriéndonos al equipo de C# y su comunidad de desarrolladores.

Para ser honesta, la respuesta podría ser “¡la misma!”. Desde que hemos unificado los equipos de VB y C# en 2008, y decidido evolucionar conjuntamente los dos lenguajes, encuentro más similitudes que diferencias en lo que constituyen las mejores actividades de la comunidad para ambos lenguajes. Tenemos dos centros de desarrollo, <http://csharp.net> y <http://msdn.com/vbasic>, y dos programas de MVP. Las conferencias a las que asisten los miembros de nuestro equipo, y los contenidos escritos y audiovisuales que producen, son aplicables igualmente tanto a C# como a VB.

[Octavio] Anteriormente hemos estado hablando sobre cierta controversia que se ha generado entre la comunidad de desarrolladores en torno a la idea de alcanzar una “paridad” en cuanto a características entre C# y VB. Muchos nos preguntamos si la plasmación de este objetivo no podría “difuminar” la identidad de cada uno de estos lenguajes.

No creo que eso sea así. Hoy, VB y C# han sido contruidos sobre el mismo CLR y pueden ser utilizados para acceder a las mismas tecnologías del *framework*. No obstante, ambos han logrado mantener sus identidades a través de su sintaxis individual, estilo y experiencia de uso del entorno respectivas. Creo que las cosas seguirán siendo así con la paridad de los lenguajes.

[Pep Lluís] ¿Me equivoco si afirmo que en la actualidad existe un número similar de desarrolladores orientados a aplicaciones utilizando C# y VB?

No, no te equivocas. Nuestros datos indican que la cantidad de desarrolladores profesionales que utilizan C# y la de los que utilizan VB son aproximadamente iguales.

[Octavio] Muchos de los que hoy utilizamos C#, y por supuesto los más “puristas” en primer lugar, opinan que el nuevo tipo de datos *dynamic* podría provocar efectos muy negativos en la calidad general del código... ¿Cuál es tu opinión al respecto? ¿Cuál es la experiencia acumulada al respecto en VB, donde *Option Strict Off* está presente desde siempre?

Antes de nada, debo decir que la nueva construcción *dynamic* ofrece enormes ventajas. Pone a nuestra disposición una manera mucho más elegante de traspasar las fronteras entre lenguajes y modelos de objetos, haciendo posible o facilitando escenarios como la interoperabilidad con Python o COM. Como bien comentas, aunque *dynamic* es tremendamente ventajoso en tales escenarios, no está pensado para ser utilizado en cualquier situación. El soporte IntelliSense para las variables *dynamic* es limitado, lo que pienso que influirá en que la gente lo utilice solo cuando sea apropiado. Finalmente, para los puristas, diré que es posible eliminar la referencia a `Microsoft.CSharp.dll`, y eso impedirá que se produzca ningún despacho dinámico.

En VS 2010, VB tendrá el mismo soporte para el DLR y la interoperabilidad entre lenguajes que se habilitan en C# a través de *dynamic*. En VB, esta funcionalidad se implementa utilizando el motor de enlace tardío de VB, que ha estado presente en el lenguaje desde antes de .NET, y que ha sido muy popular al interactuar con COM. También tendrá las mismas limitaciones de IntelliSense que hemos mencionado antes para el *dynamic* de C#.

El enlace tardío de VB requiere *Option Strict Off* (que es a lo que creo que se refiere la segunda parte de la pregunta). No obstante, quisiera aclarar un malentendido bastante extendido: es incorrecto decir que cuando *Option Strict* está en *Off* para todas las variables se hará siempre uso del enlace tardío. Incluso con *Option Strict Off*, es necesario optar explícitamente por el enlace tardío declarando la variable mediante `As`

`Object` (asumiendo que se está utilizando la inferencia de tipos, que está activa por defecto en VS 2008). Por lo tanto, *Option Strict On* no necesariamente hace nuestro código más “seguro en cuanto a tipos”. *Option Strict On* simplemente significa que el programador estará obligado a insertar los mismos *casts* que el compilador insertaría por nosotros si *Option Strict* estuviera en *Off*. Por lo tanto, el código resultante no es nada más seguro, dado que un *cast* siempre puede lanzar una excepción.

