Contenido del Manual

Parte 1: Información General Acerca de .NET Framework

Parte 2: Novedades en .NET Framework 3.5

- Lo Nuevo de .NET Framework Versión 3.5
- Lo Nuevo en Visual C#
- Lo Nuevo en Visual C++ 2008
- Lo Nuevo en el Lenguaje Visual Basic
- Lo Nuevo en ASP.NET y Desarrollo Web
- Lo Nuevo en .NET Compact Framework 3.5
- Lo Nuevo en Windows Presentation Foundation Versión 3.5
- Lo Nuevo en Visual Studio 2008
- Lo Nuevo en Visual Studio Team System (VSTS)
- Lo Nuevo en Visual Studio Tools para Office (VSTO)
- Novedades en ADO.NET
- Lo Nuevo en Visual Database Tools
- Lo Nuevo en Datos en Visual Studio 2008

Información General y Conceptual sobre .NET Framework

.NET Framework es un componente integral de Windows que admite la creación y la ejecución de la siguiente generación de aplicaciones y servicios Web XML. El diseño de .NET Framework está enfocado a cumplir los objetivos siguientes:

- Proporcionar un entorno coherente de programación orientada a objetos, en el que el código de los objetos se pueda almacenar y ejecutar de forma local, ejecutar de forma local pero distribuida en Internet o ejecutar de forma remota.
- Proporcionar un entorno de ejecución de código que reduzca lo máximo posible la implementación de software y los conflictos de versiones.
- Ofrecer un entorno de ejecución de código que fomente la ejecución segura del mismo, incluso del creado por terceras personas desconocidas o que no son de plena confianza.
- Proporcionar un entorno de ejecución de código que elimine los problemas de rendimiento de los entornos en los que se utilizan secuencias de comandos o intérpretes de comandos.
- Ofrecer al programador una experiencia coherente entre tipos de aplicaciones muy diferentes, como las basadas en Windows o en el Web.
- Basar toda la comunicación en estándares del sector para asegurar que el código de .NET Framework se puede integrar con otros tipos de código.

.NET Framework contiene dos componentes principales: Common Language Runtime y la biblioteca de clases de .NET Framework. Common Language Runtime es el fundamento de .NET Framework. El motor en tiempo de ejecución se puede considerar como un agente que administra el código en tiempo de ejecución y proporciona servicios centrales, como la administración de memoria, la administración de subprocesos y la interacción remota, al tiempo que aplica una seguridad estricta a los tipos y otras formas de especificación del código que fomentan su seguridad y solidez. De hecho, el concepto de administración de código es un principio básico del motor en tiempo de ejecución. El código destinado al motor en tiempo de ejecución se denomina código administrado, a diferencia del resto de código, que se conoce como código no administrado. La biblioteca de clases, el otro componente principal de .NET Framework, es una completa colección orientada a objetos de tipos reutilizables que se pueden emplear para desarrollar aplicaciones que abarcan desde las tradicionales herramientas de interfaz gráfica de usuario (GUI) o de línea de comandos hasta las aplicaciones basadas en las innovaciones más recientes proporcionadas por ASP.NET, como los formularios Web Forms y los servicios Web XML.

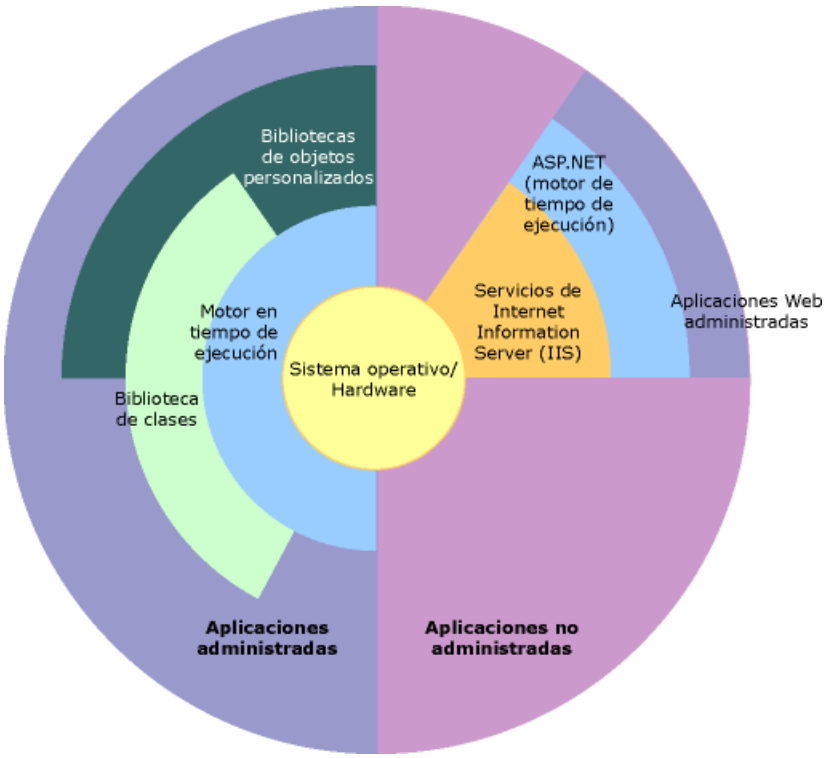
.NET Framework puede alojarse en componentes no administrados que cargan Common Language Runtime en sus procesos e inician la ejecución de código administrado, con lo que se crea un entorno de software en el que se pueden utilizar características administradas y no administradas. En .NET Framework no sólo se ofrecen varios hosts de motor en tiempo de ejecución, sino que también se admite el desarrollo de estos hosts por parte de terceros.

Por ejemplo, ASP.NET aloja el motor en tiempo de ejecución para proporcionar un entorno de servidor escalable para el código administrado. ASP.NET trabaja directamente con el motor en tiempo de ejecución para habilitar aplicaciones de ASP.NET y servicios Web XML, que se tratan más adelante en este tema.

Internet Explorer es un ejemplo de aplicación no administrada que aloja el motor en tiempo de ejecución (en forma de una extensión de tipo MIME). Al usar Internet Explorer para alojar el motor en tiempo de ejecución, puede incrustar componentes administrados o controles de Windows Forms en documentos HTML. Al alojar el motor en tiempo de ejecución de esta manera se hace posible el uso de código móvil administrado (similar a los controles de Microsoft® ActiveX®), pero con mejoras significativas que sólo el código administrado puede ofrecer, como la ejecución con confianza parcial y el almacenamiento aislado de archivos.

En la ilustración siguiente se muestra la relación de Common Language Runtime y la biblioteca de clases con las aplicaciones y el sistema en su conjunto. En la ilustración se representa igualmente cómo funciona el código administrado dentro de una arquitectura mayor.

.NET Framework en contexto



En las secciones siguientes se describen con más detalle los componentes y características principales de .NET Framework.

⊟ Características de Common Language Runtime

Common Language Runtime administra la memoria, ejecución de subprocesos, ejecución de código, comprobación de la seguridad del código, compilación y demás servicios del sistema. Estas características son intrínsecas del código administrado que se ejecuta en Common Language Runtime.

Con respecto a la seguridad, los componentes administrados reciben grados de confianza diferentes, en función de una serie de factores entre los que se incluye su origen (como Internet, red empresarial o equipo local). Esto significa que un componente administrado puede ser capaz o no de realizar

operaciones de acceso a archivos, operaciones de acceso al Registro y otras funciones delicadas, incluso si se está utilizando en la misma aplicación activa.

El motor en tiempo de ejecución impone seguridad en el acceso al código. Por ejemplo, los usuarios pueden confiar en que un archivo ejecutable incrustado en una página Web puede reproducir una animación en la pantalla o entonar una canción, pero no puede tener acceso a sus datos personales, sistema de archivos o red. Por ello, las características de seguridad del motor en tiempo de ejecución permiten que el software legítimo implementado en Internet sea excepcionalmente variado.

Además, el motor en tiempo de ejecución impone la solidez del código mediante la implementación de una infraestructura estricta de comprobación de tipos y código denominada CTS (Common Type System, Sistema de tipos común). CTS garantiza que todo el código administrado es autodescriptivo. Los diferentes compiladores de lenguajes de Microsoft y de terceros generan código administrado que se ajusta a CTS. Esto significa que el código administrado puede usar otros tipos e instancias administrados, al tiempo que se aplica inflexiblemente la fidelidad y seguridad de los tipos.

Además, el entorno administrado del motor en tiempo de ejecución elimina muchos problemas de software comunes. Por ejemplo, el motor en tiempo de ejecución controla automáticamente la disposición de los objetos, administra las referencias a éstos y los libera cuando ya no se utilizan. Esta administración automática de la memoria soluciona los dos errores más comunes de las aplicaciones: la pérdida de memoria y las referencias no válidas a la memoria.

Además, el motor en tiempo de ejecución aumenta la productividad del programador. Por ejemplo, los desarrolladores pueden crear aplicaciones en el lenguaje que prefieran y seguir sacando todo el provecho del motor en tiempo de ejecución, la biblioteca de clases y los componentes escritos en otros lenguajes por otros colegas. El proveedor de un compilador puede elegir destinarlo al motor en tiempo de ejecución. Los compiladores de lenguajes que se destinan a .NET Framework hacen que las características de .NET Framework estén disponibles para el código existente escrito en dicho lenguaje, lo que facilita enormemente el proceso de migración de las aplicaciones existentes.

Aunque el motor en tiempo de ejecución está diseñado para el software del futuro, también es compatible con el software actual y el software antiguo. La interoperabilidad entre el código administrado y no administrado permite que los desarrolladores continúen utilizando los componentes COM y las DLL que necesiten.

El motor en tiempo de ejecución está diseñado para mejorar el rendimiento. Aunque Common Language Runtime proporciona muchos servicios estándar de motor en tiempo de ejecución, el código administrado nunca se interpreta. Una característica denominada compilación JIT (Just-In-Time) permite ejecutar todo el código administrado en el lenguaje máquina nativo del sistema en el que se ejecuta. Mientras tanto, el administrador de memoria evita que la memoria se pueda fragmentar y aumenta la zona de referencia de la memoria para mejorar aún más el rendimiento.

Por último, el motor en tiempo de ejecución se puede alojar en aplicaciones de servidor de gran rendimiento, como Microsoft® SQL Server™ e Internet Information Services (IIS). Esta infraestructura permite utilizar código administrado para escribir lógica empresarial, al tiempo que se disfruta del superior rendimiento de los mejores servidores empresariales del sector que puedan alojar el motor en tiempo de ejecución.

⊟ Biblioteca de clases de .NET Framework

La biblioteca de clases de .NET Framework es una colección de tipos reutilizables que se integran estrechamente con Common Language Runtime. La biblioteca de clases está orientada a objetos, lo que proporciona tipos de los que su propio código administrado puede derivar funciones. Esto ocasiona que los tipos de .NET Framework sean sencillos de utilizar y reduce el tiempo asociado con el aprendizaje de las nuevas características de .NET Framework. Además, los componentes de terceros se pueden integrar sin dificultades con las clases de .NET Framework.

Por ejemplo, las clases de colección de .NET Framework implementan un conjunto de interfaces que puede usar para desarrollar sus propias clases de colección. Éstas se combinarán fácilmente con las clases de .NET Framework.

Como en cualquier biblioteca de clases orientada a objetos, los tipos de .NET Framework permiten realizar diversas tareas de programación comunes, como son la administración de cadenas, recolección de datos, conectividad de bases de datos y acceso a archivos. Además de estas tareas habituales, la biblioteca de clases incluye tipos adecuados para diversos escenarios de desarrollo especializados. Por ejemplo, puede utilizar .NET Framework para desarrollar los siguientes tipos de aplicaciones y servicios:

- Aplicaciones de consola.
- Aplicaciones GUI de Windows (Windows Forms).
- Aplicaciones de Windows Presentation Foundation (WPF).
- Aplicaciones de ASP.NET.
- Servicios Web.
- Servicios de Windows.
- Aplicaciones orientadas a servicios utilizando Windows Communication Foundation (WCF).
- Aplicaciones habilitadas para el flujo de trabajo utilizando Windows Workflow Foundation (WF).

Por ejemplo, las clases de Windows Forms son un conjunto completo de tipos reutilizables que simplifican enormemente el desarrollo de interfaces GUI para Windows. Si escribe una aplicación Web Form de ASP.NET, puede utilizar las clases de formularios Web Forms.

Arquitectura de .NET Framework 3.5

La versión 3.5 de .NET Framework se basa en las versiones 2.0 y 3.0 de .NET Framework, incluidos los Service Pack de estas versiones. En este tema se describe brevemente la relación que existe entre las versiones 2.0, 3.0 y 3.5 de .NET Framework.

⊟ Relación entre las versiones 2.0, 3.0 y 3.5 de .NET Framework

Los componentes que se enumeran a continuación se consideran parte de .NET Framework 3.5:

- .NET Framework 2.0
- Service Pack 1 de .NET Framework 2.0, que actualiza los ensamblados incluidos en .NET Framework 2.0.
- .NET Framework 3.0, que utiliza los ensamblados de .NET Framework 2.0 o .NET Framework 2.0 SP1 (si está instalado), e incluye los ensamblados necesarios para las tecnologías introducidas en .NET Framework 3.0. Por ejemplo, PresentationFramework.dll y PresentationCore.dll, que son necesarios para Windows Presentation Foundation (WPF), se instalan con .NET Framework 3.0.
- .Service Pack 1 de NET Framework 3.0, que actualiza los ensamblados introducidos en .NET Framework 3.0.
- Nuevos ensamblados que proporcionan una funcionalidad adicional a .NET Framework 2.0 y 3.0 y las tecnologías nuevas de .NET Framework 3.5.

Si alguno de estos componentes no se encuentra en el equipo al instalar .NET Framework 3.5, se instalará automáticamente.

Una aplicación utiliza los mismos ensamblados con independencia de si tiene como destino .NET Framework 2.0, 3.0 o 3.5. Por ejemplo, una aplicación que utilice WPF y tenga como destino .NET Framework 3.0, usará la misma instancia del ensamblado mscorlib que una aplicación que utilice formulariosWindows Forms y tenga como destino .NET Framework 2.0. Si el Service Pack 1 de .NET Framework 2.0 está instalado en el equipo, mscorlib.dll se habrá actualizado, y las dos aplicaciones utilizarán la versión actualizada de mscorlib.dll.

Nota:

La relación entre las versiones 2.0, 3.0 y 3.5 de .NET Framework es diferente a la relación que existe entre las versiones 1.0, 1.1 y 2.0 de .NET Framework, que son totalmente independientes unas de otras, por lo que una versión puede estar en un equipo con independencia de si las otras versiones se encuentran o no en dicho equipo. Cuando las versiones 1.0, 1.1 y 2.0 están en el mismo equipo, cada versión tiene su propio Common Language Runtime, sus propias bibliotecas de clases, su propio compilador, etc. Las aplicaciones pueden elegir si van a utilizar como destino la versión 1.0, 1.1 o 2.0.

⊟ Características incluidas en .NET Framework 3.5

En esta sección se resumen las tecnologías de .NET Framework 2.0, .NET Framework 3.0 y .NET Framework 3.5. Esta lista no es exhaustiva, sólo incluye las principales tecnologías de .NET Framework.

.NET Framework 2.0

Las tecnologías siguientes se incluyen en .NET Framework 2.0.

- Common Language Runtime (CLR).
- Compatibilidad con los tipos y métodos genéricos.
- Compiladores para C#, Visual Basic, C++ y J#.
- Bibliotecas de clases base.
- ADO.NET.
- ASP.NET.
- Formularios Windows Forms.
- Servicios web.

NET Framework 2.0 SP 1

El Service Pack 1 de .NET Framework 2.0 actualiza el CLR y varios de los ensamblados que se incluyen en .NET Framework 2.0, y se puede instalar con independencia de .NET Framework 3.5. La mayoría de las actualizaciones de .NET Framework 2.0 no constituyen cambios importantes, aunque hay algunos casos en que se agregan nuevos elementos de API o se modifica el comportamiento. Si su aplicación se basa en funcionalidades nuevas o modificadas, le recomendamos que cambie el destino de su aplicación a .NET Framework 3.5. Si su aplicación se basa en cambios que se incluyeron en .NET Framework 2.0 SP1, puede mantener .NET Framework 2.0 como destino de la aplicación y pedirle a sus clientes que descarguen .NET Framework 2.0 SP1.

.NET Framework 3,0

.NET Framework 3.0 necesita que .NET Framework 2.0 esté instalado en el equipo. Si un usuario instala .NET Framework 3.0 en un equipo que no tiene .NET Framework 2.0, se instalará automáticamente.

Las tecnologías siguientes se introducen en .NET Framework 3.0:

- Windows Presentation Foundation (WPF).
- Windows Communications Foundation (WCF).
- Windows Workflow Foundation (WF).

NET Framework 3.0 SP 1

El Service Pack 1 de .NET Framework 3.0 actualiza varios ensamblados incluidos en .NET Framework 3.0 y se puede instalar con independencia de .NET Framework 3.5. Estas actualizaciones incorporan cambios intrascendentes, nuevos elementos de API y funciones adicionales a las tecnologías de .NET Framework 3.0. Si su aplicación se basa en una funcionalidad nueva, le recomendamos que utilice como destino de la aplicación .NET Framework 3.5. Si se basa en los cambios que se incluyeron en .NET Framework 3.0 SP1, puede mantener .NET Framework 3.0 como destino de la aplicación y pedirle a sus clientes que descarguen .NET Framework 3.0 SP1.

Cuando se instala .NET Framework 3.0 SP1, se instala .NET Framework 2.0 SP1 si aún no se encuentra en el equipo.

.NET Framework 3.5

.NET Framework 3.5 introduce nuevas características para las tecnologías de las versiones 2.0 y 3.0 e incorpora tecnologías adicionales en forma de nuevos ensamblados. Las tecnologías siguientes se introducen en .NET Framework 3.5:

- LINQ.
- Nuevos compiladores para C#, Visual Basic y C++.
- ASP.NET AJAX.
- Tipos adicionales de la biblioteca de clases base.

Common Language Runtime (CLR)

.NET Framework proporciona un entorno en tiempo de ejecución denominado Common Language Runtime, que ejecuta el código y proporciona servicios que facilitan el proceso de desarrollo.

Información General acerca de Common Language Runtime

Los compiladores y las herramientas exponen la funcionalidad en tiempo de ejecución y permiten escribir código con las ventajas que proporciona este entorno de ejecución administrado. El código desarrollado con un compilador de lenguaje orientado al tiempo de ejecución se denomina código administrado. Este código se beneficia de características como: la integración entre lenguajes, el control de excepciones entre lenguajes, la seguridad mejorada, la compatibilidad con la implementación y las versiones, un modelo simplificado de interacción y servicios de creación de perfiles y depuración.

Para permitir al motor en tiempo de ejecución proporcionar servicios al código administrado, los compiladores de lenguajes deben emitir metadatos que describen los tipos, los miembros y las referencias del código. Los metadatos se almacenan con el código; cada archivo ejecutable portable (PE) de Common Language Runtime cargable contiene metadatos. El motor en tiempo de ejecución utiliza los metadatos para localizar y cargar clases, colocar instancias en memoria, resolver invocaciones a métodos, generar código nativo, exigir mecanismos de seguridad y establecer los límites del contexto en tiempo de ejecución.

El tiempo de ejecución controla automáticamente la disposición de los objetos y administra las referencias a éstos, liberándolos cuando ya no se utilizan. Los objetos cuya duración se administra de esta forma se denominan datos administrados. La recolección de elementos no utilizados elimina pérdidas de memoria así como otros errores habituales de programación. Con un código administrado se pueden utilizar datos administrados, datos no administrados o estos dos tipos de datos en una aplicación .NET. Framework. Como los compiladores de lenguajes proporcionan sus propios tipos, como tipos primitivos, no siempre se sabe (o no es necesario saber) si los datos se están administrando.

Common Language Runtime facilita el diseño de los componentes y de las aplicaciones cuyos objetos interactúan entre lenguajes distintos. Los objetos escritos en lenguajes diferentes pueden comunicarse entre sí, lo que permite integrar sus comportamientos de forma precisa. Por ejemplo, puede definir una clase y, a continuación, utilizar un lenguaje diferente para derivar una clase de la clase original o llamar a un método de la clase original. También se puede pasar al método de una clase una instancia de una clase escrita en un lenguaje diferente. Esta integración entre lenguajes diferentes es posible porque los compiladores y las herramientas de lenguajes orientados al motor en tiempo de ejecución utilizan un sistema de tipos común definido por el motor en tiempo de ejecución, y los lenguajes siguen las reglas en tiempo de ejecución para definir nuevos tipos, así como para crear, utilizar, almacenar y enlazar tipos.

Como parte de los metadatos, todos los componentes administrados contienen información sobre los componentes y los recursos utilizados en su creación. El motor en tiempo de ejecución utiliza esta información para garantizar que el componente o la aplicación contiene las versiones especificadas de todo lo necesario, por lo que hay menos posibilidades de que la ejecución del código se interrumpa debido a una dependencia inadecuada. La información de registro y los datos de estado ya no se almacenan en el Registro, donde puede ser difícil establecer y mantener datos. En su lugar, la información sobre tipos definidos por el usuario (y sus dependencias) se almacena con el código como

metadatos y, de este modo, las tareas de réplica y eliminación de componentes es mucho menos complicada.

Las herramientas y los compiladores de lenguajes exponen la funcionalidad del motor en tiempo de ejecución de forma que resulte útil e intuitiva para los programadores. Esto significa que algunas características en tiempo de ejecución pueden ser más evidentes en un entorno que en otro. El funcionamiento del motor en tiempo de ejecución depende de las herramientas y los compiladores utilizados. Por ejemplo, un programador de Visual Basic observará que con Common Language Runtime, el lenguaje Visual Basic contiene más características orientadas a objetos que antes. Algunas de las ventajas del motor en tiempo de ejecución son:

- Mejoras en el rendimiento.
- Capacidad para utilizar fácilmente componentes desarrollados en otros lenguajes.
- Tipos extensibles que proporciona una biblioteca de clases
- Nuevas características del lenguaje como herencia, interfaces y sobrecarga para la programación orientada a objetos; compatibilidad con el uso de subprocesos libres que permite la creación de multiprocesos; aplicaciones escalables; compatibilidad con los atributos personalizados y el control de excepciones estructurado.

Si utiliza Microsoft® Visual C++® .NET, puede escribir código administrado utilizando Visual C++, que proporcionan las ventajas de un entorno de ejecución administrado, así como el acceso a características eficaces y a tipos de datos informativos que ya le resultan familiares. Otras características del motor en tiempo de ejecución son:

- Integración entre lenguajes diferentes y, en especial, herencia entre lenguajes.
- Recolección de elementos no utilizados, que administra la duración de los objetos de modo que no es necesario el recuento de referencias.
- Objetos autodescriptivos que hacen innecesario el Lenguaje de definición de interfaces (IDL).
- Capacidad para compilar una vez y ejecutar código en cualquier CPU y sistema operativo que sea compatible con el motor en tiempo de ejecución.

También se puede escribir código administrado en lenguaje C#, que proporciona las siguientes ventajas:

- Diseño completo orientado a objetos
- Seguridad de tipos muy sólida
- Buena combinación entre la simplicidad de Visual Basic y la eficacia de C++.
- Recolección de elementos no utilizados.
- Sintaxis y palabras clave similares en C y en C++.
- Utilice delegados para una mayor seguridad y protección de tipos, en vez de punteros a funciones. Los punteros a funciones están disponibles mediante la utilización de la palabra clave

unsafe de C# y de la opción **/unsafe** del compilador de C# (Csc.exe) para datos y código no administrados.

Proceso de Ejecución Administrada

El proceso de ejecución administrada incluye los pasos siguientes:

1. Elegir un compilador.

   Para obtener los beneficios que proporciona Common Language Runtime, se deben utilizar uno o más compiladores de lenguaje orientados al tiempo de ejecución.

2. Compilar el código a Lenguaje intermedio de Microsoft (MSIL).

   La compilación convierte el código fuente en MSIL y genera los metadatos requeridos.

3. Compilar MSIL a código nativo.

   En tiempo de ejecución, un compilador Just-In-Time (JIT) convierte MSIL en código nativo. Durante esta compilación, el código debe pasar un proceso de comprobación que examina el MSIL y los metadatos para averiguar si el código garantiza la seguridad de tipos.

4. Ejecutar código.

   Common Language Runtime proporciona la infraestructura que permite que la ejecución tenga lugar, así como una amplia gama de servicios que se pueden utilizar durante la ejecución.

Administración de Memoria Automática

La administración de memoria automática es uno de los servicios que proporciona Common Language Runtime durante la ejecución administrada. El recolector de elementos no utilizados de Common Language Runtime administra la asignación y liberación de la memoria de una aplicación. Esto significa que los programadores no tienen que escribir código para realizar tareas de administración de memoria al programar aplicaciones administradas. La administración automática de la memoria puede eliminar problemas frecuentes, como olvidar liberar un objeto y causar una pérdida de memoria, o intentar tener acceso a la memoria de un objeto que ya se ha liberado. En esta sección se describe cómo asigna y libera memoria el recolector de elementos no utilizados.

⊟ Asignar memoria

Cuando se inicializa un nuevo proceso, el motor en tiempo de ejecución reserva una región contigua de espacio de direcciones para el proceso. Este espacio de direcciones reservado se denomina montón administrado. El montón administrado mantiene un puntero a la dirección a la que se asignará el siguiente objeto del montón. Inicialmente, este puntero se establece en la dirección base del montón administrado. Todos los tipos de referencia se asignan en el montón administrado. Cuando una aplicación crea el primer tipo de referencia, se le asigna memoria en la dirección base del montón administrado. Cuando la aplicación crea el siguiente objeto, el recolector de elementos no utilizados le asigna memoria en el espacio de direcciones que sigue inmediatamente al primer objeto. Siempre que haya espacio de

direcciones disponible, el recolector de elementos no utilizados continúa asignando espacio a los objetos nuevos de este modo.

La asignación de memoria desde el montón administrado es más rápida que la asignación de memoria no administrada. Como el tiempo de ejecución asigna memoria a los objetos agregando un valor a un puntero, este método es casi tan rápido como la asignación de memoria desde la pila. Además, puesto que los nuevos objetos que se asignan consecutivamente se almacenan uno junto a otro en el montón administrado, la aplicación puede tener un acceso muy rápido a los objetos.

