El IDE de Visual Studio 2005

Luis Miguel Blanco Ancos

La renovación de nuestro banco de trabajo

Y cuando decimos las más destacadas, además hemos de puntualizar que realmente vamos a abordar sólo aquellas características que efectivamente despuntan como novedad o mejora relevante, ya que tal es el cúmulo de innovaciones, que posiblemente nos harían falta varios artículos para detallarlas todas.

Dado que podemos abordar este tema desde diferentes perspectivas, nos hemos decidido a detallar los nuevos aspectos del IDE de Visual Studio 2005 (VS2005 en adelante) siguiendo aproximadamente el ritmo que nos marcaría el trabajo con un proyecto Visual Basic de tipo Windows, es decir: creación del proyecto, diseño del interfaz visual, escritura del código y depuración.

Y ya sin más dilación vamos a entrar en materia, que hay muchas cosas que contar.

Creación de proyectos, configuración y ajustes del IDE

El modo de creación de un proyecto ha variado ligeramente en VS2005. Ahora, el cuadro de diálogo de nuevo proyecto agrupa dentro de cada lenguaje diversas categorías de plantillas de proyecto en función de las aplicaciones que tengamos instaladas en nuestra máquina, pudiendo elegir desde la clásica aplicación Windows, hasta la novedosa *Windows Presentation Foundation* (más conocida como Avalon). Ver la figura 1.

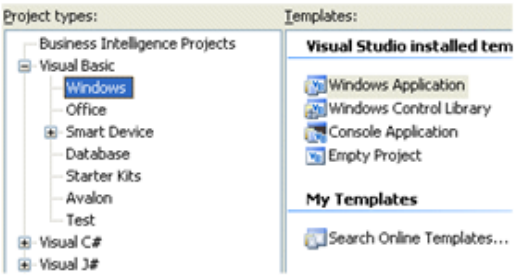

Figura 1.

El lector se preguntará a buen seguro dónde han ido a parar las plantillas para proyectos Web. El motivo reside en que ahora VS2005 separa totalmente este tipo de proyectos del resto. Si queremos crear una aplicación ASP.NET, tendremos que elegir la opción de menú “File > New > Web Site”, apareciéndonos el diálogo específico para la creación de proyectos Web: ASP.NET Web Site, Web Service, etc. En este diálogo también seleccionaremos el lenguaje de programación a utilizar, e incluso si queremos que el sitio Web sea creado al modo clásico, usando IIS, o en una ruta de archivos, con

lo cual se hará uso de un servidor Web virtual, lo que nos libera de la obligación de tener instalado IIS en la máquina de desarrollo.

Una vez creado el proyecto, debemos detenernos en su mejorado acceso y manipulación de propiedades, las cuales se muestran en una completísima ventana que las organiza en una serie de pestañas, desde las que podemos configurar todos los aspectos necesarios: nombre de ensamblado, formulario inicial y de presentación, referencias, recursos, etc. Ver la figura 2.

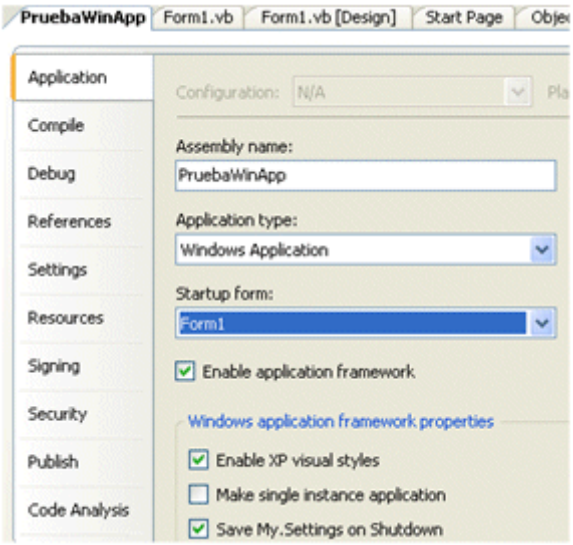

Figura 2.

Una de las novedades importantes a nivel de configuración del IDE, consiste en la capacidad de poder guardar todos los ajustes establecidos en el entorno de trabajo dentro de un archivo en formato XML con extensión .VSSETTINGS. Empleando el menú "Tools > Import and Export Settings" realizaremos esta operación mediante un sencillo asistente, con lo que podremos llevar nuestra configuración favorita a otra máquina, evitando el tedioso trabajo de tener que volver a configurar todo manualmente. La figura 3 muestra este asistente.

Figura 3.