[Pep Lluís] Es curioso, pero en ocasiones cuando reviso código, ya sea en C# o VB, tengo la sensación de que la programación cada vez se basa menos en el uso de palabras claves; acostumbrado a los viejos programas, nunca puedo evitar esa visión. Hoy prácticamente toda la programación se basa en el uso de las clases de los espacios de nombres de .NET Framework. ¿Podrías darnos una estimación de la cantidad de nuevas clases que se añadirán en .NET 4.0 con relación a .NET 3.5 SP1? ¿Cuáles serán las principales novedades en este sentido?

Pertenezco al equipo de lenguajes, y por supuesto creo que los lenguajes siempre serán una parte muy importante del *stack*. No obstante, estoy de acuerdo en que desde hace tiempo se viene produciendo una gran cantidad de innovación en el *framework*, ¡y especialmente en .NET 4.0!

Un área que realmente me apasiona es el soporte para la computación paralela. .NET Framework 4.0 incluye una nueva librería que da soporte al paralelismo de tareas y datos (*Task Parallel Library*, TPL), una implementación paralela de LINQ to Objects (*Parallel Language Integrated Query*, PLINQ), y nuevas estructuras de datos para la sincronización y la concurrencia. VS 2010 incluirá soporte para esas novedades, incluyendo características de perfilado y depuración. Conjuntamente, .NET 4.0 y VS 2010 ofrecerán una vía revolucionaria para aprovechar toda la potencia de las máquinas multi-procesador y multi-núcleo.



Lisa Feigenbaum con Octavio Hernández

.NET 4.0 incluye una nueva versión del CLR (CLR 4). Entre las novedades de CLR 4 se incluyen los eventos ETW, contratos de código, inicialización perezosa, mejoras en el perfilado y la recolección de basura y soporte para la co-varianza y contra-varianza. También incluye el DLR, que habilita la ejecución de lenguajes dinámicos sobre .NET Framework, *in-process-side-by-side*, que hace posible cargar múltiples versiones del CLR en el mismo proceso, así como *no-PIA deployment*, que implica que los ensamblados de interoperabilidad realmente dejarán de ser necesarios (de hecho, los compiladores de C# y VB embeberán dentro de los resultados de proyectos las partes de esos ensamblados que sean necesarias, y el CLR garantizará la seguridad de tipos).

.NET 4.0 añade igualmente nuevas clases a las librerías base, como **BigInteger**, **SortedSet** o los tipos de tuplas, y mejoras en lo relativo a la E/S, reflexión, multi-hilado y el acceso al Registro. *Managed Extensibility Framework* (MEF) es otra nueva librería que permitirá desarrollar aplicaciones compuestas fácilmente extensibles. Varios componentes de VS 2010 han sido creados utilizando MEF. Finalmente, .NET 4.0 incluye algunos añadidos a WCF y un nuevo modelo de flujos de trabajo para WF.

[Octavio] La nueva versión 4 añadirá a C#, además del tipado dinámico, los parámetros nombrados y opcionales y ciertas facilidades para simplificar las llamadas a COM. ¿Qué puedes decirnos de esas características, existentes desde hace mucho en VB?

Esas características han tenido históricamente un gran éxito entre los desarrolladores de VB, y les pronóstico el mismo éxito en C#. La demanda de esas funcionalidades ha sido muy grande. Y para quienes programan aplicaciones Office en C#, VS 2010 marcará una gran diferencia.

[Pep Lluís] Pues si hablamos de VS 2010, es obligatorio repasar lo que nos viene a los de VB, y en ese sentido está claro que no vamos a tener ningún problema con las continuaciones de línea implícitas. Sin embargo, ¿crees que la implementación automática de propiedades, los inicializadores de colecciones y las expresiones lambda serán de fácil asimilación para los programadores de VB?