⊟ Liberar memoria

El motor de optimización del recolector de elementos no utilizados determina cuál es el mejor momento para realizar una recolección basándose en las asignaciones realizadas. Cuando el recolector lleva a cabo una recolección, libera la memoria de los objetos que ya no usa la aplicación. Determina qué objetos ya no se usan examinando las raíces de la aplicación. Todas las aplicaciones tienen un conjunto de raíces. Cada raíz hace referencia a un objeto del montón administrado, o bien se establece en null. Las raíces de una aplicación incluyen punteros de objeto globales y estáticos, variables locales y parámetros de objetos de referencia en la pila de un subproceso, y registros de la CPU. El recolector de elementos no utilizados tiene acceso a la lista de raíces activas que mantienen el compilador Just-In-Time (JIT) y el motor en tiempo de ejecución. Con esta lista examina las raíces de la aplicación y, durante este proceso, crea un gráfico que contiene todos los objetos que no se pueden alcanzar desde las raíces.

Los objetos que no están en el gráfico no se pueden alcanzar desde las raíces de la aplicación. El recolector considera los objetos inalcanzables elementos no utilizados y libera la memoria que tienen asignada. Durante una recolección, el recolector de elementos no utilizados examina el montón administrado y busca los bloques de espacio de direcciones que ocupan los objetos que no se pueden alcanzar. Cuando encuentra cada uno de los objetos inalcanzables, usa una función de copia de memoria para compactar los objetos alcanzables en la memoria y libera los bloques de espacios de direcciones asignados a los objetos no alcanzables. Una vez que se ha compactado la memoria de los objetos alcanzables, el recolector de elementos no utilizados hace las correcciones de puntero necesarias para que las raíces de la aplicación señalen a los objetos en sus nuevas ubicaciones. También sitúa el puntero del montón administrado después del último objeto alcanzable. Tenga en cuenta que la memoria sólo se compacta si, durante una recolección, se encuentra un número significativo de objetos inalcanzables. Si todos los objetos del montón administrado sobreviven a una recolección, no hay necesidad de comprimir la memoria.

Para mejorar el rendimiento, el tiempo de ejecución asigna memoria a los objetos grandes en un montón aparte. El recolector de elementos no utilizados libera la memoria para los objetos grandes automáticamente. Sin embargo, para no mover objetos grandes en la memoria, dicha memoria no se compacta.

⊟ Generaciones y rendimiento

Para optimizar el rendimiento del recolector de elementos no utilizados, el montón administrado se divide en tres generaciones: 0, 1 y 2. El algoritmo de recolección de elementos no utilizados del motor en tiempo de ejecución se basa en varias afirmaciones que la industria de software informático ha comprobado como ciertas experimentando con esquemas de recolección de elementos no utilizados. Primero, es más rápido compactar la memoria de una parte del montón administrado que la de todo el montón. En segundo lugar, los objetos más recientes tienen una duración más corta y los objetos

antiguos tienen una duración más larga. Por último, los objetos más recientes suelen estar relacionados unos con otros y la aplicación tiene acceso a ellos más o menos al mismo tiempo.

El recolector de elementos no utilizados del motor en tiempo de ejecución guarda los nuevos objetos en la generación 0. Los objetos creados en las primeras etapas de la duración de la aplicación y que sobreviven a las recolecciones se promueven y se almacenan en las generaciones 1 y 2.El proceso de promoción de objetos se describe más adelante en este tema. Como es más rápido compactar una parte del montón administrado que todo el montón, este esquema permite que el recolector de elementos no utilizados libere la memoria en una generación específica en lugar de liberarla para todo el montón administrado cada vez que realiza una recolección.

En realidad, el recolector de elementos no utilizados realiza una recolección cuando se llena la generación 0. Si una aplicación trata de crear un nuevo objeto cuando la generación 0 está llena, el recolector de elementos no utilizados descubre que no queda espacio de direcciones en la generación 0 para asignárselo. El recolector de elementos no utilizados realiza una recolección, en un intento de liberar espacio de direcciones para el objeto en la generación 0. Primero examina los objetos de la generación 0 y no todos los objetos del montón administrado. Éste es un enfoque más eficaz, ya que los objetos nuevos suelen tener una duración más corta y se espera que la aplicación no utilice muchos de los objetos de la generación 0 cuando se realice una recolección. Además, una recolección de tan sólo la generación 0 a menudo recupera suficiente memoria para que la aplicación pueda continuar creando nuevos objetos.

Una vez que el recolector de elementos no utilizados realiza una recolección de la generación 0, compacta la memoria de los objetos que se pueden alcanzar como se ha explicado antes en este tema, en Liberar memoria. A continuación, el recolector de elementos no utilizados promueve estos objetos y considera que esta parte del montón administrado está en la generación 1. Puesto que los objetos que sobreviven a las recolecciones suelen tener una duración más larga, es lógico promoverlos a una generación superior. En consecuencia, el recolector de elementos no utilizados no tiene que volver a examinar los objetos de las generaciones 1 y 2 cada vez que realiza una recolección en la generación 0.

Una vez que el recolector de elementos no utilizados ha realizado la primera recolección de la generación 0 y ha promovido los objetos que se pueden alcanzar a la generación 1, considera lo que queda del montón administrado como generación 0. Continúa asignando memoria a los nuevos objetos en la generación 0 hasta que está llena y es necesario realizar otra recolección. En este momento, el motor de optimización del recolector de elementos no utilizados determina si es necesario examinar los objetos de generaciones más antiguas. Por ejemplo, si una recolección de la generación 0 no recupera memoria suficiente para que la aplicación pueda terminar satisfactoriamente su intento de crear un nuevo objeto, el recolector de elementos no utilizados puede realizar una recolección de la generación 1 y, después, de la generación 0. Si así no se recupera suficiente memoria, el recolector de elementos no utilizados puede realizar una recolección de las generaciones 2, 1 y 0. Después de cada recolección, el recolector de elementos no utilizados compacta los objetos alcanzables de la generación 0 y los promueve a la generación 1. Los objetos de la generación 1 que se siguen utilizando después de la recolección se promueven a la generación 2. Como el recolector de elementos no utilizados sólo admite tres generaciones, los objetos de la generación 2 que se siguen utilizando después de una recolección permanecen en la generación 2 hasta que se determina, en una recolección posterior, que no se pueden alcanzar.

⊟ Liberar memoria para recursos no administrados

En el caso de la mayoría de los objetos creados por la aplicación, puede utilizar el recolector de elementos no utilizados para realizar automáticamente las tareas de administración de memoria. Sin embargo, los recursos no administrados requieren una limpieza explícita. El tipo más habitual de recurso no administrado es un objeto que contiene un recurso del sistema operativo, como un identificador de archivo, identificador de ventana o conexión de red. Aunque el recolector de elementos no utilizados puede realizar el seguimiento del período de duración de un objeto administrado que encapsula un recurso no administrado, no tiene un conocimiento específico de cómo limpiar el recurso. Cuando se crea un objeto que encapsula un recurso no administrado, es recomendable proporcionar el código necesario para limpiar dicho recurso en un método público **Dispose**. Si se proporciona un método **Dispose**, se permite que los usuarios del objeto liberen su memoria de manera explícita cuando hayan terminado de usarlo. Si se utiliza un objeto que encapsula un recurso no administrado, se debe conocer la existencia de **Dispose** y llamarlo cuando sea necesario.

Sistema de Tipos Comunes (CTS)

El sistema de tipos común define cómo se declaran, utilizan y administran los tipos en el motor en tiempo de ejecución. Es también una parte importante de la compatibilidad en tiempo de ejecución con la integración entre lenguajes. El sistema de tipos común realiza las funciones siguientes:

- Establece un marco de trabajo que ayuda a permitir la integración entre lenguajes, la seguridad de tipos y la ejecución de código con alto rendimiento.
- Proporciona un modelo orientado a objetos que admite la implementación completa de muchos lenguajes de programación.
- Define reglas que deben seguir los lenguajes, lo que ayuda a garantizar que los objetos escritos en distintos lenguajes puedan interactuar unos con otros.

Información General acerca del Sistema de Tipos Común

En esta sección se describen los conceptos y se definen los términos que le ayudarán a comprender la implementación del sistema de tipos común de su lenguaje y a trabajar con él.

⊟ Clasificación de tipos

El sistema de tipos común es compatible con dos categorías generales de tipos, que a su vez se dividen en subcategorías:

- Tipos de valor

  Los tipos de valor contienen directamente sus datos y las instancias de los tipos de valor se asignan en la pila o se asignan en línea en una estructura. Los tipos de valor pueden ser integrados (implementados por el motor en tiempo de ejecución), definidos por el usuario o enumeraciones.

- Tipos de referencia

  Los tipos de referencia guardan una referencia a la dirección en memoria del valor y se asignan en el montón. Los tipos de referencia pueden ser tipos autodescriptivos, de puntero o de interfaz. El tipo de un tipo de referencia se puede determinar a partir de los valores de los tipos autodescriptivos. Los tipos autodescriptivos se dividen en matrices y tipos de clase. Los tipos de clase son clases definidas por el usuario, tipos de valor a los que se ha aplicado la conversión boxing y delegados.

Las variables que son tipos de valor tienen, cada una, su propia copia de los datos y, por lo tanto, las operaciones en una variable no afectan a las demás. Las variables que son tipos de referencia pueden hacer referencia al mismo objeto y, por lo tanto, las operaciones en una variable pueden afectar al mismo objeto al que hace referencia otra variable.

Todos los tipos derivan del tipo base System.::.Object.

En el siguiente ejemplo se muestra la diferencia entre los tipos de referencia y los tipos de valor.

Visual Basic

```
Class Class1
    Public Value As Integer = 0
End Class 'Class1

Class Test

    Shared Sub Main()
        Dim val1 As Integer = 0
        Dim val2 As Integer = val1
        val2 = 123
        Dim ref1 As New Class1()
        Dim ref2 As Class1 = ref1
        ref2.Value = 123
        Console.WriteLine("Values: {0}, {1}", val1, val2)
        Console.WriteLine("Refs: {0}, {1}", ref1.Value, ref2.Value)
    End Sub 'Main
End Class 'Test
```

C#

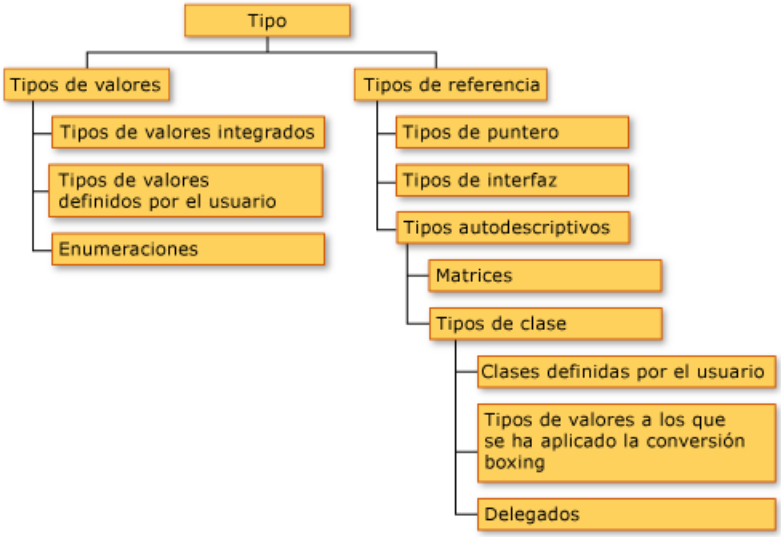
```
using System;
class Class1
{
public int Value = 0;
}
class Test
{
    static void Main() {
        int val1 = 0;
        int val2 = val1;
        val2 = 123;
        Class1 ref1 = new Class1();
        Class1 ref2 = ref1;
        ref2.Value = 123;
        Console.WriteLine("Values: {0}, {1}", val1, val2);
        Console.WriteLine("Refs: {0}, {1}", ref1.Value, ref2.Value);
    }
}
```

Los resultados de este programa son los siguientes:

```
Values: 0, 123
Refs: 123, 123
```

El diagrama siguiente ilustra cómo se relacionan estos distintos tipos. Tenga en cuenta que las instancias de los tipos pueden ser simplemente tipos de valor o tipos autodescriptivos, aunque haya subcategorías de estos tipos.

Clasificación de tipos



⊟ Valores y objetos

Los valores son representaciones binarias de datos y los tipos proporcionan una forma de interpretar estos datos. Un tipo de valor se almacena directamente como una representación binaria de los datos del tipo. El valor de un tipo de referencia es la ubicación de la secuencia de bits que representa los datos del tipo.

Cada valor tiene un tipo exacto que define por completo la representación del valor y las operaciones definidas en el valor. Los valores de los tipos autodescriptivos se denominan objetos. Si bien siempre se puede determinar el tipo exacto de un objeto examinando su valor, ello no se puede hacer con tipo de valor o un tipo de puntero. Un valor puede tener más de un tipo. El valor de un tipo que implementa una interfaz es también un valor de ese tipo de interfaz. De la misma manera, el valor de un tipo derivado de un tipo base es también un valor de ese tipo base.

⊟ Tipos y ensamblados

El motor de tiempo en ejecución utiliza ensamblados para ubicar y cargar tipos. El manifiesto del ensamblado contiene la información que el motor en tiempo de ejecución utiliza para resolver todas las referencias a tipos hechas dentro del ámbito del ensamblado.

Un nombre de tipo del motor en tiempo de ejecución tiene dos partes lógicas: el nombre del ensamblado y el nombre del tipo que se encuentra en el ensamblado. Dos tipos que tengan el mismo nombre pero estén en ensamblados distintos se definen como dos tipos diferentes.

Los ensamblados proporcionan coherencia entre el ámbito de los nombres que ve el programador y el que ve el sistema del motor en tiempo de ejecución. Los programadores escriben tipos en el contexto de un ensamblado. El contenido del ensamblado que está creando un programador establece el ámbito de los nombres que estarán disponibles en tiempo de ejecución.

Tipos y espacios de nombres

Desde el punto de vista del motor en tiempo de ejecución, un espacio de nombres no es más que una colección de nombres de tipos. Algunos lenguajes pueden tener construcciones y la sintaxis correspondiente que ayudan a los programadores a formar grupos lógicos de tipos, pero el motor en tiempo de ejecución no utiliza estas construcciones al enlazar tipos. Así, las clases **Object** y **String** forman parte del espacio de nombres **System**, pero el motor en tiempo de ejecución sólo reconoce los nombres completos de cada tipo, que son System.Object y System.String respectivamente.

Se puede generar un único ensamblado que exponga tipos que parezcan proceder de dos espacios de nombres jerárquicos distintos, como System.Collections y System.Windows.Forms. También se pueden crear dos ensamblados que exporten tipos cuyos nombres contengan MyDll.MyClass.

Si se crea una herramienta para representar los tipos de un ensamblado como pertenecientes a un espacio de nombres jerárquico, la herramienta debe enumerar los tipos de un ensamblado o grupo de ensamblados y analizar los nombres de tipo para generar una relación jerárquica.

Definiciones de Tipos

Los tipos nuevos se definen a partir de tipos existentes. Los tipos de valor, punteros, matrices y delegados integrados se definen cuando se utilizan y se conocen como tipos implícitos. Los tipos pueden estar anidados, es decir, un tipo puede ser miembro de otro.

Una definición de tipo contiene:

- Los atributos definidos en el tipo.
- La visibilidad del tipo.
- El nombre del tipo.
- El tipo base del tipo.
- Las interfaces que implementa el tipo.
- Las definiciones de todos los miembros del tipo

Atributos

Los atributos proporcionan metadatos adicionales definidos por el usuario . Los atributos se pueden aplicar prácticamente a todos los elementos del lenguaje: tipos, propiedades, métodos, etc.

Accesibilidad a tipos

Todos los tipos tienen un modificador de accesibilidad que rige su accesibilidad desde otros tipos. En la tabla siguiente se describen las accesibilidades a tipos que admite el motor en tiempo de ejecución.

Accesibilidad	Descripción
public	Todos los ensamblados pueden tener acceso al tipo.
assembly	El tipo sólo está accesible desde dentro del ensamblado.

La accesibilidad de un tipo anidado depende de su dominio de accesibilidad, que viene determinado por la accesibilidad declarada del miembro y el dominio de accesibilidad del tipo contenedor inmediato. Sin embargo, el dominio de accesibilidad de un tipo anidado no puede exceder al del tipo contenedor.

El dominio de accesibilidad del miembro anidado **M** declarado en el tipo **T** en el programa **P** se define de la siguiente manera (teniendo en cuenta que **M** puede ser un tipo en sí mismo):

- Si la accesibilidad declarada de **M** es **public**, el dominio de accesibilidad de **M** es el dominio de accesibilidad de **T**.
- Si la accesibilidad declarada de **M** es **protected internal**, el dominio de accesibilidad de **M** es la intersección del dominio de accesibilidad de **T** con el texto de programa de **P** y el texto de programa de cualquier tipo derivado de **T** declarado fuera de **P**.
- Si la accesibilidad declarada de **M** es **protected**, el dominio de accesibilidad de **M** es la intersección del dominio de accesibilidad de **T** con el texto de programa de **T** y el de cualquier tipo derivado de **T**.
- Si la accesibilidad declarada de **M** es **internal**, el dominio de accesibilidad de **M** es la intersección del dominio de accesibilidad de **T** con el texto de programa de **P**.
- Si la accesibilidad declarada de **M** es **private**, el dominio de accesibilidad de **M** es el texto de programa de **T**.

⊟ Nombres de tipo

El sistema de tipos común sólo impone dos restricciones en los nombres:

1. Todos los nombres se codifican como cadenas de caracteres Unicode (de 16 bits).
2. Los nombres no pueden tener un valor incrustado (de 16 bits) de 0x0000.

Todas las comparaciones se hacen byte a byte y, por lo tanto, distinguen entre mayúsculas y minúsculas y son independientes de la configuración regional.

Aunque un tipo puede hacer referencia a tipos de otros módulos y ensamblados, un tipo se define completamente dentro de un módulo. Los nombres de tipo sólo tienen que ser únicos en un ensamblado. Para identificar un tipo por completo, el nombre del tipo debe estar calificado por el ensamblado que contiene la implementación del tipo.

⊟ Tipos base e interfaces

Un tipo puede heredar valores y comportamientos de otro. El sistema de tipos común no permite que los tipos hereden de más de un tipo base.

Un tipo puede implementar cualquier número de interfaces. Para implementar una interfaz, un tipo debe implementar todos los miembros virtuales de la interfaz. Un tipo derivado puede implementar un método virtual, que se puede invocar estática o dinámicamente.

⊟ Miembros de tipos

Los miembros de un tipo (eventos, campos, tipos anidados, métodos y propiedades) definen su comportamiento y estado.

Miembros de Tipos

El motor en tiempo de ejecución permite definir los miembros de un tipo: eventos, campos, tipos anidados, métodos y propiedades. Cada miembro tiene una firma. En la tabla siguiente se describen los miembros de tipos utilizados en .NET Framework.

Miembro	Descripción
Evento	Define un incidente al que se puede responder, así como métodos para suscribirse a un evento y anular la suscripción y para provocar el evento. Los eventos se usan con frecuencia para informar a otros tipos de cambios de estado.
Campo	Describe y contiene parte del estado del tipo. Los campos pueden ser de cualquier tipo que admita el motor en tiempo de ejecución.
Tipo anidado	Define un tipo dentro del ámbito del tipo envolvente.
Método	Describe las operaciones que se pueden hacer en el tipo. La firma de un método especifica los tipos permitidos de todos sus argumentos y de su valor devuelto. Un constructor es una clase de método especial que crea nuevas instancias de un tipo.
Propiedad	Identifica un valor o estado del tipo y define métodos para obtener o establecer el valor de la propiedad. Las propiedades pueden ser tipos primitivos, colecciones de tipos primitivos, tipos definidos por el usuario o colecciones de tipos definidos por el usuario. Las propiedades se usan a menudo para que la interfaz pública de un tipo se mantenga independiente de la representación real del tipo.

⊟ Características de los miembros

El sistema de tipos común permite que los tipos tengan distintas características, pero no es necesario que los lenguajes admitan todas estas características. En la siguiente tabla se describen estas características de los miembros.

Característica	Se puede aplicar a	Descripción
abstract	Métodos, propiedades y eventos	El tipo no proporciona la implementación del método. Los tipos que heredan métodos abstractos y tipos que implementan interfaces con métodos abstractos deben proporcionar una implementación para el método. La única excepción es que el tipo derivado sea un tipo abstracto. Todos lo métodos abstractos son virtuales.
private, family, assembly, family y assembly, family o assembly, o public	All	Define la accesibilidad del miembro: private Sólo es accesible desde dentro del mismo tipo como miembro o dentro de un tipo anidado. family Accesible desde dentro del mismo tipo como miembro y desde tipos derivados que heredan de él. assembly Accesible sólo en el ensamblado en que está definido el tipo. family y assembly Accesible sólo desde los tipos que tengan derecho al acceso de familia y ensamblado. family o assembly

		Accesible sólo desde los tipos que tengan derecho al acceso de familia o ensamblado. public Accesible desde cualquier tipo.
final	Métodos, propiedades y eventos	El método virtual no puede ser reemplazado en un tipo derivado.
initialize-only	Campos	El valor sólo se puede inicializar y no se puede escribir en él después de la inicialización.
instance	Campos, métodos, propiedades y eventos	Si un miembro no está marcado como **static** (C# y C++), **Shared** (Visual Basic), **virtual** (C# y C++) u **Overridable** (Visual Basic), es un miembro de instancia (no hay palabra clave de la instancia). En la memoria habrá tantas copias de estos miembros como objetos que los utilicen.
literal	Campos	El valor asignado al campo es un valor fijo, conocido en tiempo de compilación, de un tipo de valor integrado. Los campos literales, a veces, se conocen como constantes.
newslot u override	All	Define cómo interactúa el miembro con los miembros heredados con la misma firma: newslot Oculta los miembros heredados con la misma firma. override Reemplaza la definición de un método virtual heredado. El valor predeterminado es newslot.
static	Campos, métodos, propiedades y eventos	El miembro pertenece al tipo en que está definido, no a una instancia particular del tipo. El miembro existe incluso si no se ha creado ninguna instancia del tipo y lo comparten todas las instancias del tipo.
virtual	Métodos, propiedades y eventos	Un tipo derivado puede implementar el método, que se puede invocar estática o dinámicamente. Si se utiliza la invocación dinámica, el tipo de la instancia que hace la llamada en tiempo de ejecución determina a qué implementación del método se llama, en lugar del tipo conocido en tiempo de compilación. Para invocar un método virtual de manera estática, es posible que haya que convertir la variable en un tipo que utilice la versión deseada del método.

⊟ Sobrecarga

Cada miembro de tipo tiene una firma única. Las firmas de método están formadas por el nombre del método y una lista de parámetros (el orden y los tipos de los argumentos del método). Se pueden definir varios métodos con el mismo nombre dentro un tipo, siempre que las firmas sean distintas. Cuando se definen dos o más métodos con el mismo nombre se dice que el método está sobrecargado. Por ejemplo, en **System.Char**, **IsDigit** está sobrecargado. Un método toma un **Char** y devuelve un valor **Boolean**. El otro método toma una **String** y un **Int32** y devuelve un valor **Boolean**. Las listas de parámetros también se pueden calificar con la restricción **varargs**, que indica que el método admite una lista de argumentos de variables.

⊟ Heredar, reemplazar y ocultar miembros

Un tipo derivado hereda todos los miembros de su tipo base, es decir, estos miembros se definen en el tipo derivado y están disponibles para él. El comportamiento o cualidades de los miembros heredados se puede modificar de dos maneras:

- Un tipo derivado puede ocultar un miembro heredado definiendo un nuevo miembro con la misma firma. Esto puede hacerse para que un miembro que era anteriormente pública se haga privado o para definir nuevo comportamiento para un método heredado marcado como **final**.
- Un tipo derivado puede reemplazar a un método virtual heredado. El método de reemplazo proporciona una nueva definición del método que se invocará según el tipo del valor en tiempo de ejecución y no el tipo de la variable conocido en tiempo de compilación. Un método puede reemplazar a un método virtual sólo si el método virtual no está marcado como **final** y el nuevo método es, al menos, tan accesible como el método virtual.

Tipos de Valor en el Sistema de Tipos Común

La mayor parte de los lenguajes de programación proporcionan tipos de datos integrados, como enteros y números de punto flotante, que se copian cuando se pasan como argumentos (es decir, los pasa el valor). En .NET Framework se denominan tipos de valor. El motor en tiempo de ejecución admite dos clases de tipos de valor:

- Tipos de valor integrados

  .NET Framework define tipos de valor integrados, como System.Int32 y System.Boolean, que corresponden y son idénticos a los tipos de datos primitivos utilizados por los lenguajes de programación.

- Tipos de valor definidos por el usuario

  El lenguaje proporcionará formas de definir sus propios tipos de valor, que se derivan de System.ValueType o System.Enum. Si desea definir un tipo que represente un valor pequeño, como un número complejo (mediante dos números de punto flotante) puede decidir definirlo como un tipo de valor, porque el tipo de valor se puede pasar eficazmente por valor. Si el tipo que va a definir se pasaría más eficazmente por referencia, entonces debe definirlo como clase.

Los tipos de valor se guardan con la misma eficacia que los tipos primitivos, pero en ellos se puede llamar a métodos, incluidos los métodos virtuales definidos en las clases System.Object y System.ValueType, además de todos los métodos definidos en el propio tipo de valor. Se pueden crear instancias de los tipos de valor, pasarlos como parámetros, guardarlos como variables locales o guardarlos en un campo de otro tipo de valor u objeto. Los tipos de valor no tienen la sobrecarga asociada al almacenamiento de una instancia de una clase y no requieren constructores.

Para cada tipo de valor, el motor en tiempo de ejecución proporciona un tipo al que se ha aplicado la conversión boxing, que es una clase que tiene el mismo estado y comportamiento que el tipo de valor. Algunos lenguajes requieren el uso de sintaxis especial cuando se necesita el tipo al que se haya aplicado la conversión boxing, otros utilizan el tipo automáticamente cuando es necesario. Cuando se define un

tipo de valor, se definen los dos tipos: al que se ha aplicado la conversión boxing y al que se ha aplicado la conversión unboxing.

Los tipos de valor pueden tener campos, propiedades y eventos. También pueden tener métodos estáticos y no estáticos. Cuando se les aplica la conversión boxing, heredan los métodos virtuales de **System.ValueType** y pueden implementar varias interfaces o ninguna.