Diseñadores y ventanas adicionales

Cuando vamos a diseñar un formulario, la caja de herramientas es el primer elemento al que solemos recurrir para arrastrar y soltar los controles que compondrán el interfaz de usuario; en esta versión, la ventana *Toolbox* contiene más categorías de controles, permitiéndonos tener desplegada más de una categoría al mismo tiempo.

Por otro lado, a la hora de alinear los controles en la superficie del formulario, contamos con unas estupendas guías visuales, que facilitan en gran medida su organización, y evitarán en muchos casos el uso de las opciones del menú *Format*.

También se van a ver reducidos nuestros "viajes" a la ventana de propiedades gracias a los *Smart Tags* (Etiquetas Inteligentes) que acompañan a muchos de los controles, y que vienen representados por un pequeño icono en forma de flecha, en la parte superior derecha del control. Al pulsar dicha etiqueta, aparecerá un pequeño diálogo con las operaciones más frecuentes que se realizan sobre el control, pudiendo efectuarlas "in situ". La figura 4 muestra un ejemplo del diseñador de formularios.

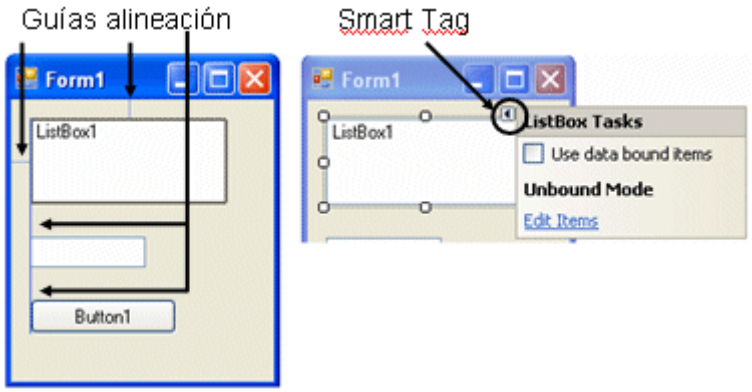

Figura 4.

Otro de los aspectos incómodos en el trabajo con el IDE, residía en el cambio de posición y acoplamiento de las diferentes ventanas que lo componen, ya que en algunos casos era realmente difícil depositar una ventana en el lugar deseado. A partir de ahora, esta tarea ya no va a revestir problema, puesto que al mover una ventana aparecerán los indicadores de posicionamiento que vemos en la figura 5; simplemente con situar la ventana sobre uno de ellos, quedará colocada en su nuevo emplazamiento.

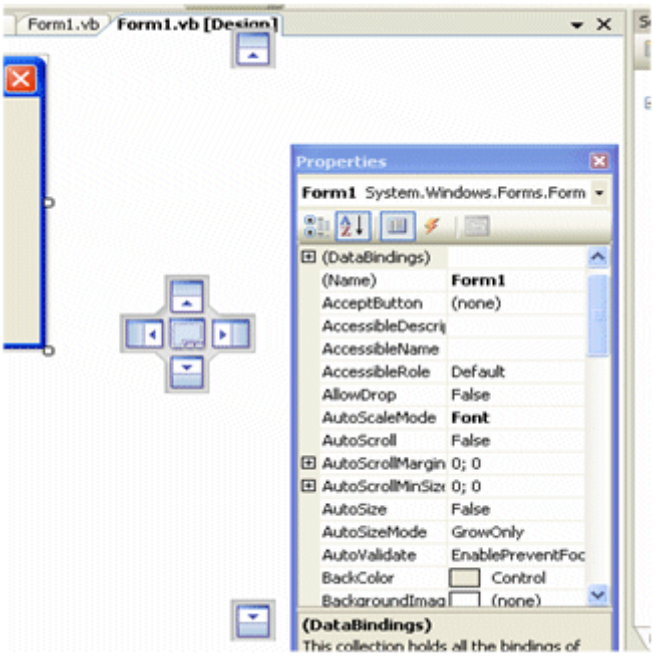

Figura 5.

Mejoras en el editor de código

Son las más importantes con diferencia, ya que el trabajo que dedicamos a la escritura de código, normalmente representa el porcentaje más alto de nuestro tiempo global de desarrollo.

En primer lugar, si abrimos el editor para ver el código de la clase del formulario, notaremos la falta del código generado por el diseñador, el cual se encuentra ahora dentro de una clase parcial, en un archivo aparte con una denominación del estilo *NombreFormulario.Designer.VB*, lo cual nos proporciona una mayor claridad y limpieza de código. Este archivo se encuentra oculto en el explorador de soluciones, por lo que debemos pulsar el botón para mostrar todos los archivos de la solución.