Creo que sí. La sintaxis de las propiedades automáticas es bastante directa, esencialmente la misma sintaxis expandida actual, pero solo la primera línea. Cuando comparo ejemplos de código para VS 2008 que utilizan propiedades en VB y C#, ¡quisiera que llegara VS 2010! Por otra parte, los inicializadores de colecciones se apoyan en la sintaxis de inicialización de objetos introducida en VB 8, así que espero que el concepto sea rápidamente familiar.

Para ser honesta, no anticipo al principio un gran uso de las sentencias *lambda* en VB, pero dada la presencia de esta característica en C# desde hace algún tiempo, se había generado cierta demanda al respecto. Las sentencias

lambda son muy útiles al programar contra diversas tecnologías; las extensiones paralelas (TPL y PLINQ) son un buen ejemplo.

[Octavio] Pues a falta los iteradores y la programación *unsafe* por un lado, y los literales XML por el otro, ¿casi estaremos a la par! Debe ser increíblemente reconfortante culminar este gran esfuerzo por parte de ambos equipos.

Nuestros planes para la evolución conjunta de C# y VB incluyen la adición simultánea de características importantes, y tapar los agujeros actuales en cuanto a diferencia de capacidades entre los dos lenguajes. El trabajo no terminará con VS 2010, nos llevará más tiempo. Tampoco pensamos alcanzar una paridad del 100%. Pensamos que a partir de cierto momento, la gente dará prioridad a las nuevas características sobre las pequeñas diferencias restantes.

Hemos hecho un progreso significativo hacia la evolución conjunta, añadiendo los inicializadores de colecciones, propiedades automáticas, literales de *array* y sentencias *lambda* a VB, y los parámetros opcionales y nombrados, el enlace tardío vía *dynamic* y la habilidad de omitir *ref* en las llamadas a COM a C#. Finalmente, hemos añadido a ambos lenguajes la interoperabilidad con lenguajes dinámicos, la co-varianza y la contra-varianza y el despliegue sin PIA.

Los iteradores probablemente llegarán a VB en versiones posteriores. Los literales XML para C# y el código no seguro para VB actualmente no entran en los planes.

[Octavio] Sé que te has comprometido mucho con las comunidades. Me gustaría conocer los planes de futuro y los nuevos recursos para los desarrolladores en los que estáis trabajando.

Actualmente me dedico a crear buenos contenidos para VS 2010, de modo que éstos ya estén disponibles cuando el producto salga al mercado. He creado recientemente dos nuevas páginas con recursos para VS 2010 en

los MSDN Developer Centers de C# (<http://msdn.microsoft.com/en-us/vcsharp/dd819407.aspx>) y VB (<http://msdn.microsoft.com/en-us/vbasic/dd819153.aspx>).

Además, en mis planes está incorporar más vídeos, *podcasts* y entrevistas a miembros del equipo (¡como ésta!). Pienso continuar incorporando a los Developer Centers contenidos creados por los MVP (como el artículo sobre LINQ que tú has escrito: <http://msdn.microsoft.com/en-us/vcsharp/aa336732.aspx>). También estaré involucrada en la organización de conferencias y eventos relacionados con Visual Studio.

[Pep Lluís]. Mucha gente siente curiosidad por conocer cuántas personas trabajan para Visual Studio como producto en Redmond, qué lenguaje utilizan los equipos de C# y VB, así como cuántas líneas de código se han necesitado para crear Visual Studio.

Ciertamente, es una tarea gigantesca gestionar todos los equipos y activos que componen Visual Studio. En total son 30 unidades de producto. Nuestra unidad (Visual Studio Languages) tiene alrededor de 100 miembros. En total, unas 1.000 personas trabajan en Visual Studio, y alrededor de 800 más se dedican a .NET Framework. Entre todos esos equipos, hay una enorme cantidad de código en constante cambio. Para mantener *builds* estables, hay un complejo sistema de ramificaciones, así como diversas medidas de control de calidad. Las características se desarrollan en ramas independientes, que no se integran a la línea principal hasta que están terminadas. Cada vez que se hace una integración, se realiza una serie de pruebas funcionales y de rendimiento para garantizar que la nueva característica no “rompe el *build*”. También desde un par de meses antes de cualquier *release* público, se lleva a cabo un riguroso proceso en el que los cambios a lo largo de toda la división son revisados por un comité centralizado llamado *ship room*.