Los tipos de valor están sellados, lo que quiere decir que de ellos no se puede derivar ningún otro tipo. Sin embargo, se pueden definir métodos virtuales directamente en el tipo de valor, a los que se puede llamar tanto en la forma del tipo al que se ha aplicado la conversión boxing como en la forma al que se ha aplicado la conversión unboxing. Aunque de un tipo de valor no se puede derivar otro tipo, en un tipo de valor se pueden definir métodos virtuales cuando se utiliza un lenguaje en el que es más cómodo trabajar con métodos virtuales que con métodos no virtuales o estáticos.

En el ejemplo siguiente se muestra cómo construir un tipo de valor para números complejos.

Visual Basic

```
Option Strict
Option Explicit

Imports System

' Value type definition for a complex number representation.
Public Structure Complex
    Public r, i As Double

    ' Constructor.
    Public Sub New(r As Double, i As Double)
        Me.r = r
        Me.i = i
    End Sub

    ' Returns one divided by the current value.
    Public ReadOnly Property Reciprocal() As Complex
        Get
            If r = 0.0 And i = 0.0 Then
                Throw New DivideByZeroException()
            End If
            Dim div As Double = r * r + i * i
            Return New Complex(r / div, -i / div)
        End Get
    End Property


    ' Conversion methods.
    Public Shared Function ToDouble(a As Complex) As Double
        Return a.r
    End Function
```

```
    Public Shared Function ToComplex(r As Double) As Complex
        Return New Complex(r, 0.0)
    End Function

    ' Basic unary methods.
    Public Shared Function ToPositive(a As Complex) As Complex
        Return a
    End Function

    Public Shared Function ToNegative(a As Complex) As Complex
        Return New Complex(-a.r, -a.i)
    End Function

    ' Basic binary methods for addition, subtraction, multiplication, and
division.
    Public Shared Function Add(a As Complex, b As Complex) As Complex
        Return New Complex(a.r + b.r, a.i + b.i)
    End Function

    Public Shared Function Subtract(a As Complex, b As Complex) As
Complex
        Return New Complex(a.r - b.r, a.i - b.i)
    End Function

    Public Shared Function Multiply(a As Complex, b As Complex) As
Complex
        Return New Complex(a.r * b.r - a.i * b.i, a.r * b.i + a.i * b.r)
    End Function

    Public Shared Function Divide(a As Complex, b As Complex) As Complex
        Return Multiply(a, b.Reciprocal)
    End Function

    ' Override the ToString method so the value appears in write
statements.
    Public Overrides Function ToString As String
        Return String.Format("({0}+{1}i)", r, i)
    End Function
End Structure

' Entry point.
Public Class ValueTypeSample

    Public Shared Sub Main()
        Dim a As New Complex(0, 1)
        Dim b As New Complex(0, - 2)
```

```
        Console.WriteLine()
        Console.WriteLine("a = " & a.ToString)
        Console.WriteLine("b = " & b.ToString)

        Console.WriteLine()
        Console.WriteLine("a + b = " & Complex.Add(a, b).ToString)
        Console.WriteLine("a - b = " & Complex.Subtract(a, b).ToString)
        Console.WriteLine("a * b = " & Complex.Multiply(a, b).ToString)
        Console.WriteLine("a / b = " & Complex.Divide(a, b).ToString)

        Console.WriteLine()
        Console.WriteLine("(double)a = " & Complex.ToDouble(a).ToString)
        Console.WriteLine("(Complex)5 = " &
Complex.ToComplex(5).ToString)
    End Sub
End Class
```

C#

```
using System;

// Value type definition for a complex number representation.
public struct Complex
{
    public double r, i;

    // Constructor.
    public Complex(double r, double i) { this.r = r; this.i = i; }

    // Returns one divided by the current value.
    public Complex Reciprocal
    {
        get
        {
            if (r == 0d && i == 0d)
                throw new DivideByZeroException();

            double div = r*r + i*i;
            return new Complex(r/div, -i/div);
        }
    }

    // Conversion operators.
    public static explicit operator double(Complex a)
    {
        return a.r;
    }
    public static implicit operator Complex(double r)
    {
```

```
        return new Complex(r,0d);
    }

    // Basic unary operators.
    public static Complex operator + (Complex a)
    {
        return a;
    }
    public static Complex operator - (Complex a)
    {
        return new Complex(-a.r, -a.i);
    }

    // Basic binary operators for addition, subtraction, multiplication,
and division.
    public static Complex operator + (Complex a, Complex b)
    {
        return new Complex(a.r + b.r, a.i + b.i);
    }
    public static Complex operator - (Complex a, Complex b)
    {
        return new Complex(a.r - b.r, a.i - b.i);
    }
    public static Complex operator * (Complex a, Complex b)
    {
        return new Complex(a.r*b.r - a.i*b.i, a.r*b.i + a.i*b.r);
    }
    public static Complex operator / (Complex a, Complex b)
    {
        return a * b.Reciprocal;
    }

    // Override the ToString method so the value appears in write
statements.
    public override string ToString() {
        return String.Format("({0}+{1}i)", r, i);
    }
}

// Entry point.
public class ValueTypeSample
{
    public static void Main()
    {
        Complex a = new Complex(0, 1);
        Complex b = new Complex(0, -2);

        Console.WriteLine();
```

```
        Console.WriteLine("a = " + a);
        Console.WriteLine("b = " + b);

        Console.WriteLine();
        Console.WriteLine("a + b = " + (a+b));
        Console.WriteLine("a - b = " + (a-b));
        Console.WriteLine("a * b = " + (a*b));
        Console.WriteLine("a / b = " + (a/b));

        Console.WriteLine();
        Console.WriteLine("(double)a = " + (double)a);
        Console.WriteLine("(Complex)5 = " + (Complex)5);
    }
}
```

Los resultados de este programa son los siguientes:

```
a = (0+1i)
b = (0+-2i)

a + b = (0+-1i)
a - b = (0+3i)
a * b = (2+0i)
a / b = (-0.5+0i)

(double)a = 0
(Complex)5 = (5+0i)
```

Clases del Sistema de Tipos Común

Si está familiarizado con la programación orientada a objetos, sabrá que una clase define las operaciones que puede realizar un objeto (métodos, eventos o propiedades) y define un valor que guarda el estado del objeto (campos). Aunque una clase, por lo general, incluye la definición y la implementación, puede tener uno o varios elementos que no tienen implementación.

Una instancia de una clase es un objeto. A la funcionalidad de un objeto se tiene acceso llamando a sus métodos y teniendo acceso a sus propiedades, eventos y campos.

En la tabla siguiente se proporcionan las descripciones de algunas de las características que el motor en tiempo de ejecución permite que tengan las clases. En esta lista no se incluyen las características disponibles mediante las clases de Atributo. Un lenguaje concreto podría hacer que no estén disponibles todas estas características.

Característica	Description
sealed	Especifica que de este tipo no se puede derivar otro.
implements	Indica que la clase utiliza una o varias interfaces proporcionando implementaciones de miembros de la interfaz.
abstract	Especifica que no se puede crear una instancia de esta clase. Para utilizarla se debe derivar de ella otra clase.
inherits	Indica que las instancias de la clase se pueden utilizar en cualquier lugar en que se especifique la clase base. Una clase derivada que hereda de una clase base puede utilizar la implementación de todos los métodos virtuales que proporciona la clase base o los puede reemplazar con su propia implementación.
exported o not exported	Indica si una clase está visible fuera del ensamblado en el que se define. Sólo se aplica a las clases de nivel superior.

Las clases anidadas tienen también características de miembro.

Los miembros de clase que no tienen implementación son miembros abstractos. Una clase que tiene uno o varios miembros abstractos es abstracta y no se pueden crear nuevas instancias de ella. Algunos lenguajes cuyo destino es el motor en tiempo de ejecución permiten marcar una clase como abstracta incluso si no tiene ningún miembro abstracto. Se puede utilizar una clase abstracta cuando sea necesario encapsular un conjunto básico de funcionalidad que las clases derivadas pueden heredar o reemplazar según sea adecuado. Las clases que no son abstractas se conocen como clases concretas.

Una clase puede implementar cualquier número de interfaces, pero sólo puede heredar de una clase base. Todas las clases deben tener al menos un constructor, que inicializa nuevas instancias de la clase.

Cada lenguaje compatible con el motor en tiempo de ejecución proporciona una forma de indicar que una clase o un miembro de clase tiene características especiales. Cuando se utiliza la sintaxis que requiere un lenguaje, el lenguaje garantiza que las características de la clase y sus miembros se almacenan (como metadatos) junto con la implementación de la clase.

Delegados del Sistema de Tipos Común

El motor en tiempo de ejecución admite tipos de referencia denominados delegados cuya finalidad es parecida a la de los punteros a función de C++. A diferencia de los punteros a función, los delegados son seguros, se pueden comprobar y proporcionan seguridad de tipos. Un tipo de delegado puede representar cualquier método con una firma compatible. Mientras que los punteros a función sólo pueden representar funciones estáticas, un delegado puede representar tanto métodos estáticos como de instancia. Los delegados se utilizan para los controladores de eventos y las funciones de devolución de llamada en .NET Framework.

Nota:

Common Language Runtime no admite la serialización de métodos globales, por lo que no se pueden utilizar los delegados para ejecutar métodos globales en otros dominios de aplicación.

Todos los delegados heredan de MulticastDelegate, que hereda de Delegate. Los lenguajes C#, Visual Basic y C++ no permiten la herencia a partir de estos tipos; en su lugar, proporcionan palabras clave para declarar delegados.

Dado que los delegados heredan de MulticastDelegate, un delegado tiene una lista de invocación, que es una lista de métodos que representa el delegado y que se ejecutan cuando se llama al delegado. Todos los métodos de la lista reciben los argumentos proporcionados cuando se invoca al delegado.

Nota:

El valor devuelto no se define para los delegados que tienen más de un método en su lista de invocación, aunque el delegado tenga un tipo de valor devuelto.

Crear y utilizar delegados

En muchos casos, como en el de los métodos de devolución de llamada, un delegado sólo representa un método y las únicas acciones que es necesario llevar a cabo se limitan a crear el delegado e invocarlo.

Por lo que se refiere a los delegados que representan varios métodos, .NET Framework proporciona métodos de las clases de delegado Delegate y MulticastDelegate para operaciones tales como agregar un método a una lista de invocación del delegado (el método Delegate.Combine), quitar un método (el método Delegate.Remove) y obtener la lista de invocación (el método Delegate.GetInvocationList).

Nota:

No es preciso utilizar estos métodos para delegados de controladores de eventos en C#, C++ ni Visual Basic, ya que estos lenguajes proporcionan sintaxis para agregar y quitar controladores de eventos.

Delegados estáticos cerrados y delegados de instancia abiertos

Los delegados pueden representar métodos **static** (**Shared** en Visual Basic) o métodos de instancia. Por lo general, cuando un delegado representa un método de instancia, la instancia se enlaza al delegado junto con el método. Por ejemplo, un delegado de controladores de eventos podría tener tres métodos de instancia en su lista de invocación, cada uno de ellos con una referencia al objeto al que pertenece el método.

En la versión 2.0 de .NET Framework también es posible crear un delegado abierto para un método de instancia. Un método de instancia tiene un parámetro de instancia implícito (representado por **this** en C# o por **Me** en Visual Basic) y se puede representar mediante un tipo de delegado que exponga este parámetro oculto. Es decir, el tipo de delegado debe tener un parámetro adicional al principio de su lista de parámetros formales, del mismo tipo que la clase a la que pertenece el método. También se admite una situación opuesta a la de este escenario, de modo que es posible enlazar el primer argumento de un método estático.

Nota:

La creación de delegados de instancia abiertos y estáticos cerrados no está admitida directamente en Visual Basic, C++ o C# para constructores delegados. En su lugar, debe utilizarse una de las sobrecargas de método Delegate.CreateDelegate que especifica objetos MethodInfo, como Delegate.CreateDelegate(Type, Object, MethodInfo, Boolean).

⊟ Reglas relajadas para enlace de delegado

En .NET Framework versión 2.0, los tipos de parámetros y el tipo de valor devuelto de un delegado debe ser compatible con los tipos de parámetros y el tipo de valor devuelto del método que representa el delegado; los tipos no tienen que coincidir exactamente.

Nota:
En las versiones 1.0 y 1.1 de .NET Framework los tipos deben coincidir exactamente.

Un parámetro de un delegado es compatible con el parámetro correspondiente de un método si el tipo del parámetro del delegado es más restrictivo que el tipo del parámetro del método, dado que así se garantiza que un argumento pasado al delegado se pueda pasar al método de un modo seguro.

Asimismo, el tipo de valor devuelto de un delegado es compatible con el tipo de valor devuelto de un método si el tipo de valor devuelto del método es más restrictivo que el tipo de valor devuelto del delegado, dado que así se garantiza que el valor devuelto del método pueda convertirse de un modo seguro al tipo de valor devuelto del delegado.

Por ejemplo, un delegado con un parámetro de tipo Hashtable y un tipo de valor devuelto Object puede representar un método con un parámetro de tipo Object y un valor devuelto de tipo Hashtable.

⊟ Delegados y llamadas asincrónicas a métodos

Cada delegado tiene un método **BeginInvoke** que permite llamar de forma asincrónica al delegado, así como un método **EndInvoke** que limpia después los recursos. Estos métodos se generan automáticamente para cada tipo de delegado. Cuando se invoca un delegado con el método **BeginInvoke**, el método que representa el delegado se ejecuta en un subproceso que pertenece a un objeto ThreadPool.

Matrices en el Sistema de Tipos Común

Un tipo de matriz se define especificando el tipo de elemento de la matriz, el rango (número de dimensiones) de la matriz y los límites superior e inferior de cada dimensión de la matriz. Todos los elementos se incluyen en todas las firmas de un tipo de matriz, aunque se pueden marcar como proporcionados dinámicamente y no estáticamente. El motor en tiempo de ejecución crea automáticamente tipos de matriz exactos cuando son necesarios; no es necesaria una definición aparte de los tipos de matriz. Las matrices de un tipo dado sólo pueden contener elementos de ese tipo.

Los valores de tipo de matriz son objetos. Los objetos de matriz se definen como una serie de ubicaciones donde se almacenan los valores del tipo de elemento de la matriz. El número de valores repetidos se determina mediante el rango y los límites de la matriz.

Los tipos de matriz derivan del tipo System.::.Array. Esta clase representa todas las matrices independientemente del tipo de elementos o de rango. Las operaciones definidas en las matrices son: asignación de una matriz basándose en la información de tamaño y de límite inferior; indización de una matriz para leer y escribir un valor; cálculo de la dirección de un elemento de una matriz (un puntero administrado) y búsqueda del rango, los límites y el número total de valores almacenados en una matriz.

Las matrices de una dimensión con un límite inferior a cero para sus elementos (denominados en ocasiones vectores) tienen un tipo basado en el tipo de los elementos de la matriz, independientemente del límite superior. Las matrices con más de una dimensión, o con una dimensión pero con un límite inferior distinto de cero, tienen el mismo tipo si tienen el mismo tipo de elemento y rango, independientemente del límite inferior de la matriz. No se admiten matices con cero dimensiones.

Interfaces en el Sistema de Tipos Común

Las interfaces pueden tener propiedades, métodos y eventos, que son miembros abstractos. Todas las clases que implementen una interfaz deben proporcionar definiciones de los miembros abstractos declarados en la interfaz. Una interfaz puede requerir que cualquier clase que implemente una interfaz implemente también otras interfaces.

A las interfaces se les aplican las restricciones siguientes:

- Una interfaz se puede declarar con cualquier tipo de accesibilidad, pero los miembros de la interfaz deben tener todos accesibilidad pública.
- No se puede asociar permisos de seguridad ni a la interfaz ni a sus miembros.
- Las interfaces no pueden definir constructores
- Las interfaces no pueden definir campos.
- Todas las propiedades, métodos y eventos abstractos definidos en una interfaz deben ser miembros de instancia, no pueden ser miembros estáticos.

Cada lenguaje debe proporcionar reglas para asignar una implementación a la interfaz que requiere el miembro, ya que varias interfaces pueden declarar un miembro con la misma firma y esos miembros pueden tener implementaciones independientes.

Punteros en el Sistema de Tipos Común

Los punteros son clases especiales de variables. Hay tres tipos de punteros compatibles con el motor en tiempo de ejecución: punteros administrados, punteros no administrados y punteros a función no administrados.

Un puntero administrado, también conocido como identificador de un objeto en el montón administrado, es un nuevo tipo de puntero disponible para las aplicaciones administradas. Los punteros administrados son referencias a un bloque administrado de memoria del montón de Common Language Runtime. En este montón se realiza una recolección automática de elementos no utilizados. Los punteros administrados se generan para argumentos de método que se pasan por referencia. Algunos lenguajes proporcionan otras formas de generar punteros administrados. Sólo los punteros administrados son compatibles con CLS.

Nota:

En Visual C++ 2002 y Visual C++ 2003, __gc * se empleaba para declarar un puntero administrado. Esto se ha reemplazado por un ^ en Visual C++ 2005; por ejemplo ArrayList^ al = gcnew ArrayList();.

Un puntero no administrado es el puntero C++ tradicional a un bloque de memoria no administrado del montón estándar C++. Como los punteros no administrados no forman parte de CLS (Common Language Specification), ciertos lenguajes pueden no proporcionar sintaxis para definir estos tipos o tener acceso a ellos. Vea la documentación del lenguaje para obtener información sobre la compatibilidad con punteros no administrados.

Un puntero a función no administrado es también un puntero C++ tradicional que hace referencia a la dirección de una función. CLS proporciona delegados como alternativa administrada a los punteros no administrados.

No es necesaria la definición explícita de un tipo de puntero. Toda la información necesaria para determinar el tipo de un puntero está ahí cuando se declara el puntero.

Mientras que los tipos de puntero son tipos de referencia, el valor de un tipo de puntero no es un objeto y no se puede determinar el tipo exacto a partir de tal valor.

El sistema de tipos común proporciona dos operaciones de tipo seguro en los tipos de puntero: la carga de un valor desde una ubicación a la que hace referencia el puntero y la escritura de un valor en dicha ubicación. Estas operaciones de tipo seguro son compatibles con CLS.

El sistema de tipos común proporciona también tres operaciones aritméticas de dirección basadas en bytes en los tipos de puntero: agregar y sustraer enteros en los punteros y sustraer un puntero de otro. Los resultados de las dos primeras operaciones aritméticas devuelven un valor del mismo tipo que el puntero original. Estas operaciones basadas en bytes no son compatibles con CLS.

Metadatos y Componentes Autodescriptivos

Hasta ahora, un componente de software (.exe o .dll) escrito en un lenguaje no podía usar con facilidad componentes de software escritos en otro lenguaje. COM supuso un paso adelante en la resolución de este problema. .NET Framework hace la interoperación entre componentes todavía más fácil, permitiendo que los compiladores emitan información de declaración adicional en todos los módulos y ensamblados. Esta información, llamada metadatos, contribuye a que los componentes interactúen sin problemas.

Información General sobre Metadatos

Los metadatos son información binaria que describe un programa, almacenada en un archivo ejecutable portable (PE) de Common Language Runtime o en memoria. Cuando se compila el código en un archivo PE, los metadatos se insertan en una parte del archivo, mientras que el código se convierte en lenguaje intermedio de Microsoft (MSIL) y se inserta en otra parte del archivo. Cada tipo y miembro definido, o al que se hace referencia, en un módulo o ensamblado se describe en los metadatos. Cuando se ejecuta código, el motor en tiempo de ejecución carga los metadatos en la memoria y hace referencia a ellos para obtener información acerca de las clases, miembros, herencia, etc., del código.

Los metadatos describen todos los tipos y miembros definidos en el código mediante un lenguaje neutro. Los metadatos almacenan la siguiente información:

- Descripción del ensamblado
  - Identidad (nombre, versión, referencia cultural, clave pública).
  - Los tipos que se exportan.
  - Otros ensamblados de los que depende éste.
  - Permisos de seguridad que hay que ejecutar.
- Descripción de los tipos.
  - Nombre, visibilidad, clase base e interfaces implementadas.
  - Miembros (métodos, campos, propiedades, eventos, tipos anidados).
- Atributos.
  - Elementos descriptivos adicionales que modifiquen los tipos y los miembros.

⊟ Ventajas de los metadatos

Los metadatos son la clave para un modelo de programación más sencillo, eliminando la necesidad de tener archivos de Lenguaje de definición de interfaz (IDL), archivos de encabezado o cualquier método externo de referencia a componentes. Los metadatos permiten que los lenguajes .NET se describan a sí mismos automáticamente usando un lenguaje de forma neutral, que no ven ni el programador ni el usuario. Además, los metadatos se pueden extender mediante el uso de atributos. Los metadatos proporcionan las siguientes ventajas principales:

- Archivos autodescriptivos

  Los módulos y ensamblados de Common Language Runtime son autodescriptivos. Los metadatos de un módulo contienen todo lo necesario para interactuar con otro módulo. Los metadatos proporcionan automáticamente la funcionalidad del IDL en COM, lo que permite usar un archivo para la definición y la implementación. Los módulos y ensamblados de Common Language Runtime no necesitan ni registrarse en el sistema operativo. En consecuencia, las descripciones que usa el motor en tiempo de ejecución reflejan siempre el código actual del archivo compilado, lo que aumenta la confiabilidad de la aplicación.

- Interoperabilidad de lenguajes y diseño más sencillo, basado en el componente.

  Los metadatos proporcionan toda la información necesaria sobre el código compilado para derivar clases de archivos PE escritos en otro lenguaje. Se puede crear una instancia de cualquier clase escrita en cualquier lenguaje administrado (cualquier lenguaje dirigido a Common Language Runtime) sin tener que preocuparse por el cálculo de referencias explícito ni por usar código de interoperabilidad personalizado.

- Atributos.

  .NET Framework permite declarar tipos específicos de metadatos, denominados atributos, en el archivo compilado. Los atributos se encuentran en todo .NET Framework y se usan para controlar más minuciosamente el comportamiento del programa en tiempo de ejecución. Además, se pueden emitir metadatos personalizados propios en los archivos .NET Framework mediante atributos personalizados definidos por el usuario.

Estructura y Uso de los Metadatos

Aunque la mayoría de los programadores no necesitan conocer los detalles de la implementación de metadatos, algunos podrían desear una descripción más detallada. En esta sección se proporciona una descripción general de cómo se almacenan los metadatos en un archivo portable ejecutable (PE) de .NET Framework y una explicación sobre la función de los metadatos en la ejecución administrada. No es necesario leer esta sección para comprender la programación .NET o el uso de los atributos.

Metadatos y la Estructura del Archivo PE

Los metadatos se almacenan en una sección de un archivo ejecutable portable (PE) de .NET Framework, mientras que el lenguaje intermedio de Microsoft (MSIL) se guarda en otra sección del mismo archivo. La parte de los metadatos del archivo contiene una serie de estructuras de datos de tablas y montones. La parte del MSIL contiene símbolos (token) de MSIL y de metadatos que hacen referencia a la parte de metadatos del archivo PE. Se pueden encontrar símbolos de metadatos al utilizar herramientas como el Desensamblador de MSIL (Ildasm.exe) para ver el MSIL del código o el Depurador en tiempo de ejecución (Cordbg.exe) para realizar un volcado de memoria.

⊟ Tablas y montones de metadatos

Cada tabla de metadatos tiene información sobre los elementos del programa. Por ejemplo, una tabla de metadatos describe las clases del código, otra los campos, etc. Si hay diez clases en el código, la tabla

de clases tendrá diez filas, una por clase. Las tablas de metadatos hacen referencia a otras tablas y montones. Por ejemplo, la tabla de metadatos de clases hace referencia a la tabla de métodos.

Los metadatos también almacenan información en cuatro estructuras de montón: cadena, objeto binario, cadena de usuario y GUID. Todas las cadenas usadas en los nombres de los tipos y los miembros se almacenan en el montón de cadenas. Por ejemplo, una tabla de métodos no almacena directamente el nombre de un método concreto, sino que señala al nombre del método almacenado en el montón de cadenas.

⊟ Símbolos (token) de metadatos

Cada fila de cada tabla de metadatos se identifica de forma exclusiva en la parte de MSIL del archivo PE mediante un símbolo de metadatos. Los símbolos de metadatos responden a un concepto similar a los punteros, que persisten en MSIL, que hacen referencia a una tabla de metadatos concreta.

Un símbolo de metadatos es un número de cuatro bytes. El byte superior indica la tabla de metadatos a la que se refiere un símbolo concreto (método, tipo, etc.). Los tres bytes restantes especifican la fila de la tabla de metadatos que corresponde al elemento de programación que se describe. Si se define un método en C# y se compila en un archivo PE, en la porción de MSIL del archivo PE podrían aparecer los símbolos de metadatos siguientes:

```
0x06000004
```

El byte superior (0x06) indica que éste es un símbolo de **MethodDef**. Los tres bytes inferiores (000004) indican a Common Language Runtime que busque en la cuarta fila de la tabla **MethodDef** la información que describe la definición de este método.

⊟ Metadatos en un archivo PE

Cuando se compila un programa para Common Language Runtime, se convierte en un archivo PE formado por tres partes. La tabla siguiente describe el contenido de cada una de estas partes.

Sección del archivo PE	Contenido de la sección del archivo PE
Encabezado del archivo PE	El índice de las secciones principales del archivo PE y la dirección del punto de entrada. El motor en tiempo de ejecución usa esta información para identificar el archivo como archivo PE y para determinar dónde comienza la ejecución al cargar el programa en la memoria.
Instrucciones MSIL	Instrucciones del Lenguaje intermedio de Microsoft (MSIL) que contiene el código. Muchas de las instrucciones MSIL van acompañadas por símbolos de metadatos.
Metadatos	Tablas y montones de metadatos. El motor en tiempo de ejecución usa esta sección para registrar información sobre todos los tipos y miembros del código. Esta sección incluye también atributos personalizados e información de seguridad.

Uso de Metadatos en Tiempo de Ejecución

Para comprender mejor los metadatos y su uso en Common Language Runtime, puede resultar útil construir un programa sencillo y mostrar cómo afectan los metadatos a su comportamiento durante su ejecución. El siguiente ejemplo de código muestra dos métodos dentro de una clase llamada MyApp. El método Main es el punto de entrada del programa, mientras que el método Add simplemente devuelve la suma de dos argumentos de enteros.