Una característica largamente demandada en Visual Basic ha sido la posibilidad de incluir comentarios XML, al estilo de C#, que nos permitan documentar adecuadamente el código. Pues bien, nuestro deseo se ha hecho realidad, ya que ahora, al escribir tres comillas simples en la cabecera de una clase, método o propiedad, se creará el esqueleto básico para documentarlo, y al hacer uso de ese elemento de código, los comentarios introducidos se visualizarán en el *ToolTip* correspondiente, como vemos en la figura 6.

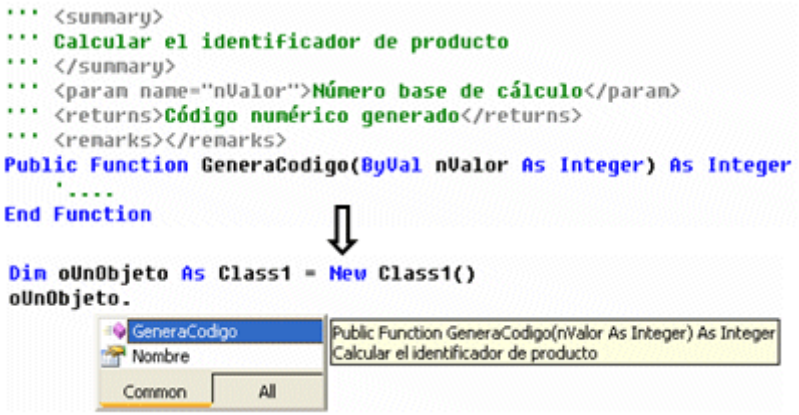

Figura 6.

Entre las incorporaciones realizadas a la tecnología *Intellisense* podemos destacar las opciones de auto corrección, que subrayan una palabra clave mal escrita, ofreciéndonos una lista de posibles correcciones. Igualmente será subrayado un identificador que pertenezca a un tipo que se intente utilizar antes de haber sido instanciado. Por otra parte, para que de un rápido vistazo podamos comprobar cuáles son las nuevas líneas de código que hemos escrito, en el margen del editor se muestran unos indicadores de color verde y amarillo, avisándonos de las líneas grabadas y las pendientes respectivamente. Veamos unos ejemplos en la figura 7.

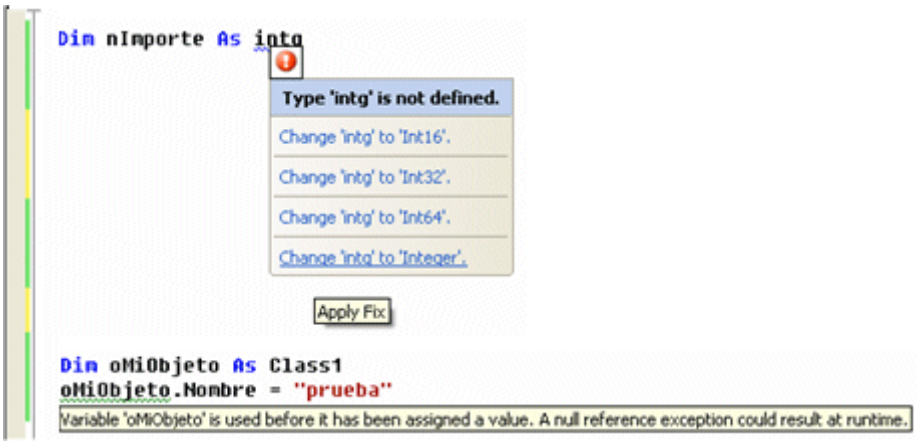

Figura 7.

Existen ciertas operaciones cuyo código normalmente repetimos hasta la saciedad en todas nuestras sesiones de trabajo. Para ahorrar tiempo en la escritura de dicho código, ahora disponemos de los *Code Snippets* (Recortes de Código), que consisten en bloques de código "prefabricados", almacenados en VS2005, y que pegamos directamente sobre el editor, ahorrando gran cantidad de tiempo de escritura.

El IDE nos proporciona una enorme cantidad de estas piezas de código para resolver multitud de situaciones, pero además, contamos con la posibilidad de crear nuestros propios recortes personalizados mediante el administrador de snippets.

Para agregar un code snippet hemos de hacer clic derecho sobre el editor y elegir la opción *Insert Snippet*, que abrirá una lista de carpetas conteniendo todos los diferentes bloques de código organizados por categorías, como vemos en la figura 8.