Visual Studio está escrito en múltiples lenguajes, incluyendo C++, C# y VB. Hay alrededor de 42 millones de líneas de código en VS 2008. Para generar todo el producto desde cero se necesita una hora para sincronizar el código, 5-6 horas de compilación y otras 6-7 horas para generar los paquetes de instalación. Para más información sobre los procesos que tienen lugar al generar Visual Studio, recientemente hemos publicado una gran entrevista con **Matt Gertz**, Developer Manager del equipo encargado de ello: <http://channel9.msdn.com/posts/funkyonex/How-We-Do-It-Building-the-Visual-Studio-Product-Line>.

[Octavio] ¿Podrías arriesgarte y confesarnos qué característica es la que más te apasiona de Visual Studio?

Creo que Pep sabe cuál es mi característica favorita. ¡No puedo dejar de mencionarla en ninguna de mis presentaciones! Mi característica favorita es **IntelliSense**. IntelliSense ha sido parte del producto desde antes de .NET, y continúa siendo parte esencial de la experiencia de uso de VS.

[Pep Lluís] Conociéndote, sé que estos últimos años has trabajado mucho en sensibilizar a la gente sobre el aprovechamiento del IntelliSense que ofrece el entorno integrado de Visual Studio y en particular el uso de los *code snippets* en la interacción con el editor. Estoy convencido de que VS 2010 llevará una magistral implementación de todo ello. ¿Puedes contarnos, a grandes rasgos, los logros conseguidos en este apartado?

¡Están sucediendo muchísimas cosas interesantes en la interfaz de usuario de VS 2010! Para empezar, hemos rescrito el editor y el *shell* de VS en WPF, algo que se nota inmediatamente, no más ejecutar el producto. Muchas características orientadas al usuario final han sido añadidas como parte de este cambio. Se ha mejorado el soporte para múltiples monitores, que nos permitirá anclar ventanas de edición en diferentes monitores. Es muy fácil hacer zoom en el editor de código

utilizando la tecla [Ctrl] y el ratón. Ha mejorado el soporte para el formato del código y la selección de bloques. La nueva página de inicio y el nuevo diálogo de “Nuevo proyecto” ofrecen ahora capacidades de búsqueda. También están disponibles un nuevo gestor de extensiones y la Galería de Visual Studio, para facilitar la integración de extensiones. Y la lista continúa...

En lo que respecta a las características de los entornos para C# y VB, también hay múltiples novedades. Las llamamos “orientadas al código”, porque hacen posible una más fácil comprensión del código y una mejor navegación por él. Mi favorita es “Navigate To”. Al teclear [Ctrl][+], aparece un diálogo que ayuda a navegar hasta un fichero o miembro de la solución. Otra gran novedad es “Generate From Usage”, que permite hacer referencia a elementos de código antes de definirlos, y se apoya en el editor de código para generar las declaraciones para nosotros. Esta será una gran herramienta de productividad, especialmente interesante para quienes hacen TDD (*Test Driven Development*). Finalmente, dos nuevas características que ayudarán a una mayor comprensión del código son “Highlight References” y “Call Hierarchy”. “Highlight References” nos permite ver fácilmente cómo un símbolo es utilizado en el fichero de código actual, resaltando todas las referencias. “Call Hierarchy” ofrece una visualización de todas las llamadas hacia y desde un método del proyecto actual; sin embargo, actualmente solo funciona para C#.

[Octavio] ¿Algún comentario final para nuestros lectores?

Para la información más reciente y completa sobre Visual Basic y C#, visitad los respectivos Developer Centers: <http://msdn.com/vbasic> y <http://csharp.net>. Probad la Beta 1 de VS 2010, disponible aquí: <http://msdn.microsoft.com/en-us/vstudio/dd582936.aspx>. Finalmente, por favor, escribidme y dejadme saber qué tipos de contenidos sobre VB o C# desearíais ver en los Developer Centers: Lisa.Feigenbaum@microsoft.com. ○