Visual Basic

```
Public Class MyApp
   Public Shared Sub Main()
      Dim ValueOne As Integer = 10
      Dim ValueTwo As Integer = 20
      Console.WriteLine("The Value is: {0}", Add(ValueOne, ValueTwo))
   End Sub

   Public Shared Function Add(One As Integer, Two As Integer) As Integer
      Return (One + Two)
   End Function
End Class
```

C#

```
using System;
public class MyApp
{
   public static int Main()
   {
      int ValueOne = 10;
      int ValueTwo = 20;
      Console.WriteLine("The Value is: {0}", Add(ValueOne, ValueTwo));
      return 0;
   }
   public static int Add(int One, int Two)
   {
      return (One + Two);
   }
}
```

Cuando se ejecuta el código, el motor en tiempo de ejecución carga el módulo en la memoria y consulta los metadatos de esta clase. Una vez cargado, el motor en tiempo de ejecución realiza una análisis exhaustivo de la secuencia de lenguaje intermedio de Microsoft (MSIL) del método para convertirla en rápidas instrucciones máquina nativas. El motor en tiempo de ejecución usa un compilador Just-In-Time (JIT) para convertir las instrucciones MSIL en código máquina nativo, método a método, según sea necesario.

En el siguiente ejemplo de código se muestra parte del MSIL producido a partir de la función Main del código anterior. El MSIL y los metadatos se pueden ver desde cualquier aplicación de .NET Framework usando el Desensamblador de MSIL (Ildasm.exe).

```
      .entrypoint
      .maxstack  3
      .locals ([0] int32 ValueOne,
               [1] int32 ValueTwo,
               [2] int32 V_2,
               [3] int32 V_3)
      IL_0000:  ldc.i4.s   10
      IL_0002:  stloc.0
      IL_0003:  ldc.i4.s   20
      IL_0005:  stloc.1
      IL_0006:  ldstr      "The Value is: {0}"
      IL_000b:  ldloc.0
      IL_000c:  ldloc.1
      IL_000d:  call int32 ConsoleApplication.MyApp::Add(int32,int32) /*
06000003 */
```

El compilador JIT lee el MSIL de todo el método, lo analiza exhaustivamente y genera instrucciones nativas efectivas para ese método. En IL_000d se encuentra un símbolo de metadatos del método Add (/* 06000003 */), y el motor en tiempo de ejecución usa dicho símbolo para consultar la tercera fila de la tabla **MethodDef**.

En la siguiente tabla se muestra parte de la tabla **MethodDef** a la que hace referencia el símbolo de los metadatos que describe el método Add. Aunque existen otras tablas de metadatos en el ensamblado y tienen sus propios valores únicos, sólo se trata esta tabla.

Fila	Dirección relativa virtual (RVA)	ImplFlags	Indicadores	Nombre (señala el montón de cadenas).	Firma (señala el montón de objetos binarios)
1	0x00002050	IL Administrado	Public ReuseSlot SpecialName RTSpecialName .ctor	.ctor (constructor)	
2	0x00002058	IL Administrado	Public Static ReuseSlot	Main	String
3	0x0000208c	IL Administrado	Public Static ReuseSlot	Add	int, int, int

Cada columna de la tabla contiene información importante sobre el código. La columna **RVA** permite que el motor en tiempo de ejecución calcule la dirección de memoria de inicio del MSIL que define este

método. Las columnas **ImplFlags** y **Flags** contienen máscaras de bits que describen el método (por ejemplo, si el método es público o privado). La columna **Nombre** indiza el nombre del método del montón de cadenas. La columna **Firma** indiza la definición de la firma del método del montón de objetos binarios.

El motor en tiempo de ejecución calcula la dirección de desplazamiento deseada desde la columna **RVA** de la tercera fila y la devuelve al compilador JIT, que, a continuación, se dirige a la nueva dirección. El compilador JIT continúa procesando el MSIL en la nueva dirección hasta que encuentra otro símbolo de metadatos y se repite el proceso.

Usando metadatos, el motor en tiempo de ejecución tiene acceso a toda la información que necesita para cargar el código y procesarlo en instrucciones máquina nativas. De este modo, los metadatos hacen posible los archivos autodescriptivos y, junto con el sistema de tipos comunes, la herencia de un lenguaje a otro.

Interoperabilidad entre Lenguajes

Common Language Runtime proporciona compatibilidad integrada para la interoperabilidad entre lenguajes. Sin embargo, esta compatibilidad no garantiza que el código escrito pueda ser utilizado por otros programadores que utilicen otro lenguaje de programación. Para garantizar el desarrollo de código administrado que pueda ser totalmente utilizado por programadores que usen cualquier lenguaje de programación, se ha definido un conjunto de características de lenguaje y reglas para aplicarlas, denominado Common Language Specification (CLS). Los componentes que siguen estas reglas y sólo exponen características CLS se consideran compatibles con CLS.

En esta sección, se describe la compatibilidad integrada en Common Language Runtime para la interoperabilidad entre lenguajes y se explica la función que realiza CLS a la hora de habilitar una interoperabilidad entre lenguajes garantizada. Se identifican las características y reglas de CLS, y se habla sobre la compatibilidad con CLS.

Información General Acerca de la Interoperabilidad de Lenguajes

La interoperabilidad entre lenguajes es la posibilidad de que el código interactúe con código escrito en un lenguaje de programación diferente. La interoperabilidad entre lenguajes puede ayudar a maximizar la reutilización de código y, por tanto, puede mejorar la eficacia del proceso de programación.

Dado que los desarrolladores utilizan una gran variedad de herramientas y tecnologías, cada una de las cuales podría admitir distintos tipos y características, desde tiempo atrás ha sido complicado garantizar la interoperabilidad entre lenguajes. No obstante, los compiladores y las herramientas de lenguaje dirigidos a Common Language Runtime se benefician de la compatibilidad que integra el motor en tiempo de ejecución para la interoperabilidad entre lenguajes.

Common Language Runtime ofrece la base necesaria para la interoperabilidad entre lenguajes al especificar e imponer un sistema de tipos común, y al suministrar metadatos. Debido a que todos los lenguajes dirigidos a Common Language Runtime siguen las reglas del sistema de tipos común para definir y utilizar los tipos, la utilización de tipos es coherente entre todos los lenguajes. Los metadatos habilitan la interoperabilidad entre lenguajes mediante la definición de un mecanismo uniforme para almacenar y recuperar la información sobre tipos. Los compiladores almacenan la información sobre tipos como metadatos y Common Language Runtime usa esta información para proporcionar servicios durante la ejecución; el motor en tiempo de ejecución puede administrar la ejecución de aplicaciones de varios lenguajes porque toda la información de tipos se almacena y recupera de la misma forma, independientemente del lenguaje en que se haya escrito el código.

El código administrado se beneficia de que el motor en tiempo de ejecución admita la interoperabilidad entre lenguajes de las maneras siguientes:

- Los tipos pueden heredar la implementación de otros tipos, pasar objetos a los métodos de otro tipo y llamar a métodos definidos para otros tipos, sea cual sea el lenguaje en que se implementen los tipos.

- Los depuradores, generadores de perfiles u otras herramientas deben reconocer un solo entorno, es decir, MSIL (Microsoft intermediate language, Lenguaje intermedio de Microsoft) y los metadatos de Common Language Runtime, para poder ser compatibles con cualquier lenguaje de programación dirigido al motor en tiempo de ejecución.

- El control de excepciones es coherente entre todos los lenguajes. El código puede producir una excepción en un lenguaje y esa excepción puede ser recibida y reconocida por un objeto escrito en otro lenguaje.

Aunque el motor en tiempo de ejecución permite que todo el código administrado se ejecute en un entorno de múltiples lenguajes, no existe ninguna garantía de que la funcionalidad de los tipos creados pueda ser utilizada completamente por los lenguajes de programación que usen otros desarrolladores. La razón principal es que el compilador de un lenguaje dirigido a Common Language Runtime usa el sistema de tipos y los metadatos que son compatibles con un conjunto de características de lenguaje propio y exclusivo. Si no se sabe en qué lenguaje se va a escribir el código que realiza la llamada, existen pocas posibilidades de saber si las características que expone un componente van a estar accesibles para el llamador. Por ejemplo, si el lenguaje que ha elegido proporciona compatibilidad para enteros sin signo, podría diseñar un método con un parámetro de tipo **UInt32**; sin embargo, ese método sería inútil para un lenguaje que no reconociese los enteros sin signo.

Para garantizar que el código administrado será accesible para los desarrolladores que utilicen cualquier lenguaje de programación, .NET Framework proporciona Common Language Specification (CLS), que describe un conjunto fundamental de características de lenguaje y define reglas sobre cómo utilizar dichas características. Para obtener más información sobre la compatibilidad con CLS en componentes y herramientas

Common Language Specification (CLS)

Para poder interactuar completamente con otros objetos, sea cual sea el lenguaje en que se hayan implementado, los objetos deben exponer a los llamadores sólo aquellas características que sean comunes para todos los lenguajes con los que deben interoperar. Por este motivo, se ha definido Common Language Specification (CLS), que es un conjunto de características de lenguaje básicas requeridas por la mayoría de las aplicaciones. Las reglas de CLS definen un subconjunto del Sistema de tipos comunes, es decir, todas las reglas que se aplican al sistema de tipos común se aplican también a CLS, salvo que se definan reglas más estrictas en CLS. CLS ayuda a mejorar y garantizar la interoperabilidad entre lenguajes mediante la definición de un conjunto de características en las que se pueden basar los programadores y que están disponibles en una gran variedad de lenguajes. CLS también establece los requisitos de compatibilidad con CLS; estos requisitos permiten determinar si el código administrado cumple la especificación CLS y hasta qué punto una herramienta dada admite la programación de código administrado que utilice las características de CLS.

Si un componente sólo utiliza las características de CLS en la API que expone a otro código (incluidas las clases derivadas), se garantiza que se puede obtener acceso al componente desde cualquier lenguaje de programación que admita CLS. Los componentes que cumplen las reglas de CLS y usan sólo las características incluidas en CLS se conocen como componentes compatibles con CLS.

La mayoría de los miembros definidos por tipos en Información general de la biblioteca de clases de .NET Framework son compatibles con CLS. Sin embargo, algunos tipos de la biblioteca de clases tienen uno o más miembros que no son compatibles con CLS. Estos miembros permiten el uso de características de lenguaje que no se encuentran en CLS. Los tipos y miembros que no son compatibles con CLS se identifican como tales en la documentación de referencia y, en todos los casos, existe una alternativa compatible con CLS.

CLS se diseñó de manera que fuese lo suficientemente amplio como para incluir las construcciones de lenguaje que normalmente necesitan los programadores y lo suficientemente pequeño como para que todos los lenguajes pudieran admitirlo. Además, se ha excluido de CLS cualquier construcción de lenguaje de la que no se puede verificar rápidamente la seguridad de tipos del código, de manera que todos los lenguajes compatibles con CLS pueden generar código que es posible comprobar.

En la tabla siguiente, se resumen las características que se incluyen en CLS y se indica si cada característica se aplica a programadores y compiladores (Todos) o sólo a compiladores. Se pretende que sea informativa, pero no detallada. Para obtener información detallada, consulte la especificación de Common Language Infrastructure, Partition I, disponible en el sitio web Microsoft Developer Network.

Característica	Se aplica a	Descripción
General		
Visibilidad	Todo	Las reglas de CLS se aplican sólo a las partes de un tipo que se expongan fuera del ensamblado que realiza la definición.
Miembros globales	Todo	Los campos y métodos **static** globales no son compatibles con CLS.
Nombres		
Caracteres y mayúsculas	Todo	Los compiladores de lenguaje compatibles con CLS deben seguir las reglas del anexo 7 del informe técnico 15 del estándar Unicode 3.0, que controla el conjunto de caracteres que pueden iniciar e incluirse en los identificadores. Este estándar está disponible en el sitio web de Unicode Consortium. Para que dos identificadores se consideren distintos, deben diferenciarse por algo más que el uso de mayúsculas o minúsculas.
Keywords	Compiladores	Los compiladores de lenguajes compatibles con CLS proporcionan un mecanismo para hacer referencia a identificadores que coinciden con palabras clave. Los compiladores de lenguajes compatibles con CLS proporcionan un mecanismo para definir y reemplazar los métodos virtuales por nombres que son palabras clave en el lenguaje.
Exclusividad	Todo	Todos los nombres de un ámbito compatible con CLS deben ser distintos, incluso cuando los nombres sean para dos clases de miembros diferentes, excepto cuando los nombres son idénticos y se resuelven mediante sobrecarga. Por ejemplo, CLS no permite que un tipo único use el mismo nombre para un método y un campo.
Prototipos	Todo	Todos los tipos devueltos y los tipos de parámetros que aparecen en un prototipo de tipo o miembro deben ser compatibles con CLS.

Tipos		
Tipos primitivos	Todo	La biblioteca de clases de .NET Framework incluye tipos que se corresponden con los tipos de datos primitivos que usan los compiladores. De entre estos tipos, son compatibles con CLS los siguientes: Byte, Int16, Int32, Int64, Single, Double, Boolean, Char, Decimal, IntPtr y String.
Tipos encuadrados	Todo	Los tipos de valor encuadrados (tipos de valor que se han convertido en objetos) no son parte de CLS. Utilice en su lugar System.Object, System.ValueType o System.Enum, según corresponda.
Visibilidad	Todo	Las declaraciones de tipos y miembros no deben contener tipos que sean menos visibles o accesibles que el tipo o miembro que se va a declarar.
Métodos de interfaz	Compiladores	Los compiladores de lenguajes compatibles con CLS deben incluir sintaxis para el caso de que un tipo único implemente dos interfaces y cada una de esas interfaces requiera la definición de un método con el mismo nombre y el mismo prototipo. Tales métodos se deben considerar distintos y no necesitan tener la misma implementación.
Cierre	Todo	Los miembros individuales de interfaces y clases abstractas compatibles con CLS deben definirse para ser compatibles con CLS.
Invocación de constructores	Todo	Para poder tener acceso a datos de instancias heredadas, un constructor debe llamar al constructor de la clase base.
Referencias a tipos	Todo	Las referencias a tipos no son compatibles con CLS. Una referencia a un tipo es una construcción especial que contiene una referencia a un objeto y una referencia a un tipo. Las referencias a tipos permiten que Common Language Runtime proporcione compatibilidad con los estilos de C++ a los métodos que tienen un número variable de argumentos.
Miembros de tipos		
Sobrecarga	Todo	Se permite la sobrecarga de propiedades indizadas, métodos y constructores; no se deben sobrecargar campos y eventos. Las propiedades no se deben sobrecargar por tipo (es decir, por el tipo de valor devuelto de su método Get), pero se pueden sobrecargar con números o tipos de índices diferentes. Sólo se permite la sobrecarga de métodos si se basa en el número y en los tipos de parámetros y, en el caso de los métodos genéricos, si se basa en el número de parámetros genéricos. La sobrecarga de operadores no se incluye en CLS. Sin embargo, CLS proporciona directrices sobre cómo incluir nombres útiles (como Add()) y cómo establecer un bit en los metadatos. Se recomienda que los compiladores que decidan incluir la sobrecarga de operadores sigan estas directrices, aunque no están obligados a ello.
Exclusividad de miembros de sobrecarga	Todo	Los campos y los tipos anidados deben distinguirse únicamente por comparación de identificador. Los métodos, las propiedades y los eventos que tengan el mismo nombre (por comparación de identificador) deben distinguirse por algo más que el tipo de valor devuelto.

Operadores de conversión	Todo	Si **op_Implicit** u **op_Explicit** se sobrecarga sobre el tipo de valor devuelto, se debe incluir una forma alternativa para proporcionar la conversión.
Métodos		
Accesibilidad de los métodos reemplazados	Todo	La accesibilidad no se debe cambiar al reemplazar los métodos heredados, excepto cuando se reemplace un método heredado a partir de un ensamblado diferente con accesibilidad **FamilyOrAssembly**. En este caso, el reemplazo debe tener accesibilidad **Family**.
Listas de argumentos	Todo	La única convención de llamada admitida por CLS es la convención de llamada administrada estándar; no se permiten las listas de argumentos de longitud variable. (Utilice la palabra clave **ParamArray** de Microsoft Visual Basic y la palabra clave **params** de C# para permitir un número variable de argumentos.)
Propiedades		
Metadatos de descriptor de acceso	Compiladores	Los métodos Get y Set que implementan los métodos de una propiedad se marcan con el identificador **mdSpecialName** en los metadatos.
Modificadores	Todo	La propiedad y sus descriptores de acceso deben ser **static**, **virtual** o **instance**.
Nombres de descriptores de acceso	Todo	Las propiedades deben seguir los patrones de nombres específicos. Para una propiedad denominada **Name**, el método Get, si se define, se denominará **get_Name** y el método Set, si se define, se denominará **set_Name**.
Tipo de valor devuelto y argumentos	Todo	El tipo de la propiedad es el tipo de valor devuelto del método Get y el tipo del último argumento del método Set. Los tipos de los parámetros de la propiedad son los tipos de los parámetros del método Get y los tipos de todos los parámetros del método Set, excepto el último. Todos estos tipos deben ser compatibles con CLS y no pueden ser punteros administrados; no deben pasarse por referencia.
Eventos		
Métodos de evento	Todo	Los métodos para agregar y quitar un evento deben estar ambos presentes o ausentes.
Metadatos de los métodos de evento	Compiladores	Los métodos que implementen un evento deben marcarse con el identificador **mdSpecialName** en los metadatos.
Accesibilidad del descriptor de acceso	Todo	La accesibilidad de los métodos para agregar, quitar y provocar un evento debe ser idéntica.
Modificadores	Todo	Los métodos para agregar, quitar y provocar un evento deben ser **static**, **virtual** o **instance**.
Nombres de los métodos de evento	Todo	Los eventos deben seguir los patrones de nombres específicos. Para un evento denominado **MyEvent**, el método Add, si se define, se denominará **add_MyEvent**, el método Remove, si se define, se denominará **remove_MyEvent** y el método Raise se denominará **raise_MyEvent**.

Argumentos	Todo	Los métodos que se utilizan para agregar y quitar un evento deben tomar un parámetro cuyo tipo defina el tipo del evento, y ese tipo debe derivarse de System..::.Delegate.
Tipos de puntero		
Punteros	Todo	Los tipos de puntero y los tipos de puntero a función no son compatibles con CLS.
Interfaces		
Prototipos de miembros	Todo	Las interfaces compatibles con CLS no deben requerir la definición de métodos no compatibles con CLS para su implementación.
Modificadores de miembros	Todo	Las interfaces compatibles con CLS no pueden definir métodos estáticos ni pueden definir campos. Pueden definir propiedades, eventos y métodos virtuales.
Tipos de referencia		
Invocación de constructores	Todo	Para los tipos de referencia, se llama a los constructores de objetos sólo como parte de la creación de un objeto, y los objetos se inicializan una sola vez.
Tipos de clase		
Herencia	Todo	Las clases compatibles con CLS deben heredarse de clases compatibles con CLS (System..::.Object es compatible con CLS).
Matrices [1]		
Tipos de elemento	Todo	Los elementos de matriz deben ser tipos compatibles con CLS.
Dimensiones	Todo	Las matrices deben tener un número fijo de dimensiones, mayores que cero.
Límites	Todo	Todas las dimensiones de una matriz deben tener un límite menor que cero.
Enumeraciones		
Tipo subyacente	Todo	El tipo subyacente de una enumeración debe ser un tipo entero integrado en CLS (Byte, Int16, Int32 o Int64).
FlagsAttribute	Compiladores	La presencia del atributo personalizado System..::.FlagsAttribute en la definición de una enumeración indica que la enumeración debe tratarse como un conjunto de campos de bits (indicadores), y la ausencia de este atributo indica que el tipo debe verse como un grupo de constantes enumeradas. Se recomienda que en los lenguajes se utilice **FlagsAttribute** o sintaxis específica del lenguaje para distinguir estos dos tipos de enumeraciones.
Miembros de campo	Todo	Los campos **static** literales de una enumeración deben ser del mismo tipo que el tipo de la enumeración.
Excepciones		
Herencia	Todo	Los objetos que se produzcan deberán ser de tipo

		System..::.Exception o heredarse de **System.Exception**.
Atributos personalizados		
Codificaciones de valores	Compiladores	Los compiladores compatibles con CLS deben tratar sólo un subconjunto de las codificaciones de los atributos personalizados (la representación de los atributos personalizados en los metadatos). Los únicos tipos que pueden aparecer en estas codificaciones son: System..::.Type, System..::.String, System..::.Char, System..::.Boolean, System..::.Byte, System..::.Int16, System..::.Int32, System..::.Int64, System..::.Single, System..::.Double y cualquier tipo de enumeración basado en un tipo entero base compatible con CLS.
Metadatos		
Compatibilidad con CLS	Todo	Los tipos que tienen una compatibilidad con CLS distinta a la del ensamblado en el que se definen deberán marcarse correspondientemente mediante System..::.CLSCompliantAttribute. De manera similar, los miembros cuya compatibilidad con CLS sea diferente de la de su tipo también se deben marcar. Si un miembro o tipo está marcado como no compatible con CLS, debe proporcionarse una alternativa compatible con CLS.
Genéricos		
Nombres de tipo	Compiladores	El nombre de un tipo genérico debe codificar el número de parámetros de tipo declarados en el tipo. El nombre de un tipo genérico anidado debe codificar el número de parámetros de tipo recién introducidos en el tipo.
Tipos anidados	Compiladores	Los tipos anidados deben tener, como mínimo, el mismo número de parámetros genéricos que el tipo contenedor. Los parámetros genéricos de un tipo anidado se corresponden por posición con los parámetros genéricos del tipo contenedor.
Restricciones	Todo	Todo tipo genérico deberá declarar restricciones suficientes como para garantizar que cualquier restricción de las interfaces o del tipo base se vea satisfecha por las restricciones del tipo genérico.
Tipos de restricción	Todo	Los tipos que se utilizan como restricciones en parámetros genéricos deberán ser compatibles con CLS.
Prototipos de miembros	Todo	Se entiende que la visibilidad y accesibilidad de los miembros (incluidos los tipos anidados) de un tipo genérico del que se ha creado una instancia quedan restringidas al ámbito específico de creación de instancias, y no a la declaración de tipos genéricos en general.
Métodos genéricos	Todo	Cada método genérico abstracto o virtual deberá tener su propia implementación concreta (no abstracta) predeterminada

1. Las matrices escalonadas, es decir, matrices de matrices son compatibles con CLS. En la versión 1.0 de .NET Framework, el compilador de C# informa erróneamente de que no lo son.

Escribir Código conforme con CLS

En general, la compatibilidad con Common Language Specification (CLS) hace referencia a la exigencia de que se sigan las reglas y restricciones de CLS. Sin embargo, el concepto tiene un significado más específico, dependiendo de si se está describiendo código compatible con CLS o herramientas de programación compatibles con CLS, como un compilador. Las herramientas compatibles con CLS pueden ayudar a escribir código compatible con CLS.

⊟ Código compatible con CLS

Si desea que su código sea compatible con CLS, debe exponer la funcionalidad de manera que sea compatible con CLS en los lugares siguientes:

- Definiciones de clases públicas.
- Definiciones de los miembros públicos de las clases públicas, y de los miembros accesibles por las clases derivadas (acceso **family**).
- Parámetros y tipos devueltos de los métodos públicos de las clases públicas, y de los métodos accesibles para las clases derivadas.

No es necesario que cumplan las reglas de CLS las características que utilice en las definiciones de las clases privadas, en las definiciones de los métodos privados para las clases públicas y en las variables locales. También puede utilizar las características de lenguaje que desee en el código que implementa la clase y todavía tener un componente compatible con CLS.

Nota:
La matrices escalonadas, es decir, matrices de matrices son compatibles con CLS. En la versión 1.0 de .NET Framework, el compilador de C# informa erróneamente de que no lo son.

Se puede marcar ensamblados, módulos, tipos y miembros como compatibles o no compatibles con CLS mediante CLSCompliantAttribute. Todos los ensamblados que se desee que sean compatibles con CLS deben marcarse como tales. Un ensamblado que no se marque como compatible con CLS se considera que no es compatible con CLS. Si no se aplica ningún atributo de CLS a un tipo, se asume que ese tipo tiene la misma compatibilidad con CLS que el ensamblado en el que se define el tipo. De manera similar, si no se aplica ningún atributo de CLS a un miembro, se considera que ese miembro tiene la misma compatibilidad con CLS que el tipo que lo define. No se puede marcar un elemento de programa como compatible con CLS si el elemento que lo encierra no está marcado como compatible con CLS. En el ejemplo que aparece al final de este tema se ilustra el uso de **CLSCompliantAttribute**.

Los ensamblados, módulos y tipos pueden ser compatibles con CLS incluso aunque algunas partes del ensamblado, módulo o tipo no sean compatibles con CLS, siempre y cuando se cumplan dos condiciones:

- Si el elemento se marca como compatible con CLS, las partes que no son compatibles con CLS deben marcarse mediante **CLSCompliantAttribute**, estableciendo su argumento en **false**.
- Para cada miembro que no sea compatible con CLS, se debe suministrar un miembro alternativo comparable que sea compatible con CLS.

Si diseña una biblioteca de clases compatible con CLS, se garantiza que la biblioteca será interoperable con una gran variedad de lenguajes de programación; por tanto, probablemente, la biblioteca tendrá una base de clientes más amplia que una versión que no sea compatible con CLS.

.NET Framework proporciona una biblioteca de clases compatible con CLS.

⊟ Herramientas compatibles con CLS

Se ha acordado que los lenguajes dirigidos a Common Language Runtime admitan las características de CLS y sigan las reglas de CLS destinadas a los compiladores. Estos compiladores de lenguajes simplifican la compatibilidad con CLS al hacer que los tipos de datos y las características de CLS estén disponibles para crear componentes. Los niveles de compatibilidad con CLS entre los compiladores y otras herramientas se describen del modo siguiente:

- Herramientas de consumidor compatibles con CLS.

  Las herramientas de consumidor son lenguajes que permiten a los programadores tener acceso a todas las características suministradas por las bibliotecas compatibles con CLS. Es posible que los programadores que usen estos lenguajes no puedan extender las bibliotecas compatibles con CLS mediante la creación de nuevos tipos, pero pueden usar cualquier tipo definido por una biblioteca compatible. El nivel de compatibilidad puede ser de utilidad cuando se desea tener acceso a una biblioteca de clases de .NET Framework pero no se necesita crear nuevos objetos para el consumo de otros, como sucede cuando se usan los formularios Web Forms en una página ASP.NET o cuando se crea una interfaz de usuario de Windows Forms.

- Herramientas extensoras compatibles con CLS.

  Las herramientas extensoras son lenguajes que permiten a los programadores usar y extender los tipos definidos en las bibliotecas compatibles con CLS. Los programadores pueden utilizar tipos existentes, así como definir tipos nuevos. Las herramientas extensoras deben cumplir todas las reglas que siguen las herramientas de consumidor, así como algunas reglas adicionales que se describen en la especificación de Common Language Infrastructure, Partition I - Architecture, disponible en el sitio web Microsoft Developer Network.