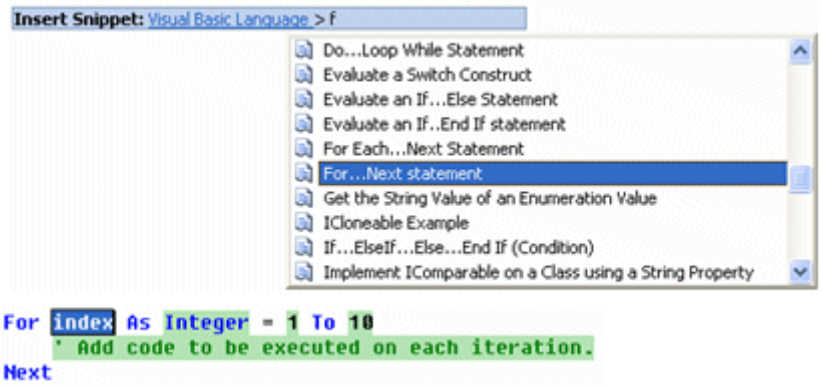

Figura 8.

A través del diagrama de clases, podemos diseñar visualmente el esqueleto de una clase, con la particularidad de que el editor de código y el diagrama se encuentran sincronizados, es decir, cualquier cambio que hagamos sobre una clase, por ejemplo en el editor, se propagará inmediatamente al diagrama y viceversa. La figura 9 nos muestra esta ventana del diagrama.

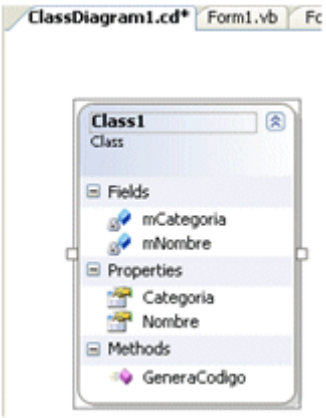

Figura 9.

Otro de los mecanismos que nos puede ayudar a agilizar la escritura es *Refactoring*, que nos permite reestructurar nuestro código de una manera rápida y fácil.

Por ejemplo, supongamos que estamos escribiendo un fragmento de código y estimamos que puede ser candidato a convertirse en un método independiente; lo que tenemos que hacer es seleccionar dichas líneas de código, y con un clic derecho sobre las mismas, elegiremos la opción “Refactoring > Extract method”, que abrirá un cuadro de diálogo en el que introduciremos el nombre de un nuevo método para las líneas seleccionadas, y *Refactoring* nos creará el método con dichas líneas, situando una llamada al mismo en el emplazamiento original, como vemos en la figura 10.

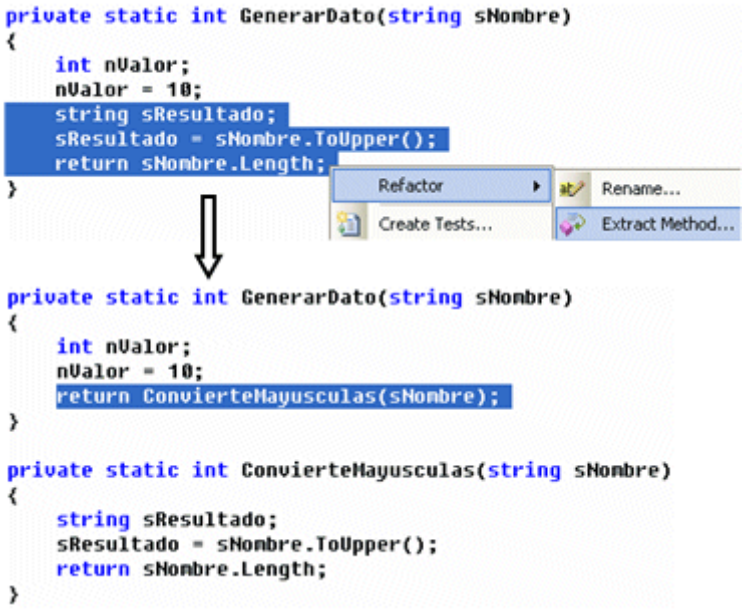

Figura 10.

En algunas ocasiones, para realizar pruebas, comenzamos a codificar usando los nombres por defecto que genera el IDE: Form1, Class1, etc., y al terminar, si la prueba es satisfactoria y queremos mantener el código escrito, nos vemos obligados a recorrerlo para cambiar dichos nombres por otros más significativos, lo cual resulta muy tedioso, sobre todo si hemos escrito bastante código.