Cuando se diseñan componentes compatibles con CLS propios, puede ser útil emplear una herramienta compatible con CLS. Escribir componentes compatibles con CLS sin esta ayuda es más difícil, ya que quizás no tenga acceso a todas las características de CLS que desee usar.

Algunos compiladores de lenguajes compatibles con CLS, como el compilador de C# o Visual Basic, permiten especificar que el código debe ser compatible con CLS. Estos compiladores pueden comprobar la compatibilidad con CLS e informar de cuándo el código usa funcionalidad no admitida por CLS. Los compiladores de C# y Visual Basic permiten marcar un elemento de programa como compatible con CLS, lo que hace que el compilador genere un error en tiempo de compilación si el código no es compatible con CLS. Por ejemplo, el código siguiente genera una advertencia del compilador:

Visual Basic

```
<Assembly: CLSCompliant(True)>
<CLSCompliant(True)> Public Class MyCompliantClass
   Public Sub ChangeValue(value As UInt32)
   End Sub
```

```
   Public Shared Sub Main()
      Dim i As Integer = 2
      Console.WriteLine(i)
   End Sub
End Class
```

C#

```
using System;
// Assembly marked as compliant.
[assembly: CLSCompliant(true)]
// Class marked as compliant.
[CLSCompliant(true)]
public class MyCompliantClass {
   // ChangeValue exposes UInt32, which is not in CLS.
   // A compile-time warning results.
   public void ChangeValue(UInt32 value){ }
   public static void Main( ) {
   int i = 2;
   Console.WriteLine(i);
   }
}
```

Este código genera la siguiente advertencia en C#:

```
warning CS3001: Argument type 'uint' is not CLS-compliant
```

o la siguiente advertencia en Visual Basic:

```
warning BC40028: Type of parameter 'value' is not CLS-compliant.
```

Para quitar la advertencia, puede indicar que ChangeValue no es compatible, como se muestra en el ejemplo siguiente.

Visual Basic

```
' Assembly marked as compliant.
<Assembly: CLSCompliant(True)>
' Class marked as compliant.
<CLSCompliant(True)> Public Class MyCompliantClass
   ' Method marked as not compliant.
   <CLSCompliant(False)> Public Sub ChangeValue(value As UInt32)
   End Sub
   Public Shared Sub Main()
      Dim i As Integer = 2
      Console.WriteLine(i)
   End Sub
End Class
```

C#

```
using System;
// Assembly marked as compliant.
[assembly: CLSCompliantAttribute(true)]
// Class marked as compliant.
[CLSCompliantAttribute(true)]
public class MyCompliantClass {
   // Method marked as not compliant.
   [CLSCompliantAttribute(false)]
   public void ChangeValue(UInt32 value){ }
   public static void Main( ) {
   int i = 2;
   Console.WriteLine(i);
   }
}
```

Este código no genera ninguna advertencia del compilador. El resultado es 2

Ensamblados en Common Language Runtime

Los ensamblados son las unidades de creación de las aplicaciones .NET Framework; constituyen la unidad fundamental de implementación, control de versiones, reutilización, ámbitos de activación y permisos de seguridad. Un ensamblado es una colección de tipos y recursos creados para funcionar en conjunto y formar una unidad lógica de funcionalidad. Los ensamblados proporcionan a Common Language Runtime la información necesaria para conocer las implementaciones de tipos. Para el motor en tiempo de ejecución, un tipo no existe si no es en el contexto de un ensamblado.

Información General sobre Ensamblados

Los ensamblados son una parte fundamental de la programación con .NET Framework. Un ensamblado realiza las funciones siguientes:

- Contiene el código que ejecuta Common Language Runtime. El código del lenguaje intermedio de Microsoft (MSIL) de un archivo ejecutable portable (PE) no se ejecuta si no tiene asociado un manifiesto de ensamblado. Hay que tener en cuenta que cada ensamblado sólo puede tener un punto de entrada (es decir, **DllMain**, **WinMain** o **Main**).

- Crea un límite de seguridad. Un ensamblado es la unidad en la que se solicitan y conceden los permisos.

- Crea un límite de tipos. La identidad de cada tipo incluye el nombre del ensamblado en que reside. Por ello, un tipo MyType cargado en el ámbito de un ensamblado no es igual que un tipo denominado MyType cargado en el ámbito de otro ensamblado.

- Crea un límite de ámbito de referencia. El manifiesto del ensamblado contiene los metadatos del ensamblado que se utilizan para resolver tipos y satisfacer las solicitudes de recursos. Especifica los tipos y recursos que se exponen fuera del ensamblado. El manifiesto también enumera otros ensamblados de los que depende.

- Forma un límite de versión. El ensamblado es la unidad versionable más pequeña de Common Language Runtime; todos los tipos y recursos del mismo ensamblado pertenecen a la misma versión. El manifiesto del ensamblado describe las dependencias de versión que se especifiquen para los ensamblados dependientes.

- Crea una unidad de implementación. Cuando se inicia una aplicación, sólo deben estar presentes los ensamblados a los que llama la aplicación inicialmente. Los demás ensamblados, como los recursos de localización o los ensamblados que contengan clases de utilidad, se pueden recuperar a petición. De este modo, se puede mantener la simplicidad y transparencia de las aplicaciones la primera vez que se descargan.

- Es la unidad que permite la ejecución simultánea.

Los ensamblados pueden ser estáticos o dinámicos. Los ensamblados estáticos pueden incluir tipos de .NET Framework (interfaces y clases), así como recursos para el ensamblado (mapas de bits, archivos JPEG, archivos de recursos, etc.). Los ensamblados estáticos se almacenan en el disco, en archivos ejecutables portables PE. También se puede utilizar .NET Framework para crear ensamblados dinámicos,

que se ejecutan directamente desde la memoria y no se guardan en el disco antes de su ejecución. Los ensamblados dinámicos se pueden guardar en el disco una vez que se hayan ejecutado.

Existen varias formas de crear ensamblados. Puede utilizar herramientas de desarrollo, como Visual Studio 2005, que haya empleado anteriormente para crear archivos .dll o .exe. Puede utilizar las herramientas suministradas en Kit de desarrollo de software de Windows (SDK) para generar ensamblados con módulos creados en otros entornos de programación. También puede utilizar las API de Common Language Runtime, como Reflection.Emit, para crear ensamblados dinámicos.

Ventajas de los Ensamblados (Assemblies)

Los ensamblados están diseñados para simplificar la implementación de las aplicaciones y para solucionar los posibles problemas de versiones de las aplicaciones basadas en componentes.

El usuario final y los programadores conocen perfectamente los problemas de versiones e implementación que surgen hoy en día con los sistemas basados en componentes. Algunos usuarios finales se han visto frustrados al instalar una nueva aplicación en su equipo y ver cómo deja de funcionar otra aplicación que ya tenían instalada. Muchos programadores han estado horas y horas intentando mantener la coherencia de todas las entradas del Registro necesarias para activar una clase COM.

Con el uso de ensamblados en .NET Framework se han resuelto muchos problemas de implementación. Debido a que son componentes autodescriptivos que no dependen de las entradas del Registro, los ensamblados permiten instalar las aplicaciones sin problemas. También simplifican la desinstalación y replicación de las aplicaciones.

⊟ Problemas de versiones

Actualmente existen dos problemas de versiones con las aplicaciones de Win32:

- No se pueden expresar las reglas de las versiones entre las partes de una aplicación y que las imponga el sistema operativo. El planteamiento actual se basa en la compatibilidad con versiones anteriores, que a menudo es difícil de garantizar. Las definiciones de interfaz deben ser estáticas, una vez publicadas, y un solo fragmento de código debe mantener la compatibilidad con las versiones anteriores. Además, el código normalmente se diseña de manera que en cualquier momento pueda existir y ejecutarse una sola versión del mismo en un equipo.

- No hay ninguna forma de mantener la coherencia entre conjuntos de componentes integrados y el conjunto que está presente en tiempo de ejecución.

Cuando estos dos problemas de versiones se combinan, generan conflictos de DLL que consisten en que, al instalar una aplicación, se puede interrumpir accidentalmente otra aplicación existente porque un componente de software o una DLL instalados no eran totalmente compatibles con la versión anterior. Cuando se llega a esta situación, el sistema no tiene forma de diagnosticar y reparar el problema.

⊟ El fin de los conflictos de DLL

En Microsoft® Windows® 2000 se dieron los primeros pasos para erradicar estos problemas. Incluye dos funciones que permiten solucionar parcialmente los conflictos de DLL:

- Windows 2000 permite crear aplicaciones cliente donde los archivos .dll dependientes se sitúan en el mismo directorio que el archivo .exe de la aplicación. Windows 2000 se puede configurar para buscar un componente en el directorio donde se encuentra el archivo .exe antes de comprobar la ruta de acceso completa o buscar la ruta de acceso normal. De esta manera, los componentes pueden ser independientes de los componentes ya instalados y utilizados por otras aplicaciones.
- Windows 2000 bloquea los archivos que se incluyen en el sistema operativo en el directorio System32 de manera que no se puedan reemplazar accidentalmente al instalar aplicaciones.

El uso de los ensamblados de Common Language Runtime es un paso más hacia la solución completa de los conflictos de DLL.

⊟ La solución de los ensamblados

Con el objetivo de solucionar los problemas de las versiones, así como los problemas restantes que desembocan en conflictos de DLL, el motor en tiempo de ejecución utiliza ensamblados para los siguientes fines:

- Permitir a los programadores especificar reglas de versiones entre distintos componentes de software.
- Proporcionar la infraestructura para que se cumplan las reglas de versiones.
- Proporcionar la infraestructura que permita que varias versiones de un componente se puedan ejecutar simultáneamente, lo que se conoce como ejecución simultánea.

Contenido de los Ensamblados

En general, un ensamblado estático está formado por cuatro elementos:

- El manifiesto del ensamblado, que contiene los metadatos del ensamblado.
- Los metadatos de tipos.
- El código de lenguaje intermedio de Microsoft (MSIL) que implementa los tipos.
- Un conjunto de recursos.

El manifiesto de ensamblado es el único elemento obligatorio, pero se necesitan o bien los tipos o bien los recursos para proporcionar al ensamblado una funcionalidad significativa.

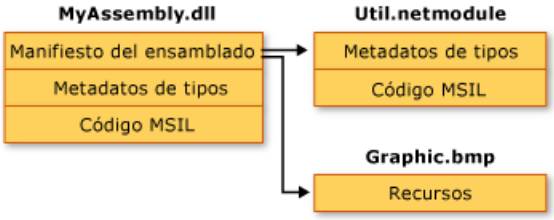
Estos elementos se pueden agrupar en un ensamblado de varias formas. Se puede agrupar todos los elementos en un solo archivo físico, como se observa en la ilustración siguiente.

Ensamblado de único archivo

Alternativamente, los elementos de un ensamblado se pueden incluir en varios archivos. Estos archivos pueden ser módulos de código compilado (.netmodule), recursos (como archivos .bmp o .jpg) u otros archivos que requiera la aplicación. Puede crear un ensamblado de múltiples archivos para combinar módulos escritos en idiomas diferentes y optimizar la descarga de una aplicación al colocar los tipos que rara vez se utilizan en un módulo que se descargue sólo cuando sea necesario.

En la siguiente ilustración, el programador de una aplicación hipotética ha decidido separar el código de alguna utilidad en un módulo diferente y mantener un archivo de recursos grande (en este caso, una imagen .bmp) en su archivo original. .NET Framework descarga un archivo sólo cuando se hace referencia al mismo, por lo que la descarga de código se ve optimizada cuando se mantiene el código al que no se hace referencia frecuentemente en un archivo aparte de la aplicación.

Ensamblado de múltiples archivos



Nota:

El sistema de archivos no vincula físicamente los archivos que forman un ensamblado de múltiples archivos. En su lugar, se vinculan a través del manifiesto del ensamblado y Common Language Runtime los administra como una unidad.

En esta ilustración, los tres archivos pertenecen a un ensamblado, como se describe en el manifiesto del ensamblado que contiene MyAssembly.dll. Para el sistema de archivos se trata de tres archivos independientes. Tenga en cuenta que el archivo Util.netmodule se compiló como un módulo porque no contiene información de ensamblado. Cuando se creó el ensamblado, se agregó el manifiesto del ensamblado a MyAssembly.dll, indicando su relación con Util.netmodule y Graphic.bmp.

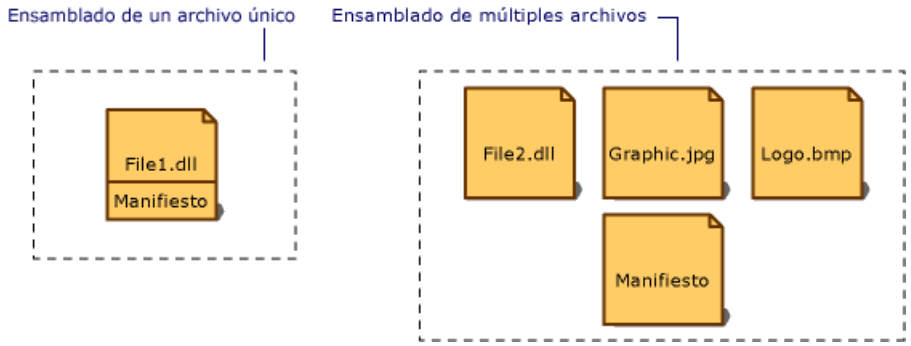
Actualmente, al diseñar el código fuente, se toman decisiones explícitas sobre cómo repartir la funcionalidad de una aplicación entre uno o más archivos. Al diseñar el código de .NET Framework, tomará decisiones similares sobre cómo dividir la funcionalidad en uno o más ensamblados.

Manifiesto del Ensamblado

Todos los ensamblados, ya sean estáticos o dinámicos, contienen una colección de datos que describen cómo se relacionan entre sí los elementos del ensamblado. El manifiesto contiene estos metadatos del ensamblado. Un manifiesto de ensamblado contiene todos los metadatos necesarios para especificar los requisitos de versión y la identidad de seguridad del ensamblado, y todos los metadatos necesarios para definir el ámbito del ensamblado y resolver las referencias a los recursos y las clases. El manifiesto del ensamblado se puede almacenar en un archivo PE (.exe o .dll) con código de lenguaje intermedio de Microsoft (MSIL) o en un archivo PE independiente que contenga sólo la información sobre el manifiesto del ensamblado.

En la siguiente ilustración se muestran las distintas formas de almacenar un manifiesto.

Tipos de ensamblados



En el caso de un ensamblado que tenga un solo archivo asociado, el manifiesto se incluye en el archivo PE para formar un ensamblado de un solo archivo. Se puede crear un ensamblado de múltiples archivos con un archivo de manifiesto independiente o incorporando el manifiesto en uno de los archivos PE del ensamblado.

Un manifiesto de ensamblado realiza las funciones siguientes:

- Enumera los archivos que componen el ensamblado.
- Controla cómo se asignan las referencias a los tipos y recursos del ensamblado a los archivos que contienen sus declaraciones e implementaciones.
- Enumera otros ensamblados de los que depende este ensamblado.
- Proporciona un nivel de direccionamiento indirecto entre los consumidores del ensamblado y los detalles de implementación del ensamblado.
- Convierte el ensamblado en autodescriptivo.

⊟ Contenido del manifiesto del ensamblado

En la tabla siguiente, se muestra la información que contiene el manifiesto del ensamblado. Los cuatro primeros elementos (nombre del ensamblado, número de versión, referencia cultural e información sobre el nombre seguro) constituyen la identidad del ensamblado.

Información	Descripción
Nombre del ensamblado	Cadena de texto donde se especifica el nombre del ensamblado.
Número de versión	Número de versión principal y secundaria, y número de revisión y de versión de compilación. Common Language Runtime utiliza estos números para exigir las directivas de versión.
Referencia cultural	Información sobre la referencia cultural o idioma que admite el ensamblado. Esta información se debe utilizar sólo para designar un ensamblado como ensamblado satélite que contiene información específica sobre la referencia cultural o el idioma. (Se asume que un ensamblado con información de referencia cultural es un ensamblado satélite).
Información sobre el nombre seguro	Clave pública del editor si el ensamblado tiene un nombre seguro
Lista de todos los archivos del ensamblado	Un código hash de cada archivo que contiene el ensamblado y nombre de archivo . Tenga en cuenta que todos los archivos que componen el ensamblado deben encontrarse en el mismo directorio que el archivo que contiene el manifiesto de ensamblado.
Información de referencia de tipos	Información que utiliza el motor en tiempo de ejecución para asignar una referencia de tipos al archivo que contiene su declaración e implementación. Se utiliza para tipos que se exportan desde el ensamblado.
Información sobre ensamblados a los que se hace referencia	Lista de otros ensamblados a los que este ensamblado hace referencia estáticamente. Cada referencia incluye el nombre del ensamblado dependiente, los metadatos del ensamblado (versión, referencia cultural, sistema operativo, etc.) y la clave pública, si el ensamblado tiene un nombre seguro.

Si desea agregar o modificar información en el manifiesto del ensamblado, puede utilizar atributos de ensamblado en su código. Se puede cambiar la información de versión y los atributos informativos, como Trademark, Copyright, Product, Company e Informational Version.

Caché de Ensamblados Global (GAC)

Cada equipo donde se instala Common Language Runtime tiene una memoria caché de código denominada caché de ensamblados global. La caché de ensamblados global almacena los ensamblados designados específicamente para ser compartidos por varias aplicaciones del equipo.

Se recomienda compartir los ensamblados mediante su instalación en la caché de ensamblados global sólo cuando sea necesario. Como norma general, mantenga las dependencias de los ensamblados privadas y coloque los ensamblados en el directorio de la aplicación, a menos que sea explícitamente necesario compartir un ensamblado en concreto. Además, no es necesario instalar los ensamblados en la caché de ensamblados global para que obtenga acceso a ellos el código de interoperabilidad COM o el código no administrado.

Nota:

Habrá algunos escenarios en los que no desee instalar un ensamblado en la caché de ensamblados global. Si coloca uno de los ensamblados que componen una aplicación en la memoria caché de ensamblados global, no podrá replicar ni instalar la aplicación utilizando el comando **xcopy** para copiar el directorio de la aplicación. También debe mover el ensamblado en la caché de ensamblados global.

Existen varias formas de implementar un ensamblado en la caché de ensamblados global:

- Usar un instalador diseñado para funcionar con la caché de ensamblados global. Es la opción preferida para instalar ensamblados en la caché de ensamblados global.
- Utilice la herramienta de desarrollador Caché de ensamblados global (Gacutil.exe), que se suministra con Kit de desarrollo de software de Windows (SDK).
- Usar el Explorador de Windows para incluir ensamblados en la caché.

Nota:

En escenarios de implementación, se recomienda utilizar Windows Installer 2.0 para instalar los ensamblados en la caché de ensamblados global. Utilice el Explorador de Windows o la herramienta Caché de ensamblados global sólo en escenarios de programación, porque no proporcionan funciones de recuento de referencias de ensamblados y otras funciones que se incluyen con Windows Installer.

Con frecuencia, los administradores protegen el directorio systemroot con una lista de control de acceso (ACL) para controlar el acceso de escritura y ejecución. Puesto que la caché de ensamblados global está instalada en un subdirectorio del directorio systemroot, hereda la lista (ACL) de dicho directorio. Es recomendable que sólo puedan eliminar archivos de la caché de ensamblados global los usuarios que tengan privilegios de administrador.

Los ensamblados implementados en la caché de ensamblados global deben tener nombres seguros. Cuando se agrega un ensamblado a la caché de ensamblados global, se realizan comprobaciones de integridad de todos los archivos que componen el ensamblado. La caché realiza estas comprobaciones de integridad para garantizar que no se ha manipulado ningún ensamblado, por ejemplo, cuando se ha modificado un archivo pero el manifiesto no refleja el cambio.

Ensamblados con Nombre Seguro (Strong Name)

Un nombre seguro está formado por la identidad del ensamblado (nombre de texto sencillo, número de versión e información de referencia cultural, si se proporciona), además de una clave pública y una firma digital. Se genera a partir de un archivo del ensamblado (el archivo que contiene el manifiesto del ensamblado, que, a su vez, contiene los nombres y códigos hash de todos los archivos que componen el ensamblado) mediante la clave privada correspondiente. Microsoft® Visual Studio® .NET y otras herramientas de programación que se incluyen en Kit de desarrollo de software de Windows (SDK)

pueden asignar nombres seguros a un ensamblado. Los ensamblados con el mismo nombre seguro tienen que ser idénticos.

Para garantizar que un nombre es exclusivo globalmente, firme el ensamblado con un nombre seguro. En concreto, los nombres seguros cumplen los siguientes requisitos:

- Garantizan la exclusividad del nombre al basarse en pares de claves únicas. Nadie puede generar un nombre de ensamblado igual, porque un ensamblado generado con una clave privada tiene un nombre distinto al de un ensamblado generado con otra clave privada.
- Protegen la procedencia de la versión de un ensamblado. Un nombre seguro garantiza que otra persona no puede crear una versión posterior de un ensamblado. Los usuarios pueden estar seguros de que una versión del ensamblado que están cargando procede del mismo editor que creó la versión con la que se construyó la aplicación.
- Proporcionan una comprobación de integridad importante. Al pasar las comprobaciones de seguridad de .NET Framework, se garantiza que el contenido del ensamblado no se ha modificado desde que se construyó. Sin embargo, tenga presente que los nombres seguros no implican de por sí un nivel de confianza como el que proporcionan, por ejemplo, una firma digital y un certificado.

Cuando se hace referencia a un ensamblado con nombre seguro, se espera obtener ciertas ventajas como la protección de las versiones y los nombres. Si el ensamblado con nombre seguro hace referencia a un ensamblado con nombre sencillo, que no tiene estas ventajas, se pierden tales ventajas y se vuelve a los conflictos de DLL. Por tanto, los ensamblados con nombre seguro sólo pueden hacer referencia a otros ensamblados con nombre seguro.

Consideraciones de Seguridad sobre Ensamblados

- Cuando se construye un ensamblado, se puede especificar el conjunto de permisos que son necesarios para la ejecución del mismo. La concesión de permisos específicos para un ensamblado se basa en la evidencia.

La evidencia se utiliza de dos formas distintas:

- La evidencia de entrada se combina con la evidencia recopilada por el cargador para crear un juego final de evidencias que se utiliza en la resolución de directivas. Entre los métodos que utilizan esta semántica se incluyen: **Assembly.Load**, **Assembly.LoadFrom** y **Activator.CreateInstance**.
- La evidencia de entrada se utiliza sin modificaciones como el conjunto final de evidencias utilizadas en la resolución de directivas. Entre los métodos que utilizan esta semántica se incluyen: **Assembly.Load(byte[])** y **AppDomain.DefineDynamicAssembly()**.

Se pueden conceder permisos opcionales mediante la directiva de seguridad establecida en el equipo donde se ejecutará el ensamblado. Si desea que su código controle todas las posibles excepciones de seguridad, puede:

- Insertar una solicitud de permiso para todos los permisos que deba tener su código y controlar por adelantado los errores que se puedan producir en tiempo de carga si no se conceden tales permisos.
- No utilizar una solicitud de permiso para obtener los permisos que pueda necesitar su código, pero estar preparado para controlar las excepciones de seguridad que se pueden producir si no se conceden los permisos.

Nota:
La seguridad es una cuestión compleja, con muchas opciones posibles entre las que elegir.

En el momento de la carga, se utiliza la evidencia del ensamblado como entrada para la directiva de seguridad. El administrador del equipo y la empresa y la configuración de directivas de usuario establecen la directiva de seguridad, que determina el conjunto de permisos que se concede a todo el código administrado cuando se ejecuta. La directiva de seguridad se puede establecer para la compañía del ensamblado (si tiene una firma generada utilizando la herramienta de firma), para el sitio y la zona Web (en términos de Internet Explorer) de los que se descargó el ensamblado o para el nombre seguro del ensamblado. Por ejemplo, el administrador de un equipo puede establecer una directiva de seguridad que permita que todo el código descargado desde un sitio Web y firmado por una compañía de software dada pueda tener acceso a una base de datos del equipo, pero no le otorga permiso para escribir en el disco del equipo.

⊟ Ensamblados con nombre seguro y herramientas de firma

Se puede firmar un ensamblado de dos formas diferentes y, a la vez, complementarias: con un nombre seguro o utilizando Herramienta Firma de archivos (Signcode.exe) en .NET Framework (versiones 1.0 y 1.1) o Herramienta Firma (SignTool.exe) en versiones posteriores de .NET Framework. Al firmar un ensamblado con un nombre seguro, se agrega un cifrado mediante clave pública al archivo que contiene el manifiesto del ensamblado. La firma mediante nombres seguros ayuda a comprobar la unicidad del nombre, impide la simulación de nombres y proporciona a los llamadores alguna identidad cuando se resuelve una referencia.

Sin embargo, no hay ningún nivel de confianza asociado a un nombre seguro, lo que hace que Herramienta Firma de archivos (Signcode.exe) y Herramienta Firma (SignTool.exe) adquieran un carácter importante. Las dos herramientas de firma requieren que una compañía de software demuestre su identidad a una autoridad de terceros y obtenga un certificado. Este certificado se incrusta en el archivo y el administrador puede utilizarlo para decidir si debe confiar en la autenticidad del código.

Es posible asignar un nombre seguro y una firma digital creadas mediante la utilización de Herramienta Firma de archivos (Signcode.exe) o Herramienta Firma (SignTool.exe) en un ensamblado, o se puede utilizar cualquiera de las dos opciones por separado. Ambas herramientas de firma sólo pueden firmar archivos de uno en uno; en el caso de un ensamblado de múltiples archivos, se firma el archivo que contiene el manifiesto de ensamblado. Se almacena un nombre seguro en el archivo que contiene el manifiesto de ensamblado, pero la firma creada mediante la utilización de Herramienta Firma de archivos (Signcode.exe) o Herramienta Firma (SignTool.exe) se almacena en una ranura reservada del archivo ejecutable portable (PE) que contiene el manifiesto. Se puede utilizar la firma de un ensamblado (con o sin nombre seguro) mediante Herramienta Firma de archivos (Signcode.exe) o Herramienta Firma

(SignTool.exe) cuando ya se disponga de una jerarquía de confianza que se base en firmas generadas de Herramienta Firma de archivos (Signcode.exe) o Herramienta Firma (SignTool.exe) o cuando una determinada directiva utilice sólo la parte de la clave y no compruebe una cadena de confianza.