Esta operación podemos realizarla ahora en un único paso, seleccionando el nombre a cambiar, haciendo clic derecho y eligiendo la opción *Rename*, la cual se encargará de propagar los cambios necesarios a lo largo de todo el código.

Novedades del depurador

En primer lugar tenemos que anunciar la vuelta al depurador de una de las características clásicas de Visual Basic: “Editar y Continuar”, ahora también disponible para C#. Sí en efecto, volvemos a contar en el IDE con la posibilidad de modificar el código al mismo tiempo que lo estamos depurando, lo que supone un importante aumento de productividad, evitando en muchas ocasiones tener que cancelar la ejecución, modificar y volver a ejecutar, con el retraso que ello supone en el tiempo global de desarrollo.

Respecto a la consulta de información en tiempo de depuración, tenemos como novedad los *Data Tips*, consistentes en una viñeta flotante que visualiza los valores contenidos en los miembros de una clase propia o de la plataforma .NET. Tan sólo hemos de situar el ratón sobre la variable que alberga el objeto, y el data tip con su contenido será visualizado. Adicionalmente, también tenemos visualizadores en diferentes formatos para estos valores, dependiendo de cuál sea el valor que estamos consultando; si hay un visualizador disponible lo sabremos porque a su lado aparece un icono de lupa. Ver la figura 11.

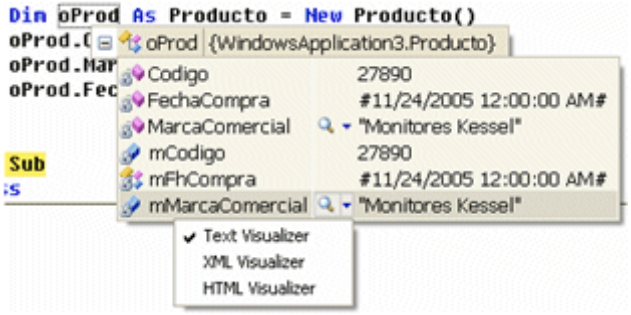

Figura 11.

Cuando se produce un error en tiempo de ejecución, entra en acción el "Asistente de Excepciones", una estupenda herramienta que muestra un cuadro de diálogo con la descripción de la excepción, una lista de posibles soluciones, y acceso a un completo detalle del problema, a la documentación de .NET, etc. Ver la figura 12.

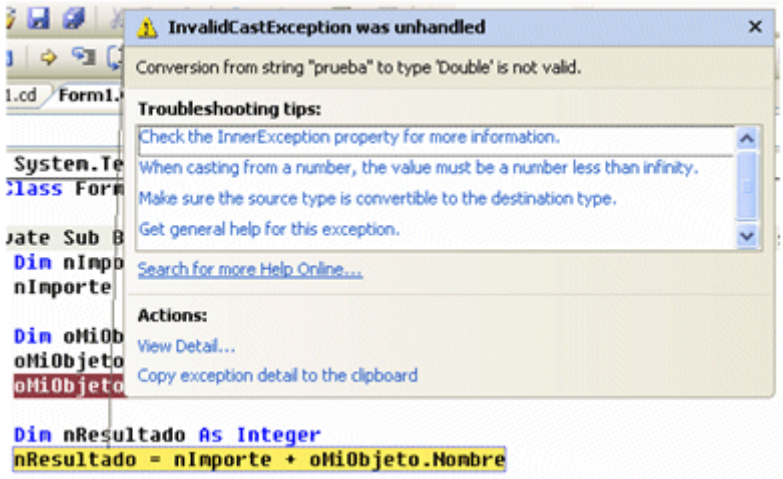

Figura 12.

Cerrando el IDE

Y esta ha sido la relación de primicias más remarcables con las que desembarca el nuevo y flamante Visual Studio 2005. Como hemos comentado al comienzo de este artículo, hay muchas más, entre las que podríamos mencionar de forma breve, la capacidad de la ventana Inmediato para evaluar expresiones en tiempo de diseño; la nueva ventana *Error List*, que muestra los errores producidos, por lo que ya no se usa la ventana *Task List* para este menester; los atributos del depurador, que permiten personalizar el modo en el que el depurador visualiza la información; el nuevo sistema de ayuda, con capacidades mejoradas de búsqueda, almacenamiento para entradas favoritas de la documentación, etc.

En definitiva, una nueva versión de un gran entorno de desarrollo de aplicaciones, que potencia nuestra capacidad de trabajo al tiempo que la simplifica.