Nota:

Cuando se utilicen tanto un nombre seguro como una firma de la herramienta de firma en un ensamblado, primero se debe asignar el nombre seguro.

Common Language Runtime también realiza una comprobación de código hash; el manifiesto del ensamblado contiene una lista de todos los archivos que componen el ensamblado, incluido el código hash de cada archivo que existía cuando se construyó el manifiesto. A medida que se carga cada archivo, se extrae el hash de su contenido y se compara con el valor hash almacenado en el manifiesto. Si los dos valores no coinciden, el ensamblado no se carga.

Debido a que los nombres seguros y las firmas que utilizan Herramienta Firma de archivos (Signcode.exe) o Herramienta Firma (SignTool.exe) garantizan la integridad, se puede basar la directiva de seguridad de acceso a código en estas dos formas de evidencia de ensamblado. Los nombres seguros y las firmas que utilizan Herramienta Firma de archivos (Signcode.exe) o Herramienta Firma (SignTool.exe) garantizan la integridad mediante firmas digitales y certificados. Todas las tecnologías mencionadas (comprobación de código hash, nombres seguros y firmas que utilizan Herramienta Firma de archivos (Signcode.exe) o Herramienta Firma (SignTool.exe)) se combinan para garantizar que el ensamblado no ha sufrido ninguna alteración.

Versiones de los Ensamblados

La creación de versiones de ensamblados mediante Common Language Runtime se realiza en el nivel de ensamblado. La versión específica de un ensamblado y las versiones de los ensamblados dependientes se guardan en el manifiesto del ensamblado. La directiva de versiones predeterminada para el motor en tiempo de ejecución es que las aplicaciones se ejecuten sólo en las versiones con las que se crearon y comprobaron, a menos que se reemplace con una directiva de versiones explícita en los archivos de configuración (el archivo de configuración de la aplicación, el archivo de directivas de la compañía de software y el archivo de configuración del administrador del equipo).

Nota:

La creación de versiones sólo se realiza en ensamblados con nombres seguros.

El motor en tiempo de ejecución ejecuta varios pasos para resolver la solicitud de enlace de un ensamblado:

1. Comprueba la referencia del ensamblado original para determinar la versión del ensamblado que se va a enlazar.

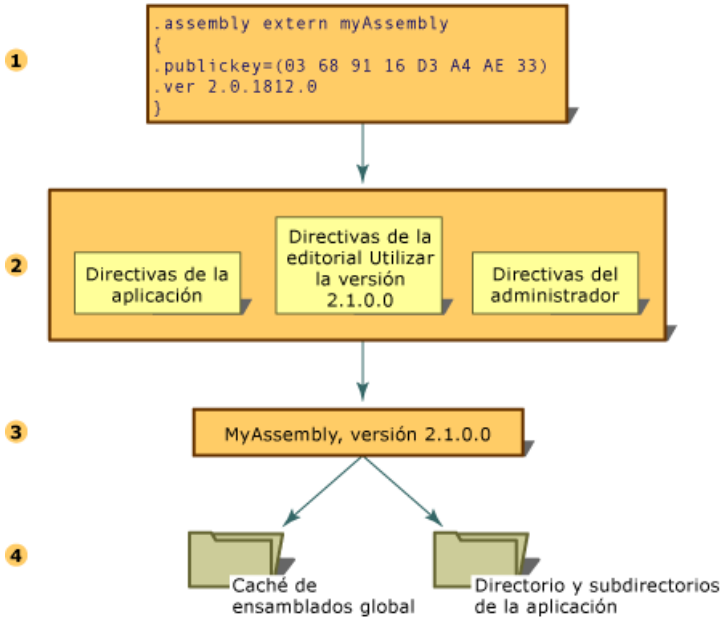
2. Comprueba todos los archivos de configuración correspondientes para aplicar la directiva de versiones.

3. Determina el ensamblado correcto a partir de la referencia del ensamblado original y las posibles redirecciones especificadas en los archivos de configuración, y determina la versión que debería enlazarse al ensamblado que realiza la llamada.

4. Comprueba la caché de ensamblados global, el código base especificado en los archivos de configuración y, después, comprueba el directorio y los subdirectorios de la aplicación mediante las reglas de búsqueda que se describen en Cómo el motor en tiempo de ejecución ubica ensamblados.

La ilustración siguiente muestra estos pasos.

Resolver la solicitud de enlace de un ensamblado



Información de versiones

Cada ensamblado tiene dos maneras distintas de expresar la información de versión:

- El número de versión del ensamblado, que, junto con el nombre del ensamblado y la información de referencia cultural, es parte de la identidad del ensamblado. El motor en tiempo de ejecución utiliza este número para imponer la directiva de versiones y juega un papel importante en el proceso de resolución de tipos en tiempo de ejecución.

- Una versión informativa, que es una cadena que representa información adicional sobre la versión que se incluye sólo con carácter informativo.

Número de versión del ensamblado

Cada ensamblado tiene un número de versión como parte de su identidad. Por tanto, el motor en tiempo de ejecución considera que son totalmente diferentes dos ensamblados que se diferencien por el número de versión. El número de versión se representa físicamente como una cadena de cuatro partes con el formato siguiente:

<versión principal>.<versión secundaria>.<número de versión de compilación>.<revisión>

Por ejemplo, la versión 1.5.1254.0 indica que 1 es la versión principal, 5 es la versión secundaria, 1254 es el número de la versión de compilación y 0 es el número de revisión.

El número de versión se almacena en el manifiesto del ensamblado junto con otra información de identidad, incluidos el nombre del ensamblado y la clave pública, así como información sobre las relaciones e identidades de otros ensamblados relacionados con la aplicación.

Cuando se construye un ensamblado, la herramienta de programación registra la información de dependencia de cada ensamblado al que se haga referencia en el manifiesto. El motor en tiempo de ejecución utiliza estos números de versión, junto con la información de configuración establecida por un administrador, una aplicación o una compañía de software, para cargar la versión correcta de un ensamblado al que se hace referencia.

El motor en tiempo de ejecución distingue los ensamblados normales de los ensamblados con nombres seguros para crear las versiones. La comprobación de versión sólo se produce para los ensamblados con nombres seguros.

Versión informativa del ensamblado
La versión informativa es una cadena que asocia a un ensamblado información adicional sobre la versión sólo con carácter informativo; esta información no se utiliza en tiempo de ejecución. La versión informativa basada en texto corresponde a la literatura de marketing del producto, al paquete o al nombre del producto, y no la utiliza el motor en tiempo de ejecución. Por ejemplo, una versión informativa podría ser "Common Language Runtime version 1.0" o "NET Control SP 2". En la ficha Versión del cuadro de diálogo de propiedades del archivo en Microsoft Windows, esta información aparece en el elemento "Product Version".

Nota:
Aunque es posible especificar cualquier texto, aparecerá un mensaje de advertencia durante la compilación si la cadena no tiene el formato utilizado por el número de versión del ensamblado o si, teniendo ese formato, contiene comodines. Esta advertencia es inofensiva.

La versión informativa se representa utilizando el atributo System.Reflection..::.AssemblyInformationalVersionAttribute personalizado.

Colocación de Ensamblados

En la mayoría de las aplicaciones .NET Framework, los ensamblados que componen una aplicación se colocan en el directorio de la aplicación, en un subdirectorio de este último o en la caché de ensamblados global (si el ensamblado está compartido). Para reemplazar la ubicación en la que Common Language Runtime busca un ensamblado, utilice Elemento <codeBase> en un archivo de configuración. Si el ensamblado no tiene un nombre seguro, la ubicación especificada mediante Elemento <codeBase> se limita al directorio de la aplicación o a un subdirectorio de éste. Si el ensamblado tiene un nombre seguro, Elemento <codeBase> puede especificar cualquier ubicación en el equipo o la red.

Se aplican reglas similares a la búsqueda de ensamblados cuando se trabaja con código no administrado o aplicaciones de interoperabilidad COM: si el ensamblado va a compartirse entre varias aplicaciones,

debe instalarse en la caché de ensamblados global. Los ensamblados que se utilizan con código no administrado deben exportarse como biblioteca de tipos y registrarse. Los ensamblados utilizados por la interoperabilidad COM deben registrarse en el catálogo, aunque, en algunos casos, este registro es automático.

Ensamblados y Ejecución Simultánea

La ejecución simultánea es la capacidad de almacenar y ejecutar varias versiones de una aplicación o un componente en el mismo equipo. Esto significa que puede tener varias versiones del motor en tiempo de ejecución y varias versiones de aplicaciones y componentes que utilizan una misma versión del motor en tiempo de ejecución en el mismo equipo y a la vez. La ejecución simultánea ofrece un mayor control sobre las versiones de un componente a las que se enlaza una aplicación, y sobre la versión del motor en tiempo de ejecución que utiliza una aplicación.

El hecho de que se admita el almacenamiento y la ejecución simultáneos de distintas versiones del mismo ensamblado es una parte integral de la creación de nombres seguros incluida en la infraestructura del motor en tiempo de ejecución. Debido a que el número de versión del ensamblado con nombre seguro forma parte de su identidad, el motor en tiempo de ejecución puede almacenar múltiples versiones del mismo ensamblado en la caché de ensamblados global y cargar esos ensamblados en tiempo de ejecución.

Aunque el motor en tiempo de ejecución permite crear aplicaciones simultáneas, la ejecución simultánea no es automática.

Dominios de Aplicación (Application Domains)

Normalmente, los sistemas operativos y los entornos de Common Language Runtime proporcionan algún tipo de aislamiento entre las aplicaciones. Por ejemplo, Microsoft Windows utiliza los procesos para aislar las aplicaciones. Este aislamiento es necesario para garantizar que el código que se ejecuta en una aplicación no afecta negativamente a otras aplicaciones no relacionadas.

Los dominios de aplicación proporcionan un límite de aislamiento para la seguridad, confiabilidad y control de versiones, así como para descargar los ensamblados. Los dominios de aplicación suelen ser creados por hosts de motor en tiempo de ejecución, que son los responsables de cargar automáticamente Common Language Runtime antes de que se ejecute una aplicación.

En esta sección se explica cómo utilizar los dominios de aplicación para obtener el aislamiento entre los ensamblados.

Información General sobre Dominios de Aplicación

Tradicionalmente se han utilizado límites de proceso para aislar las aplicaciones que se ejecutan en un mismo equipo. Cada aplicación se carga en un proceso independiente que aísla la aplicación de las demás que se estén ejecutando en el mismo equipo.

Las aplicaciones se aíslan porque las direcciones de memoria son específicas de cada proceso; un puntero de memoria pasado de un proceso a otro no se puede utilizar de ninguna manera coherente en el proceso de destino. Tampoco se pueden realizar llamadas directas entre dos procesos. En su lugar, se deben utilizar servidores proxy, que proporcionan un nivel de direccionamiento indirecto.

El código administrado debe pasar por un proceso de verificación para poder ejecutarse (a menos que el administrador haya concedido permiso para omitir la comprobación). El proceso de comprobación determina si el código puede intentar el acceso a direcciones de memoria no válidas o realizar alguna otra acción que pudiera hacer que el proceso en el que se ejecuta deje de funcionar correctamente. Cuando el código pasa la prueba de comprobación, se dice que tiene seguridad de tipos. La posibilidad de comprobar la seguridad de tipos del código permite que Common Language Runtime proporcione un gran nivel de seguridad respecto a los límites del proceso, con un costo de rendimiento mucho menor.

Los dominios de aplicación constituyen una unidad de procesamiento más segura y versátil que puede utilizar Common Language Runtime para proporcionar el aislamiento entre las aplicaciones. En un solo proceso se pueden ejecutar varios dominios de aplicación con el mismo nivel de aislamiento que existiría en procesos independientes, sin incurrir en la sobrecarga adicional que supone realizar llamadas entre procesos o cambiar de un proceso a otro. La posibilidad de ejecutar múltiples aplicaciones en un solo proceso aumenta la escalabilidad del servidor de manera importante.

Aislar las aplicaciones es también importante para la seguridad de las mismas. Por ejemplo, en un solo proceso de explorador se pueden ejecutar controles de varias aplicaciones Web de tal forma que no puedan tener acceso a los datos y recursos de los demás controles.

Éstas son las ventajas del aislamiento que ofrecen los dominios de aplicación:

- Los errores de una aplicación no pueden afectar a otras aplicaciones. Debido a que el código seguro no puede generar problemas de memoria, el uso de dominios de aplicación garantiza que el código que se ejecute en un dominio no afectará a las demás aplicaciones del proceso.
- Es posible detener aplicaciones concretas sin detener todo el proceso. El uso de dominios de aplicación permite descargar el código que se ejecuta en una sola aplicación.

> **Nota:**
>
> No se puede descargar ensamblados o tipos por separado. Sólo se puede descargar un dominio completo.

- El código que se ejecuta en una aplicación no puede tener acceso directo al código o a los recursos de otra aplicación. Common Language Runtime impone este aislamiento al impedir que se realicen llamadas directas entre objetos de dominios de aplicación diferentes. Los objetos que se pasan entre dominios se copian o se obtiene acceso a ellos mediante proxy. Si el objeto se copia, la llamada al objeto es local. En otras palabras, el llamador y el objeto al que se hace referencia se encuentran en el mismo dominio de aplicación. Si se tiene acceso al objeto a través de un proxy, la llamada al objeto es remota. En este caso, el llamador y el objeto al que se hace referencia se encuentran en dominios de aplicación diferentes. En las llamadas entre dominios se utiliza la misma infraestructura de llamada remota que en las llamadas entre dos procesos o entre dos equipos. En consecuencia, los metadatos del objeto al que se hace referencia deben estar disponibles para ambos dominios de aplicación a fin de que la llamada al método no provoque un error en la compilación JIT. Si el dominio que llama no tiene acceso a los metadatos del objeto al que se está llamando, se podría producir un error de compilación con una excepción del tipo **System.IO.FileNotFound**. El objeto es quien decide el mecanismo para determinar cómo se puede obtener acceso a los objetos entre dominios.
- La aplicación en la que se ejecuta el código establece el comportamiento del mismo. En otras palabras, el dominio de aplicación proporciona valores de configuración tales como las directivas de versión de la aplicación, la ubicación de los ensamblados remotos a los que tiene acceso e información sobre dónde encontrar los ensamblados que se cargan en el dominio.
- El dominio de aplicación en el que se ejecuta el código puede controlar los permisos que se conceden al código.

Dominios de Aplicación y Ensamblados

En este tema se describe la relación entre los dominios de aplicación y los ensamblados. Debe cargar un ensamblado en un dominio de aplicación para poder ejecutar el código que contiene. Al ejecutar una aplicación típica, se cargan varios ensamblados en un dominio de aplicación.

El modo en que se carga un ensamblado determina si varios dominios de aplicación pueden compartir el código compilado Just-in-time (JIT) del ensamblado en el proceso y si el ensamblado se puede descargar del proceso.

- Si un ensamblado se carga con dominio neutro, todos los dominios de aplicación que comparten el mismo conjunto de permisos de seguridad pueden compartir el mismo código compilado JIT,

lo que reduce la cantidad de memoria que necesita la aplicación. Sin embargo, el ensamblado nunca se puede descargar del proceso.

- Si un ensamblado no se carga con dominio neutro, debe utilizarse la compilación JIT de ese ensamblado en los dominios de aplicación en que se carga. Sin embargo, el ensamblado se puede descargar del proceso; para ello, tendrán que descargarse todos los dominios de aplicación en que está cargado el ensamblado.

El host en tiempo de ejecución determina si los ensamblados se cargan con dominio neutro cuando se carga el motor en tiempo de ejecución en un proceso. En las aplicaciones administradas, aplique el atributo LoaderOptimizationAttribute al método de punto de entrada del proceso y especifique un valor de la enumeración LoaderOptimization asociada. En las aplicaciones no administradas que alojan Common Language Runtime, especifique el indicador adecuado cuando llame al método CorBindToRuntimeEx (Función).

Existen tres opciones para cargar ensamblados neutrales respecto al dominio:

- SingleDomain no carga ensamblados con dominio neutro, a excepción de Mscorlib que siempre se carga con dominio neutro. Esta configuración se denomina dominio simple porque suele utilizarse cuando el host ejecuta una sola aplicación en el proceso.

- MultiDomain carga todos los ensamblados con dominio neutro. Use esta configuración cuando en el proceso haya varios dominios de aplicación que ejecutan el mismo código.

- MultiDomainHost carga con dominio neutro los ensamblados que tienen un nombre seguro si se han instalado junto con todas sus dependencias en la caché de ensamblados global. La carga y la compilación JIT de los demás ensamblados se realiza de forma independiente en cada dominio de aplicación y, por tanto, estos ensamblados pueden descargarse del proceso. Utilice esta configuración cuando ejecute más de una aplicación en el mismo proceso, o si tiene un grupo heterogéneo de ensamblados compartidos por varios dominios de aplicación y ensamblados que es necesario descargar del proceso.

El código compilado JIT no se puede compartir en los ensamblados que se cargan en la carga de ensamblado por contexto especificado por el usuario utilizando el método LoadFrom de la clase Assembly, o que se cargan a partir de imágenes que utilizan las sobrecargas del método Load que especifican matrices de bytes.

Los ensamblados que se han compilado en código nativo utilizando Generador de imágenes nativas (Ngen.exe) se pueden compartir entre dominios de aplicación si la primera vez que se cargaron en un proceso lo hicieron con dominio neutro.

El código compilado JIT del ensamblado que contiene el punto de entrada de la aplicación sólo se puede compartir si pueden hacerlo todas sus dependencias.

Un ensamblado con dominio neutro puede someterse a la compilación JIT varias veces. Por ejemplo, cuando los conjuntos de permisos de seguridad de dos dominios de aplicación son diferentes, no pueden compartir el mismo código compilado JIT. Sin embargo, cada copia del ensamblado objeto de compilación JIT se puede compartir con otros dominios de aplicación que tengan el mismo conjunto de permisos de seguridad.

A la hora de determinar si va a cargar los ensamblados con dominio neutro, deberá decidir si prefiere reducir el consumo de memoria u otros factores relativos al rendimiento.

- El acceso a los métodos y datos estáticos es más lento en los ensamblados de dominio neutro porque es necesario aislar los ensamblados. Cada dominio de aplicación que tiene acceso al ensamblado debe disponer de una copia independiente de los datos estáticos para impedir que las referencias a objetos en los campos estáticos atraviesen los límites del dominio. Como resultado, el motor en tiempo de ejecución contiene lógica adicional para dirigir un llamador a la copia correspondiente del método o los datos estáticos. Esta lógica adicional retarda la llamada.
- Todas las dependencias de un ensamblado deben estar presentes y cargarse cuando el ensamblado se carga con dominio neutro, ya que si una dependencia no se puede cargar con dominio neutro, impedirá también que el ensamblado se cargue con dominio neutro.

Dominios de Aplicación y Subprocesos

Un dominio de aplicación constituye un límite de aislamiento para la seguridad, el control de versiones, la confiabilidad y la descarga de código administrado. Los subprocesos son la herramienta del sistema operativo que utiliza Common Language Runtime para ejecutar código. En tiempo de ejecución, todo el código administrado se carga en un dominio de aplicación y se ejecuta mediante un subproceso administrado.

No existe una correlación uno a uno entre los dominios de aplicación y los subprocesos. En un momento dado, se pueden ejecutar varios subprocesos en un solo dominio de aplicación y un subproceso determinado no está confinado a un solo dominio de aplicación. En otras palabras, los subprocesos pueden cruzar los límites del dominio de aplicación; no se crea un nuevo subproceso para cada dominio de aplicación.

En un momento dado, todos los subprocesos se ejecutan en un dominio de aplicación. Cero, uno, o más de un subproceso podrían estar ejecutándose en cualquier dominio de aplicación determinado. En tiempo de ejecución, se realiza un seguimiento de qué subprocesos se están ejecutando en qué dominios de aplicación. Se puede localizar en cualquier momento el dominio donde se está ejecutando un subproceso llamando al método GetDomain.

⊟ Dominios de aplicación y referencias culturales

Se puede asociar un objeto CultureInfo a un subproceso. Sin embargo, para evitar que entre código malintencionado en otros dominios de aplicación, el objeto CultureInfo se establece automáticamente en sólo lectura cuando su subproceso cruza el límite del dominio de aplicación.

Si se ha personalizado el objeto CultureInfo, por ejemplo, con un Calendar personalizado, se genera una excepción InvalidOperationException cuando el subproceso intenta cruzar el límite del dominio de aplicación.

Programar con Dominios de Aplicación

Normalmente, los dominios de aplicación son creados y manipulados mediante programación por los hosts de motor en tiempo de ejecución. Sin embargo, a veces, puede darse la circunstancia de que un programa de aplicación deba trabajar con dominios de aplicación. Por ejemplo, un programa de aplicación podría cargar un componente de aplicación en un dominio para poder descargar el dominio (y el componente) sin necesidad de detener toda la aplicación.

La clase AppDomain es la interfaz de programación para los dominios de aplicación. Esta clase incluye métodos para crear y descargar dominios, crear instancias de tipos en dominios y registrar diversas notificaciones, como por ejemplo cuando una aplicación descarga un dominio. En la tabla siguiente, se presentan los métodos de AppDomain más utilizados.

Método de AppDomain	Descripción
CreateDomain	Crea un nuevo dominio de aplicación. Se recomienda utilizar una sobrecarga de este método que especifique un objeto AppDomainSetup. Ésta es la forma recomendada para establecer las propiedades de un nuevo dominio, como la base de la aplicación o el directorio raíz para la aplicación; la ubicación del archivo de configuración para el dominio; y la ruta de búsqueda que va a utilizar Common Language Runtime para cargar ensamblados en el dominio.
ExecuteAssembly y ExecuteAssemblyByName	Ejecuta un ensamblado en el dominio de aplicación. Éste es un método de instancia, por lo que se puede utilizar para ejecutar el código en otro dominio de aplicación al que se haga referencia.
CreateInstanceAndUnwrap	Crea una instancia de un tipo especificado en el dominio de aplicación y devuelve un proxy. Utilice este método para evitar que se cargue el ensamblado que contiene el tipo creado en el ensamblado de llamada.
Unload	Cierra el dominio correctamente. El dominio de aplicación no se descarga hasta que todos los subprocesos que se están ejecutando en el dominio se hayan detenido o ya no se encuentren en el dominio.
Nota:	
Common Language Runtime no admite la serialización de métodos globales, por lo que no se pueden utilizar los delegados para ejecutar métodos globales en otros dominios de aplicación.	

Las interfaces no administradas que se describen en la especificación de las interfaces de alojamiento de Common Language Runtime también proporcionan acceso a los dominios de aplicación. Los hosts de motor en tiempo de ejecución pueden usar interfaces de código no administrado para crear y obtener acceso a los dominios de aplicación en un proceso.

Información General de la Biblioteca de Clases de .NET Framework (BCL)

.NET Framework incluye clases, interfaces y tipos de valores que agilizan y optimizan el proceso de desarrollo y proporcionan acceso a las funciones del sistema. Para facilitar la interoperabilidad entre lenguajes, la mayoría de los tipos de .NET Framework cumplen la especificación de lenguaje común (CLS) y, por tanto, se pueden utilizar en todos los lenguajes de programación cuyo compilador satisfaga los requisitos de CLS.

Los tipos de .NET Framework son la base sobre la que se crean aplicaciones, componentes y controles de .NET. .NET Framework incluye tipos que realizan las funciones siguientes:

- Representar tipos de datos base y excepciones.
- Encapsular estructuras de datos.
- Realizar E/S.
- Obtener acceso a información sobre tipos cargados.
- Invocar las comprobaciones de seguridad de .NET Framework.
- Proporcionar: acceso a datos, interfaz gráfica para el usuario (GUI) independiente de cliente e interfaz GUI de cliente controlada por el servidor.

.NET Framework proporciona un conjunto completo de interfaces, así como clases abstractas y concretas (no abstractas). Se pueden utilizar las clases concretas tal como están o, en muchos casos, derivar las clases propias de ellas. Para utilizar la funcionalidad de una interfaz se puede crear una clase que implemente la interfaz o derivar una clase de una de las clases de .NET Framework que implementa la interfaz.

⊟ Convenciones de nomenclatura

Los tipos de .NET Framework utilizan un esquema de nomenclatura con sintaxis de punto lo que indica la existencia de una jerarquía. Esta técnica agrupa tipos relacionados en espacios de nombres para que se pueda buscar y hacer referencia a ellos más fácilmente. La primera parte del nombre completo, hasta el punto situado más a la derecha, es el nombre del espacio de nombres. La última parte es el nombre de tipo. Por ejemplo, **System.Collections.ArrayList** representa el tipo **ArrayList** que pertenece al espacio de nombres **System.Collections**. Los tipos de **System.Collections** se pueden utilizar para manipular colecciones de objetos.

Este esquema de nomenclatura facilita a los programadores de bibliotecas la tarea de extender .NET Framework para poder crear grupos jerárquicos de tipos y asignarles nombre de forma coherente e ilustrativa. También permite identificar de forma inequívoca los tipos mediante su nombre completo (es decir, por su espacio de nombres y nombre de tipo), lo que evita que se produzcan conflictos entre los nombres de tipo. Se supone que los programadores de bibliotecas utilizarán la siguiente directriz cuando creen nombres para sus propios espacios de nombres:

NombreCompañía.NombreTecnología

Por ejemplo, el espacio de nombres Microsoft.Word cumple esta directriz.

El uso de modelos de nomenclatura para agrupar tipos relacionados en espacios de nombres es una forma muy útil de crear y documentar bibliotecas de clases. Sin embargo, este esquema de nomenclatura no influye en la visibilidad, el acceso a miembros, la herencia, la seguridad o el enlace. Se puede hacer la partición de un espacio de nombres en varios ensamblados y un ensamblado individual puede contener tipos de varios espacios de nombres. El ensamblado proporciona la estructura formal para el control de versiones, la implementación, la seguridad, la carga y la visibilidad en Common Language Runtime.

⊟ System (Espacio de nombres)

El espacio de nombres System es el espacio de nombres de la raíz de los tipos fundamentales de .NET Framework. Este espacio de nombres contiene clases que representan los tipos de datos base que se utilizan en todas las aplicaciones: Object (raíz de la jerarquía de herencia), Byte, Char, Array, Int32, String, etc. Muchos de estos tipos se corresponden con los tipos de datos primitivos que utiliza el lenguaje de programación. Cuando se escribe código utilizando tipos de .NET Framework se puede utilizar la palabra clave correspondiente del lenguaje cuando se espera un tipo de datos base de .NET Framework.

En la tabla siguiente se muestra una lista de los tipos base que proporciona .NET Framework, se describe brevemente cada tipo y se indica el tipo correspondiente de Visual Basic, C#, C++ y JScript.

Categoría	Nombre de la clase	Descripción	Tipo de datos en Visual Basic	Tipo de datos en C#	Tipo de datos de C++	Tipo de datos en JScript
Integer	Byte	Entero de 8 bits sin signo.	**Byte**	**byte**	**char**	**Byte**
	SByte	Entero de 8 bits con signo. No cumple CLS.	**SByte**	**sbyte**	**signed char**	**SByte**
	Int16	Entero de 16 bits con signo.	**Tipo Short**	**short**	**short**	**short**
	Int32	Entero de 32 bits con signo.	**Integer**	**int**	**int** O bien **long**	**int**
	Int64	Entero de 64 bits con signo.	**Tipo Long**	**long**	**__int64**	**long**
	UInt16	Entero de 16 bits sin signo. No cumple CLS.	**UShort**	**ushort**	**unsigned short**	**UInt16**
	UInt32	Entero de 32 bits sin signo. No cumple CLS.	**UInteger**	**uint**	**unsigned int** O bien **unsigned long**	**UInt32**
	UInt64	Entero de 64 bits sin	**ULong**	**ulong**	**unsigned**	**UInt64**

		signo. No cumple CLS.			**__int64**	
Punto flotante	Single	Número de punto flotante (32 bits) de precisión simple.	**Sencillo**	**float**	**float**	**float**
	Double	Número de punto flotante (64 bits) de doble precisión.	**Tipo Double**	**double**	**double**	**double**
Lógico	Boolean	Valor booleano (verdadero o falso).	**Booleano**	**bool**	**bool**	**bool**
Otros	Char	Carácter Unicode (16 bits).	**Tipo Char**	**char**	**wchar_t**	**char**
	Decimal	Valor decimal (128 bits).	**Decimal**	**decimal**	**Decimal**	**Decimal**
	IntPtr	Entero con signo cuyo tamaño depende de la plataforma subyacente (valor de 32 bits en una plataforma de 32 bits y valor de 64 bits en una plataforma de 64 bits).	**IntPtr** No dispone de un tipo integrado.	**IntPtr** No dispone de un tipo integrado.	**IntPtr** No dispone de un tipo integrado.	**IntPtr**
	UIntPtr	Entero sin signo cuyo tamaño depende de la plataforma subyacente (valor de 32 bits en una plataforma de 32 bits y valor de 64 bits en una plataforma de 64 bits). No cumple CLS.	**UIntPtr** No dispone de un tipo integrado.	**UIntPtr** No dispone de un tipo integrado.	**UIntPtr** No dispone de un tipo integrado.	**UIntPtr**
Objetos de clase	Object	Base de la jerarquía de objetos.	**Objecto**	**object**	**Object ***	**Objecto**
	String	Cadena inmutable de longitud fija de caracteres Unicode.	**Cadena**	**string**	**String***	**Cadena**

Además de los tipos de datos base, el espacio de nombres System contiene más de 100 clases, que comprenden desde las clases que controlan excepciones hasta las clases que tratan conceptos básicos en tiempo de ejecución, como los dominios de aplicación y el recolector de elementos no utilizados. El espacio de nombres System también contiene muchos espacios de nombres de segundo nivel.

Hosts del Motor en Tiempo de Ejecución

Common Language Runtime se ha diseñado para admitir distintos tipos de aplicaciones, desde aplicaciones de servidor Web hasta aplicaciones con una interfaz de usuario eficaz y tradicional de Windows. Cada tipo de aplicación requiere un host de motor en tiempo de ejecución que la inicie. El host de motor en tiempo de ejecución carga el motor en tiempo de ejecución en un proceso, crea los dominios de aplicación en el proceso y carga el código de usuario en los dominios de aplicación.

.NET Framework incluye varios hosts de motor en tiempo de ejecución, incluidos los de la tabla siguiente.

Host del motor en tiempo de ejecución	Descripción
ASP.NET	Carga el motor en tiempo de ejecución en el proceso que va a controlar la solicitud Web. ASP.NET crea también un dominio de aplicación para cada aplicación Web que se vaya a ejecutar en un servidor Web.
Microsoft Internet Explorer	Crea dominios de aplicación en los que ejecutar controles administrados. .NET Framework admite la descarga y ejecución de controles de explorador. El motor en tiempo de ejecución interactúa con el mecanismo de extensibilidad de Microsoft Internet Explorer a través de un filtro MIME para crear los dominios de aplicación donde se van a ejecutar los controles administrados. De manera predeterminada, se crea un dominio de aplicación para cada sitio Web.
Ejecutable del shell	Invoca el código que aloja el motor en tiempo de ejecución para transferir el control al motor en tiempo de ejecución cada vez que se inicia la ejecución de un archivo desde el shell.

Microsoft proporciona un conjunto de API para que escriba sus propios hosts de motor en tiempo de ejecución.

Lo Nuevo de .NET Framework Versión 3.5

Este tema contiene información sobre las características nuevas y mejoradas de .NET Framework versión 3.5.

⊟ .NET Compact Framework

.NET Compact Framework versión 3.5 amplía la compatibilidad con aplicaciones móviles distribuidas al incorporar la tecnología Windows Communication Foundation (WCF). También agrega nuevas características de lenguaje como LINQ, incluye nuevas API basadas en los comentarios de la comunidad y mejora la depuración con herramientas y características de diagnóstico actualizadas.

⊟ ASP.NET

.NET Framework 3.5 incorpora características mejoradas en áreas concretas de ASP.NET y Visual Web Developer. El avance más significativo es la mejora de la compatibilidad con el desarrollo de sitios web habilitados para AJAX. ASP.NET agrega compatibilidad con el desarrollo de AJAX centrado en el servidor mediante un conjunto de nuevos controles y nuevas API. Puede habilitar una página ASP.NET 2.0 existente en AJAX agregando un control ScriptManager y un control UpdatePanel, de modo que la página pueda actualizarse sin que sea necesario realizar una actualización de la página completa.

ASP.NET agrega también compatibilidad con el desarrollo de AJAX centrado en el cliente a través de una biblioteca de cliente denominada Microsoft AJAX Library. Microsoft AJAX Library es compatible con el desarrollo centrado en el cliente y orientado a objetos, que es independiente del explorador. Utilizando las clases de biblioteca de ECMAScript (JavaScript), puede habilitar comportamientos enriquecidos de la interfaz de usuario sin necesidad de realizar viajes de ida y vuelta al servidor (round trip). Puede combinar el grado de desarrollo que se centra en el servidor y en el cliente para satisfacer las necesidades de su aplicación. Por otro lado, Visual Web Developer mejora la compatibilidad de IntelliSense con JavaScript e incorpora la compatibilidad con Microsoft AJAX Library.

ASP.NET y Visual Web Developer admiten ahora la creación de servicios web basados en ASMX y WCF y hacen posible que se pueda utilizar sin problemas cualquier implementación de páginas web con Microsoft AJAX Library. Además, los servicios de aplicación del servidor, incluida la autenticación de formularios, la administración de funciones y los perfiles, se exponen ahora como servicios web que pueden utilizarse en aplicaciones compatibles con WCF, como el script de cliente y los clientes de formularios Windows Forms. ASP.NET permite que todas las aplicaciones basadas en web compartan estos servicios de aplicación comunes.

Otras mejoras que incluye ASP.NET son un nuevo control de datos, ListView, para mostrar los datos; un nuevo control de origen de datos, LinqDataSource, que expone Language Integrated Query (LINQ) a los desarrolladores web a través de las arquitecturas de controles de origen de datos de ASP.NET; una herramienta nueva, Herramienta Combinación de ASP.NET (Aspnet_merge.exe), para la combinación de ensamblados precompilados, y una estrecha integración con IIS 7.0. ListView es un control con un alto grado de personalización (utiliza plantillas y estilos) que también permite realizar operaciones de edición, inserción y eliminación, además de ofrecer funciones de ordenación y paginación. La funcionalidad de paginación de ListView se proporciona a través de un nuevo control denominado DataPager. La herramienta de combinación se puede utilizar para combinar ensamblados y agregar compatibilidad con un buen número de escenarios de administración de lanzamientos e implementación. La integración de ASP.NET y IIS 7.0 ofrece la posibilidad de utilizar los servicios de ASP.NET, como la autenticación y el almacenamiento en caché, con cualquier tipo de contenido. También ofrece la posibilidad de desarrollar

módulos de canalización de servidor en código administrado de ASP.NET y admite la configuración unificada de módulos y controladores.

Otras mejoras de Visual Web Developer incluyen la compatibilidad con varios destinos, la inclusión de proyectos de aplicaciones web, una nueva vista Diseño, nuevas herramientas de diseño de Hojas de estilos en cascada (CSS) y compatibilidad con LINQ en bases de datos de SQL. La compatibilidad con varios destinos permite utilizar Visual Web Developer para desarrollar aplicaciones web destinadas a versiones concretas de .NET Framework, incluidas las versiones 2.0, 3.0 y 3.5.

⊟ Complementos y extensibilidad

El ensamblado System.AddIn.dll de .NET Framework 3.5 proporciona un grado de compatibilidad eficaz y flexible a los programadores de aplicaciones extensibles. Introduce una nueva arquitectura y un nuevo modelo que ayudan a los programadores en las tareas preliminares al agregar extensibilidad a una aplicación y garantizar que sus extensiones siguen funcionando cuando la aplicación host cambia. El modelo proporciona las características siguientes:

- Detección

  Puede buscar y administrar con facilidad conjuntos de complementos en diversas ubicaciones de un equipo con la clase AddInStore. Puede utilizar esta clase para buscar y obtener información sobre los complementos mediante sus tipos base sin tener que cargarlos.

- Activación

  Una vez que una aplicación elige un complemento, la clase AddInToken facilita su activación. Sólo debe elegir un nivel de aislamiento y un recinto de seguridad, y el sistema se encargará de todo lo demás.

- Aislamiento

  La compatibilidad con dominios de aplicación y el aislamiento de procesos de complementos está integrada. El nivel de aislamiento de cada complemento depende del host. El sistema se ocupa de cargar los dominios de aplicación y los procesos y de cerrarlos una vez que sus complementos detienen su ejecución.

- Recintos de seguridad

  Resulta sencillo configurar los complementos con un nivel de confianza predeterminado o personalizado. Los conjuntos de permisos admitidos son los permisos de Internet, Intranet, plena confianza y el conjunto de permisos del host, así como las sobrecargas que permiten al host especificar un conjunto de permisos personalizados.

- Composición de la interfaz de usuario

  El modelo de complementos admite la composición directa de controles de Windows Presentation Foundation (WPF) que traspasan los límites del dominio de aplicación. Puede hacer que los complementos contribuyan directamente en la interfaz de usuario del host a la vez que mantiene los beneficios que suponen el aislamiento, la capacidad de descarga, los recintos de seguridad y el control de versiones.

- Control de versiones

  La arquitectura de los complementos hace posible que los hosts introduzcan nuevas versiones de su modelo de objetos sin interrumpir la compatibilidad con los complementos existentes y sin que esto afecte en modo alguno a la experiencia del desarrollador con los complementos nuevos.

⊟ Common Language Runtime

Colecciones

La clase HashSet<(Of <(T>)>) proporciona operaciones de conjuntos de alto rendimiento a .NET Framework. Un conjunto es una colección que no contiene ningún elemento duplicado y cuyos elementos no están ordenados de un modo determinado.

Diagnósticos

La clase EventSchemaTraceListener proporciona la traza de eventos conformes al esquema de un extremo a otro. Puede utilizar la traza de un extremo a otro en un sistema con componentes heterogéneos que atraviesan los límites de los subprocesos, de AppDomain, de los procesos y de los equipos. Se ha definido un esquema de eventos normalizado para habilitar la traza entre estos límites. Este esquema es compartido por varias tecnologías de traza, incluidas las herramientas de diagnóstico de Windows Vista, como el Visor de eventos. El esquema también permite agregar elementos personalizados conformes al esquema.

La clase EventSchemaTraceListener se ha adaptado para registrar el rendimiento a la vez que se mantiene compatibilidad implícita con la traza sin bloqueo.

E/S y canalizaciones

Las canalizaciones proporcionan comunicación entre procesos que se ejecutan en el mismo equipo o en cualquier otro equipo de Windows de una red. .NET Framework proporciona acceso a dos tipos de canalizaciones: las canalizaciones anónimas y las canalizaciones con nombre.

Recolección de elementos no utilizados

La clase GCSettings tiene una nueva propiedad LatencyMode que se puede utilizar para ajustar el momento en que el recolector de elementos no utilizados irrumpe en la aplicación. Esta propiedad se establece en uno de los valores de la nueva enumeración [System.Runtime.GCLatencyMode].

La clase GC tiene una nueva sobrecarga del método Collect(Int32, GCCollectionMode) que se puede utilizar para ajustar el comportamiento de una operación de recolección de elementos no utilizados forzada. Por ejemplo, puede utilizar esta sobrecarga para especificar que el recolector de elementos no utilizados debe determinar si el momento actual es el apropiado para reclamar los objetos. Esta sobrecarga toma un valor de la nueva enumeración GCCollectionMode.

Reflexión y emisión de reflexión con confianza parcial

Los ensamblados que se ejecutan con confianza parcial ahora pueden emitir código y ejecutarlo. El código emitido que sólo llama a tipos y métodos públicos no necesita ningún otro permiso además de los que solicitan los tipos y los métodos a los que tiene acceso. El nuevo constructor DynamicMethod(String, Type, array<Type>[]()[]) facilita la emisión de este tipo de código.

Cuando el código emitido necesita obtener acceso a datos privados, el nuevo constructor DynamicMethod(String, Type, array<Type>[]()[], Boolean) permite el acceso restringido. El host debe conceder permisos ReflectionPermission con el nuevo marcador RestrictedMemberAccess para habilitar

esta característica, que permite al código emitido obtener acceso a datos privados exclusivamente en aquellos tipos y métodos de los ensamblados que tienen un nivel de confianza equivalente o menor.

De igual forma, en la reflexión el host concede RestrictedMemberAccess, lo que permite utilizar de forma restringida métodos para obtener acceso a propiedades privadas, llamar a métodos privados, etc., pero sólo en ensamblados con un nivel de confianza equivalente o menor.

Subprocesos

Bloqueo de lectura y escritura mejorado
La nueva clase ReaderWriterLockSlim proporciona un rendimiento significativamente mejor que ReaderWriterLock y es comparable a la instrucción **lock** (**SyncLock** en Visual Basic). Las transiciones entre los estados de bloqueo se han simplificado para facilitar la programación y reducir la posibilidad de interbloqueo. La nueva clase admite la recursividad para simplificar la migración de **lock** y ReaderWriterLock.

Mejoras de rendimiento de ThreadPool
El rendimiento de la distribución de los elementos de trabajo y las tareas de E/S en el grupo de subprocesos administrados ha mejorado significativamente. La distribución se administra ahora en el código administrado, sin que se produzcan transiciones al código no administrado y con menos bloqueos. Es preferible utilizar ThreadPool que implementaciones del grupo de subprocesos administrados específicas de la aplicación.

Características mejoradas para las zonas horarias
Dos nuevos tipos, DateTimeOffset y TimeZoneInfo, mejoran la compatibilidad con las zonas horarias y facilitan el desarrollo de aplicaciones que trabajan con fechas y horas de diferentes zonas horarias.

TimeZoneInfo
La nueva clase TimeZoneInfo reemplaza con creces la funcionalidad de la clase TimeZone existente. Puede utilizar TimeZoneInfo para recuperar cualquier zona horaria definida en el Registro, y no sólo la zona horaria local y la hora universal coordinada (UTC). También puede utilizar esta clase para definir zonas horarias personalizadas, serializar y deserializar los datos de la zona horaria personalizada y convertir valores entre distintas zonas horarias.

DateTimeOffset
La nueva estructura DateTimeOffset amplía la estructura DateTime para facilitar el trabajo con valores de tiempo entre distintas zonas horarias. La estructura DateTimeOffset almacena información de fecha y hora como un valor de fecha y hora UTC junto con un valor de desfase que indica cuánto difiere ese valor de la hora UTC.

⊟ Criptografía

Manifiestos de ClickOnce
Existen nuevas clases de criptografía para la comprobación y obtención de información sobre las firmas de manifiestos de aplicaciones ClickOnce. La clase ManifestSignatureInformation obtiene información sobre una firma de manifiesto cuando se utiliza la sobrecarga de su método VerifySignature()()(). Puede utilizar la enumeración ManifestKinds para especificar los manifiestos que se van a comprobar. El resultado de la comprobación es uno de los valores de la enumeración SignatureVerificationResult. ManifestSignatureInformationCollection proporciona una colección de sólo lectura de los objetos

ManifestSignatureInformation de las firmas comprobadas. Además, las clases siguientes proporcionan información de firma específica:

- StrongNameSignatureVerification

  Contiene información sobre la firma de nombre seguro de un manifiesto.

- AuthenticodeSignatureInformation

  Representa la información sobre la firma Authenticode de un manifiesto.

- TimestampInformation

  Contiene información sobre la marca de tiempo de una firma Authenticode.

- TrustStatus

  Proporciona un mecanismo sencillo para comprobar si se confía en una firma Authenticode.

Compatibilidad con Suite B

.NET Framework 3.5 admite el conjunto de algoritmos criptográficos Suite B publicado por la Agencia de Seguridad Nacional (NSA).

Se incluyen los siguientes algoritmos:

- Estándar de cifrado avanzado (AES) con tamaños clave de cifrado de 128 y 256 bits.
- Algoritmo hash seguro (SHA-256 y SHA-384) para aplicar un algoritmo hash.
- Algoritmo de firma digital de curva elíptica (ECDSA) que utiliza curvas de módulos primos de 256 bits y 384 bits. Este algoritmo lo proporciona la clase ECDsaCng. Permite firmar con una clave privada y realizar las comprobaciones con una clave pública.
- Diffie-Hellman de curva elíptica (ECDH), que utiliza curvas de módulos primos de 256 bits y 384 bits para el acuerdo confidencial o intercambio de claves. Este algoritmo lo proporciona la clase ECDiffieHellmanCng.

Los contenedores de código administrado para las implementaciones certificadas del Estándar federal de procesamiento de información (FIPS) de AES, SHA-256 y SHA-384 están disponibles en las nuevas clases AesCryptoServiceProvider, SHA256CryptoServiceProvider y SHA384CryptoServiceProvider.

Las clases de Criptografía de próxima compilación (CNG) proporcionan una implementación administrada de la Crypto API (CAPI) nativa. La clase de contenedor de claves CngKey es fundamental en este grupo, pues abstrae el almacenamiento y el uso de claves CNG. Esta clase permite almacenar de forma segura un par de claves o una clave pública y hacer referencia a ella utilizando un nombre de cadena simple. Las clases ECDsaCng y ECDiffieHellmanCng utilizan objetos CngKey.

La clase CngKey se utiliza en otras numerosas operaciones, entre las que se incluyen la apertura, creación, eliminación y exportación de claves. También proporciona acceso al identificador de clave subyacente que se va a utilizar en las llamadas directas a las API nativas.

Existe un buen número de clases CNG compatibles, como CngProvider, que mantiene un proveedor de almacenamiento de claves, CngAlgorithm, que mantiene un algoritmo de CNG, y CngProperty, que mantiene propiedades de clave que se utilizan habitualmente.

⊟ Conexión de red

Conexión de red punto a punto

La conexión de red punto a punto es una tecnología de red sin servidor que permite que varios dispositivos de red compartan recursos y se comuniquen directamente entre sí. El espacio de nombres System.Net.PeerToPeer proporciona un conjunto de clases compatibles con el Protocolo de resolución de nombres de mismo nivel (PNRP) que permite detectar otros nodos del mismo nivel a través de los objetos PeerName registrados en una nube punto a punto. PNRP puede resolver nombres del mismo nivel en direcciones IP de tipo IPv6 o IPv4.

Colaboración a través de la conexión de red punto a punto

El espacio de nombres System.Net.PeerToPeer.Collaboration proporciona un conjunto de clases que admiten la colaboración a través de la infraestructura de conexión de red punto a punto. Estas clases simplifican el proceso por el que las aplicaciones pueden:

- Realizar un seguimiento de la presencia de elementos del mismo nivel sin un servidor.
- Enviar invitaciones a los participantes.
- Detectar elementos del mismo nivel en la misma subred o LAN.
- Administrar contactos.
- Interactuar con elementos del mismo nivel.

La infraestructura de colaboración punto a punto de Microsoft proporciona un marco basado en conexiones de red punto a punto para actividades de colaboración sin servidor. El uso de este marco permite que aplicaciones de red descentralizadas utilicen el potencial colectivo de varios equipos a través de una subred o Internet. Estos tipos de aplicaciones se pueden utilizar en actividades como el planeamiento de la colaboración, la comunicación, la distribución de contenido o incluso servicios de contacto con contrincantes de juego en red.

Características mejoradas de rendimiento del socket

La clase Socket se ha mejorado para su uso en aplicaciones que utilizan la E/S de red asincrónica con el fin de lograr el mayor rendimiento posible. Se ha agregado una serie de nuevas clases que forman parte de un conjunto de mejoras del espacio de nombres Socket. Estas clases proporcionan un modelo asincrónico alternativo que se puede utilizar en aplicaciones de socket de alto rendimiento especializadas. Estas mejoras se han diseñado específicamente para aplicaciones de servidores de red que requieren un alto rendimiento.

⊟ Windows Communication Foundation

Integración con WCF y WF — Servicios de flujo de trabajo

.NET Framework 3.5 unifica los marcos de Windows Workflow Foundation (WF) y Windows Communication Foundation (WCF) para que puedan utilizar WF como un mecanismo de creación de servicios de WCF o para que pueda exponer el flujo de trabajo de WF existente como un servicio. De este modo, es posible crear servicios que se pueden almacenar, pueden transferir datos con facilidad dentro y fuera de un flujo de trabajo y pueden exigir protocolos en el nivel de la aplicación.

Servicios duraderos

.NET Framework 3.5 también incorpora la compatibilidad con los servicios de WCF que utilizan el modelo de persistencia de WF para almacenar la información de estado del servicio. Estos servicios duraderos conservan la información de estado en el nivel de la aplicación, de modo que si una sesión se interrumpe y se reanuda de nuevo más tarde, la información de estado de ese servicio se puede volver a cargar desde el almacén de persistencia.

Modelo de programación web de WCF
El modelo de programación web de WCF permite a los programadores generar servicios de tipo Web con WCF. El modelo de programación web incluye una capacidad de procesamiento de URI enriquecida, admite todos los verbos de HTTP, incluso GET, y un sencillo modelo de programación que permite trabajar con una gran variedad de formatos de mensajes (entre los que se incluye XML, JSON y secuencias binarias opacas).

Distribución de WCF
WCF incluye ahora un modelo de objetos con establecimiento inflexible de tipos para procesar las fuentes de distribución, incluidos los formatos Atom 1.0 y RSS 2.0.

WCF y confianza parcial
En .NET Framework 3.5, las aplicaciones que se ejecutan con permisos reducidos pueden utilizar un subconjunto limitado de las características de WCF. Las aplicaciones de servidor que se ejecutan con permisos de nivel de confianza medios de ASP.NET pueden utilizar el modelo de servicio de WCF para crear servicios HTTP básicos. Las aplicaciones cliente que se ejecutan con permisos de Zona de Internet (como aplicaciones de explorador XAML o aplicaciones sin firmar implementadas con ClickOnce) pueden utilizar los proxys de WCF para utilizar los servicios HTTP. Asimismo, las características del modelo de programación web de WCF (incluido AJAX y Distribución) están disponibles para su uso en aplicaciones con confianza parcial.

Integración de WCF y ASP.NET AJAX
La integración de WCF con las funciones de AJAX (Asynchronous JavaScript and XML) en ASP.NET proporciona un modelo de programación de un extremo a otro para la compilación de aplicaciones web que pueden utilizar servicios de WCF. En las aplicaciones web de tipo AJAX, el cliente (por ejemplo, el explorador de una aplicación web) intercambia cantidades pequeñas de datos con el servidor utilizando solicitudes asincrónicas. La integración con características de AJAX en ASP.NET proporciona un mecanismo sencillo para generar servicios web de WCF a los que se puede obtener acceso mediante el JavaScript de cliente del explorador.

Interoperabilidad de servicios web
En .NET Framework 3.5, Microsoft mantiene su compromiso con la interoperabilidad y los estándares públicos e incorpora la compatibilidad con los nuevos estándares de servicios web de transacciones, que son más seguros y confiables:

- Web Services Reliable Messaging v1.1
- Web Services Reliable Messaging Policy Assertion v1.1
- WS-SecureConversation v1.3
- WS-Trust v1.3
- WS-SecurityPolicy v1.2
- Web Services Atomic Transaction (WS-AtomicTransaction) Version 1.1
- Web Services Coordination (WS-Coordination) Version 1.1

- Web Services Policy 1.5 - Framework
- Web Services Policy 1.5 - Attachment

La implementación de estos protocolos se realiza utilizando los nuevos enlaces estándar, <ws2007HttpBinding> y <ws2007FederationHttpBinding>, que están documentados en Guía de interoperabilidad de los protocolos de servicios web.

⊟ Windows Presentation Foundation

En .NET Framework 3.5, Windows Presentation Foundation contiene modificaciones y mejoras en numerosas áreas, entre las que se incluyen el control de versiones, el modelo de la aplicación, el enlace de datos, los controles, los documentos, las anotaciones y los elementos de la interfaz de usuario 3D.

⊟ Windows Workflow Foundation

Integración con WCF y WF — Servicios de flujo de trabajo

.NET Framework 3.5 unifica los marcos de Windows Workflow Foundation (WF) y Windows Communication Foundation (WCF) para que puedan utilizar WF como un mecanismo de creación de servicios de WCF o para que pueda exponer el flujo de trabajo de WF existente como un servicio. De este modo, es posible crear servicios que se pueden almacenar, pueden transferir datos con facilidad dentro y fuera de un flujo de trabajo y pueden exigir protocolos en el nivel de la aplicación.

Reglas

El motor de reglas admite ahora los métodos de extensión, la sobrecarga de operadores y el uso del nuevo operador en las reglas.

⊟ Formularios Windows Forms

Características mejoradas de Click-Once

Se han realizado varias mejoras en ClickOnce. Entre las características mejoradas se incluye la implementación desde varias ubicaciones y la personalización de marca de terceros. La herramienta Mage.exe, que en ocasiones se utiliza con ClickOnce, se ha actualizado para .NET Framework 3.5.

Servicios de autenticación, funciones y configuración

Los servicios de aplicaciones cliente son nuevos en .NET Framework 3.5 y permiten a las aplicaciones basadas en Windows (incluidos los formularios Windows Forms y las aplicaciones de Windows Presentation Foundation) tener fácil acceso a los servicios de inicio de sesión, funciones y perfiles de ASP.NET. Con estos servicios podrá autenticar a los usuarios y recuperar las funciones de usuario y la configuración de aplicaciones de un servidor compartido.

Puede habilitar los servicios de aplicaciones cliente si especifica y configura los proveedores de servicios del cliente en su archivo de configuración de la aplicación o en el Diseñador de proyectos de Visual Studio. Estos proveedores se acoplan al modelo de extensibilidad web y permiten tener acceso a los servicios web a través de las API de inicio de sesión, funciones y configuración de .NET Framework. Los servicios de la aplicación cliente también admiten una conectividad ocasional mediante el almacenamiento y recuperación de información acerca del usuario en una caché de datos local cuando la aplicación está sin conexión.

Compatibilidad con Windows Vista

Las aplicaciones de formularios Windows Forms funcionan perfectamente en Windows Vista y se han actualizado para que tengan el mismo aspecto que las aplicaciones escritas específicamente para Windows Vista, siempre que sea posible. Los cuadros de diálogo de archivos comunes se actualizan automáticamente a la versión de Windows Vista. .NET Framework 3.5 admite también el icono de escudo del Control de cuentas de usuario (UAC).

Compatibilidad con WPF

Puede utilizar los formularios Windows Forms para hospedar controles y contenido de Windows Presentation Foundation (WPF) junto con los controles de formularios Windows Forms. También puede abrir las ventanas de WPF desde un formulario Windows Forms.

⊟ LINQ

Language-Integrated Query (LINQ) es una nueva característica de Visual Studio 2008 y .NET Framework 3.5. LINQ que incorpora eficaces capacidades de consulta a la sintaxis de los lenguajes C# y Visual Basic en forma de patrones de consulta estándar fáciles de aprender. Esta tecnología se puede extender para que sea compatible con cualquier tipo de almacén de datos. .NET Framework 3.5 incluye ensamblados de proveedor de LINQ que habilitan el uso de LINQ para consultar colecciones de .NET Framework, bases de datos de SQL Server, conjuntos de datos ADO.NET y documentos XML.

Los componentes de LINQ que forman parte de .NET Framework 3.5 son:

- El espacio de nombres System.Linq, que contiene el conjunto de operadores de consulta estándar, además de tipos e interfaces que se utilizan en la infraestructura de una consulta de LINQ. Este espacio de nombres está en el ensamblado System.Core.dll.
- El espacio de nombres System.Data.Linq, que contiene clases que admiten la interacción con bases de datos relacionales en aplicaciones de LINQ to SQL.
- El espacio de nombres System.Data.Linq.Mapping, que contiene clases que se pueden utilizar para generar un modelo de objetos de LINQ to SQL que represente la estructura y el contenido de una base de datos relacional.
- El espacio de nombres System.Xml.Linq, que contiene clases de LINQ to XML. LINQ to XML es una interfaz de programación XML en memoria que permite modificar los documentos XML de forma eficaz y sencilla. Con LINQ to XML, puede cargar XML, serializar XML, crear árboles XML desde el principio, manipular árboles XML en memoria y realizar validaciones con XSD. También puede usar una combinación de estas características para transformar la forma de los árboles XML.
- Nuevos tipos de los espacios de nombres System.Web.UI.WebControls y System.Web.UI.Design.WebControls. Estos nuevos tipos, como LinqDataSource, admiten el uso de LINQ en páginas web ASP.NET a través de un control de origen de datos.
- Las clases DataRowComparer, DataRowExtensions y DataTableExtensions del espacio de nombres System.Data admiten consultas de LINQ en objetos DataSet de ADO.NET.

En la biblioteca de clases, los métodos de extensión de LINQ que se aplican a una clase se muestran en la página de los miembros de la clase, en el panel **Contenido** y en el panel **Índice**.

Árboles de expresión

Los árboles de expresiones son nuevos en .NET Framework 3.5 y proporcionan un mecanismo para representar código de nivel de lenguaje en forma de datos. El espacio de nombres System.Linq.Expressions contiene los tipos que conforman los bloques de creación de los árboles de expresiones. Estos tipos se pueden utilizar para representar diferentes expresiones de código, por ejemplo una llamada al método o una comparación de igualdad.

Los árboles de expresiones se utilizan habitualmente en las consultas de LINQ que tienen como destino un origen de datos remoto, como una base de datos SQL. Estas consultas se representan como árboles de expresiones, y esta representación permite a los proveedores de consultas examinarlos y traducirlos a un lenguaje de consulta específico del dominio.

El espacio de nombres System.Linq.Expressions está en el ensamblado System.Core.dll.

Lenguajes de programación

Son tres los lenguajes de programación de Microsoft que tienen .NET Framework como destino explícito.

- Visual C#
- Visual C++ 2008
- Visual Basic

Lo Nuevo en Visual C#

⊟ Lenguaje C# 3.0

El lenguaje C# 3.0 y su compilador presentan varias características de lenguaje nuevas. Estas nuevas construcciones de lenguaje son útiles por separado en varios contextos y colectivamente para realizar consultas Language-Integrated Query (LINQ).

La tabla siguiente contiene las nuevas características del lenguaje C# 3.0:

Feature	Description
Matrices y variables locales con tipo implícito	Cuando se utiliza con variables locales, la palabra clave var indica al compilador que deduzca el tipo de los elementos de variable o matriz en la expresión situada en la parte derecha de la instrucción de inicialización.
Inicializadores de objeto	Habilita la inicialización de objetos sin llamadas explícitas a un constructor.
Inicializadores de colección	Habilita la inicialización de colecciones con una lista de inicialización en lugar de llamadas concretas a **Add** u otro método.
Métodos de extensión	Extienda las clases existentes con métodos estáticos que puedan invocarse mediante la sintaxis de método de instancia.
Tipos anónimos	Habilita la creación inmediata de tipos estructurados sin nombre que se pueden agregar a colecciones y a los que se puede tener acceso utilizando **var**.
Expresiones lambda	Habilita expresiones insertadas con parámetros de entrada que se pueden enlazar a delegados o árboles de expresión.
Palabras clave de consultas	Palabras clave que especifican cláusulas en una expresión de consulta: • Cláusulas **from** • Cláusula **where** (opcional) • Cláusulas de ordenación (opcional) • Cláusula **join** (opcional) • Cláusula **select** o **group** • Cláusula **into** (opcional)
Propiedades autoimplementadas	Habilita la declaración de propiedades utilizando la sintaxis simplificada.
Definiciones de métodos parciales	Ahora los tipos parciales pueden contener métodos parciales.

⊟ Compilador de C# 3.0

Modificadores del compilador /win32Manifest y /noWin32Manifest.

Estos nuevos modificadores del compilador se utilizan para especificar niveles de ejecución solicitados para programas que se ejecutan en Windows Vista.

⊟ IDE de Visual C#

Feature	Description
Compatibilidad con múltiples versiones	Visual Studio 2008 permite especificar una versión de .NET Framework para el proyecto, .NET Framework 2.0, 3.0, o 3.5. El destino de .NET Framework de una aplicación es la versión de .NET Framework requerida en un equipo para que la aplicación se ejecute en dicho equipo.
Plantillas y tipos de proyecto nuevos	Se proporcionan varias plantillas de proyecto nuevas para Windows Presentation Foundation, Windows Communication Foundation y proyectos web.
Compatibilidad con IntelliSense para C# 3.0	El editor de código de Visual C# proporciona finalización de instrucciones e información rápida para ofrecer compatibilidad con las siguientes construcciones de lenguaje nuevas en C# 3.0: • Variables locales con tipo implícito • Expresiones de consulta • Métodos de extensión • Inicializadores de objeto/colección • Tipos anónimos • Expresiones lambda • Métodos Partial
Compatibilidad de refactorización para C# 3.0	Las características de refactorización, **Cambiar nombre**, **Cambiar firma**, **Extraer método**y **Promocionar variable local**, se han actualizado para ofrecer compatibilidad con las siguientes construcciones de lenguaje nuevas: • Expresiones de consulta • Métodos de extensión • Expresiones lambda Además, la refactorización proporciona nuevas opciones y advertencias para ayudar a evitar las consecuencias imprevistas de una acción de refactorización.
Formato del código	El editor de código admite opciones de formato para varias construcciones de lenguaje C# 3.0 nuevas, incluidas las expresiones de consulta.
Organizar instrucciones Using	El editor de código de Visual C# ofrece una forma sencilla de ordenar y quitar declaraciones **using** y **extern**.

Lo Nuevo en Visual C++ 2008

Este tema presenta las nuevas características y cambios que se pueden encontrar en este lanzamiento de Visual Studio 2008.

⊟ Entorno de desarrollo integrado (IDE) de Visual C++

Los cuadros de diálogo que se crean en aplicaciones ATL, MFC y Win32 ahora cumplen las instrucciones de estilo de Windows Vista.

Al crear un nuevo proyecto mediante Visual Studio 2008, todos los cuadros de diálogo que inserte en la aplicación cumplirán la instrucción de estilo de Windows Vista. Si vuelve a compilar un proyecto que creó con una versión anterior de Visual Studio, cualquier cuadro de diálogo existente mantendrá la misma apariencia que tenía previamente.

El Asistente para proyectos ATL ahora tiene una opción para registrar componentes para todos los usuarios.

Comenzando con Visual Studio 2008, los componentes COM y las bibliotecas de tipos que crea el **Asistente para proyectos ATL** se registran en el nodo **HKEY_CURRENT_USER** del Registro, a menos que seleccione **Registrar componentes para todos los usuarios**.

Se puede redirigir la escritura en el Registro.

Con la introducción de Windows Vista, la escritura en ciertas áreas del Registro requiere un programa que se ejecute en modo elevado. No es conveniente ejecutar siempre Visual Studio en modo elevado. La redirección por usuario redirige automáticamente la escritura en el Registro de HKEY_CLASSES_ROOT a HKEY_CURRENT_USER sin ningún cambio de programación.

El Diseñador de clases ahora tiene compatibilidad limitada para el código de C++ nativo.

En las versiones anteriores de Visual Studio, el Diseñador de clases solamente funcionaba con Visual C# y Visual Basic. Los usuarios de C++ ahora pueden utilizar el Diseñador de clases, pero sólo en modo de sólo lectura.

⊟ Bibliotecas de Visual C++

Biblioteca de STL/CLR

Visual C++ ahora incluye la Biblioteca de STL/CLR.

La Biblioteca de STL/CLR es un paquete de la Biblioteca de plantillas estándar (STL), un subconjunto de la Biblioteca estándar de C++, para su uso con C++ y .NET Framework Common Language Runtime (CLR). Con STL/CLR, ahora puede utilizar todos los contenedores, iteradores y algoritmos de STL en un entorno administrado.

Biblioteca MFC

Windows Vista admite controles comunes.

Se han agregado más de 150 métodos en 18 clases nuevas o existentes para admitir las características de Windows Vista o mejorar la funcionalidad de las clases MFC actuales.

La nueva clase CNetAddressCtrl permite especificar y validar direcciones IPv4 y IPv6 o nombres DNS. La nueva clase CPagerCtrl simplifica uso del control de localizador (pager) de Windows. Y la nueva clase CSplitButton simplifica el uso del control splitbutton de Windows para seleccionar una acción predeterminada u opcional.

Biblioteca de compatibilidad de C++

C++ introduce la biblioteca de cálculos de referencias.

La biblioteca proporciona un método fácil y optimizado para calcular referencias de los datos entre los entornos nativo y administrado. La biblioteca es una alternativa a enfoques más complejos y menos eficaces, como PInvoke.

Servidor ATL

El servidor ATL se lanza como un proyecto de origen compartido.

La mayoría del código del servidor ATL se ha lanzado como un proyecto de origen compartido en CodePlex y no se instala como parte de Visual Studio 2008.

Las clases de codificación y descodificación de datos de atlenc.h y las funciones de utilidad y clases de atlutil.h y atlpath.h ahora forman parte de la biblioteca de ATL.

Microsoft seguirá admitiendo las versiones del servidor ATL incluidas en lanzamientos anteriores de Visual Studio siempre que se admitan dichas versiones de Visual Studio. CodePlex continuará el desarrollo del código del servidor ATL como un proyecto de comunidad. Microsoft no admite una versión de CodePlex del servidor ATL.

⊟ Compilador y vinculador de Visual C++

Cambios del compilador

El compilador admite compilaciones incrementales administradas.

Al especificar esta opción, el compilador no volverá a compilar el código cuando cambie un ensamblado al que se hace referencia. En su lugar, realizará una compilación incremental. Los archivos se vuelven a compilar sólo si los cambios afectan al código dependiente.

El compilador admite la microarquitectura de Intel Core.

El compilador contiene optimización para la microarquitectura de Intel Core durante la generación de código. Esta optimización está activada de forma predeterminada y no se puede deshabilitar, ya que también ayuda a los procesadores Pentium 4 y a otros procesadores.

Las funciones intrínsecas admiten los nuevos procesadores AMD e Intel.

Varias instrucciones intrínsecas nuevas admiten la mayor funcionalidad de los procesadores AMD e Intel más recientes.

La función __cpuid está actualizada.

La función __cpuid admite ahora varias características nuevas de las últimas revisiones de los procesadores AMD e Intel.

La opción del compilador /MP reduce el tiempo de compilación total.

La opción /MP puede reducir significativamente el tiempo total que se tarda en compilar varios archivos de código fuente creando varios procesos que compilan los archivos simultáneamente. Esta opción resulta especialmente útil en los equipos que admiten hyperthreading, varios procesadores o varios núcleos.

La opción del compilador /Wp64 y la palabra clave __w64 están obsoletas.

La opción del compilador /Wp64 y la palabra clave __w64, que detectan problemas de portabilidad de 64 bits, están obsoletas y se quitarán en una versión futura del compilador. En lugar de esta opción del compilador y palabra clave, utilice un compilador de Visual C++ diseñado para una plataforma de 64 bits.

/Qfast_transcendentals

Genera código insertado para las funciones transcendentales.

/Qimprecise_fwaits

Quita los comandos **fwait** internos de los bloques **try** cuando se utiliza la opción del compilador /fp:except.

Cambios del vinculador

La información del Control de cuentas de usuario ahora se incrusta en archivos de manifiesto de aplicaciones ejecutables mediante el vinculador de Visual C++ (link.exe).

Esta característica está habilitada de forma predeterminada.

El vinculador ahora tiene la opción /DYNAMICBASE para habilitar la característica de selección aleatoria del diseño del espacio de direcciones de Windows Vista.

Esta opción modifica el encabezado de un archivo ejecutable para indicar si la aplicación debería reubicarse de forma aleatoria en el momento de la carga.

⊟ Ejemplos de Visual C++

Nuevos ejemplos de Visual Studio 2008

Ejemplo CFileDialog: Registrar el orden de los eventos

Crea un cuadro de diálogo personalizado que muestra eventos que se generan al crear un CFileDialog.

Ejemplo CMNCTRL3: Muestra los nuevos controles MFC disponibles en Visual Studio 2008

Muestra algunos de los nuevos controles disponibles con MFC en Windows Vista, incluidos el botón de vínculo de comando (CButton), el control de paginación (CPagerCtrl), el botón de división (CSplitButton) y el control de dirección de red (CNetAddressCtrl).

Ejemplo NETADDR: Ejemplo Vista Net Address Verifier Control

Muestra el uso del control "Net Address Verifier" de Windows Vista.

Ejemplo StlClrLibrary: Muestra las características de STL/CLR

Muestra algunas de las funciones disponibles al utilizar la Biblioteca de STL/CLR.

⊟ Visual C++ Express

Adiciones a Visual C++ Express

Visual C++ Express ahora incluye el SDK de Windows

Los usuarios de Visual C++ Express ahora pueden utilizar el SDK de Windows sin tener que descargarlo e instalarlo por separado. Visual C++ Express ahora también incluye el Asistente para proyectos Win32.

Lo Nuevo en el Lenguaje Visual Basic

Visual Basic 2008 introduce nuevas características de lenguaje, como la inferencia de tipos locales, inicializadores de objetos, tipos anónimos y métodos de extensión. Estas características sirven de apoyo a una nueva característica importante, Language-Integrated Query (LINQ), pero también son útiles por separado.

Este tema introduce las nuevas características y proporciona vínculos a información adicional sobre ellas.

Característica	Descripción
Consultas (Visual Basic)	En este tema, puede buscar más información sobre las consultas, incluida la información sobre las palabras clave siguientes: • **From** • **Where** • **Select** • **Order By** • **Join** • **Group By** • **Skip** • **Take** • **Distinct**
XML en Visual Basic	Ahora puede incluir XML como tipos de datos de primera clase en el código de Visual Basic, de manera que sea rápido y fácil crear, transformar, modificar y consultar XML.
Inferencia de tipo de variable local	Mediante la inferencia de tipos locales (también denominada escritura implícita), el compilador determina los tipos de datos de las variables locales según los valores que se usan para inicializarlas.
Inicializadores de objeto: Tipos con nombre y anónimos	Con los inicializadores de objetos, puede inicializar un objeto de datos complejo en una expresión, sin una llamada explícita a un constructor.
Tipos anónimos	La característica de tipos anónimos permite crear instancias de un nuevo tipo de datos sin escribir primero la definición de la clase. En su lugar, el compilador crea una definición de tipos de datos basada en las propiedades especificadas al declarar la instancia.
Métodos de extensión (Visual Basic)	Los métodos de extensión le permiten agregar métodos a un tipo de datos desde fuera del tipo. Los métodos se pueden invocar como si fueran métodos de instancia normales del tipo de datos
Expresiones lambda	Una expresión lambda es una función sin nombre que se puede utilizar donde haya un tipo delegado válido.

Lo Nuevo en ASP.NET y Desarrollo Web

.NET Framework versión 3.5 incluye mejoras para ASP.NET en áreas concretas. Visual Studio 2008 y Microsoft Visual Web Developer Express también incluyen mejoras y nuevas características para el desarrollo mejorado de web.

Los adelantos más significativos son una mejor compatibilidad para desarrollar sitios web habilitados para AJAX y compatibilidad con Language-Integrated Query (LINQ). Los adelantos incluyen nuevos tipos y controles de servidor, una nueva biblioteca de tipos de cliente orientada a objetos y total compatibilidad con IntelliSense en Visual Studio 2008 y Microsoft Visual Web Developer Express para trabajar con ECMAScript (JavaScript o JScript).

⊟ Mejoras de ASP.NET

.NET Framework versión 3.5 incluye mejoras para ASP.NET en las áreas siguientes:

- Nuevos controles de servidor, tipos y una biblioteca de scripts de cliente que funcionan juntos para permitir el desarrollo de aplicaciones web con estilo AJAX.
- Extensión de la autenticación de formularios basada en servidor, administración de funciones y servicios de perfil como servicios web que pueden usar las aplicaciones basadas en web.
- Un nuevo control de datos ListView que muestra datos y proporciona una interfaz de usuario con un alto grado de personalización.
- Un nuevo control LinqDataSource que expone Language-Integrated Query (LINQ) a través de la arquitectura de controles de origen de datos ASP.NET.
- Una nueva herramienta de combinación (Aspnet_merge.exe) que combina los ensamblados precompilados para admitir la implementación flexible y la administración de lanzamientos.

.NET Framework versión 3.5 también se integra con IIS 7.0. Ahora puede usar servicios ASP.NET como la autenticación de formularios y el almacenamiento en caché para todos los tipos de contenido, no sólo páginas web ASP.NET (archivos .aspx). Esto se debe a que ASP.NET e IIS 7.0 usan la misma canalización de solicitudes. La canalización de procesamiento de solicitudes unificada implica que puede usar código administrado para desarrollar módulos de canalización HTTP que trabajen con todas las solicitudes en IIS. Además, los módulos y controladores IIS y ASP.NET admiten ahora la configuración unificada.

Desarrollo de AJAX

.NET Framework versión 3.5 permite crear aplicaciones web que representan interfaces de usuario de próxima generación con componentes de cliente reutilizables. Puede desarrollar las páginas web aplicando un enfoque basado en servidor, basado en cliente o una combinación de ambos, según sus requisitos. Los modelos de programación basados en cliente y en servidor AJAX incluyen:

- Controles de servidor compatibles con desarrollo de AJAX basado en servidor. Esto incluye los controles ScriptManager, UpdatePanel, UpdateProgress y Timer. Con estos controles se puede crear un comportamiento de cliente enriquecido con un breve script de cliente o sin script, como la representación parcial de páginas y la presentación del progreso de actualización durante las devoluciones de datos asincrónicas.

- Microsoft AJAX Library, que es compatible con el desarrollo basado en cliente y orientado a objetos que es independiente del explorador. Además de ser compatible con los controles de servidor habilitados para AJAX, con la biblioteca de clientes podrá desarrollar componentes de cliente personalizados que amplían los elementos DOM o que representan un elemento DOM.

- Clases de servidor que le permiten desarrollar controles de servidor que se asignan a los componentes de cliente personalizados cuyos eventos y propiedades se establecen mediante declaración. Los tipos de servidor que son compatibles con esta funcionalidad incluyen los controles que se derivan de las clases base ExtenderControl o ScriptControl, o bien que implementan las interfaces IExtenderControl o IScriptControl.

- Compatibilidad para la globalización y localización del script con script de cliente. Con la globalización es posible mostrar fechas y números basados en un valor de referencia cultural (configuración regional). Con la localización puede especificar el contenido localizado (texto, imágenes, etc.) en los componentes de cliente de los elementos de la interfaz de usuario o de los mensajes de excepción.

- Acceso a los servicios web y a los servicios de autenticación de ASP.NET, de administración de funciones y de aplicación de perfiles.

.NET Framework versión 3.5 permite habilitar en una página, de forma sencilla, las actualizaciones parciales asincrónicas de la misma, lo que evita la sobrecarga de las devoluciones de datos de página completa. Sólo tiene que colocar el marcado y los controles existentes dentro de los controles UpdatePanel. Las devoluciones de datos dentro de un control UpdatePanel se convierten en devoluciones de datos asincrónicas y actualizan sólo la parte de la página incluida dentro del panel, lo cual hace que la utilización por parte del usuario sea más fluida. Puede mostrar el progreso de la actualización parcial de la página mediante la utilización de controles UpdateProgress.

Obtener información sobre el desarrollo de AJAX en ASP.NET
La documentación proporciona abundante información para ayudarle a obtener información sobre cómo desarrollar aplicaciones web con estilo AJAX en ASP.NET. Para comenzar, siga la secuencia de temas descrita en Agregar funcionalidad de cliente y AJAX.

Servicios web y servicios de aplicación
.NET Framework versión 3.5 permite crear servicios web basados en ASP.NET (.asmx) y WCF a los que puede llamar desde las páginas web en script de cliente con Microsoft AJAX Library. También puede llamar a los servicios de aplicación basados en servidor que se exponen como servicios web, lo que incluye la autenticación de formularios, la administración de funciones y los perfiles. Estos servicios de aplicación se pueden usar en aplicaciones compatibles con WCF, lo que incluye páginas web habilitadas para AJAX y clientes de formularios Windows Forms. Como resultado, las aplicaciones que se generan con estas tecnologías ASP.NET o WCF pueden compartir información que facilitan los servicios de aplicación.

Control de datos ListView
El control ListView combina diferentes aspectos de controles de datos existentes. El control ListView resulta útil para mostrar datos de cualquier estructura de repetición, de forma similar a los controles DataList y Repeater. Sin embargo, a diferencia de estos controles, el control ListView admite las

operaciones de edición, inserción y eliminación, así como la ordenación y la paginación. El nuevo control DataPager proporciona la funcionalidad de paginación para ListView.

El control ListView es un control con alto grado de personalización que permite usar plantillas y estilos para definir la interfaz de usuario del control. Al igual que en los controles Repeater, DataList y FormView, las plantillas del control ListView no se predefinen para representar una interfaz de usuario concreta en el explorador.

Control DataPager
El control DataPager se usa para recorrer página a página los datos mostrados por un control que implementa la interfaz IPageableItemContainer, como el control ListView. El control DataPager admite la interfaz de usuario de paginación integrada. Puede especificar la interfaz de usuario de paginación con el objeto NumericPagerField, que permite a los usuarios seleccionar una página por número de página. También puede usar el objeto NextPreviousPagerField, que permite a los usuarios desplazarse por las páginas una página a la vez, o saltar a la primero o última páginas. También puede crear una interfaz de usuario de paginación personalizada con el objeto TemplatePagerField.

Control LinqDataSource
El control LinqDataSource expone Language-Integrated Query (LINQ) a través de la arquitectura de controles de origen de datos ASP.NET. El control LinqDataSource se usa cuando se crea una página web que recupera o modifica datos y se desea usar el modelo de programación que proporciona LINQ. Puede simplificar el código de una página web permitiendo que el control LinqDataSource cree automáticamente los comandos para interactuar con los datos. Si usa el control LinqDataSource, puede reducir la cantidad de código que debe escribir para realizar operaciones de datos en comparación con las mismas operaciones en el control SqlDataSource o el control ObjectDataSource. Asimismo, cuando se utiliza el control LinqDataSource, sólo es necesario conocer un modelo de programación para interactuar con tipos diferentes de orígenes de datos.

Puede usar el marcado declarativo para crear un control LinqDataSource que conecte con los datos de una base de datos o de una recolección de datos como una colección. En el marcado, puede especificar los criterios para mostrar, filtrar, ordenar y agrupar los datos. Cuando el origen de datos es una tabla de base de datos SQL, también puede configurar un control LinqDataSource para actualizar, insertar y eliminar datos. Para realizar estas tareas, no necesita escribir los comandos SQL. La clase LinqDataSource proporciona un modelo de eventos que permite personalizar el comportamiento de visualización y actualización.

Herramienta de combinación de ASP.NET
La herramienta de combinación de ASP.NET (Aspnet_merge.exe) permite combinar y administrar ensamblados creados por la herramienta de precompilación de ASP.NET (Aspnet_compiler.exe). (La herramienta de combinación se lanzó anteriormente como un complemento para Visual Studio 2005.) La herramienta de combinación crea ensamblados únicos para el sitio. Puede crear un ensamblado para el sitio web entero, para cada carpeta del sitio web o sólo para los archivos que constituyen la interfaz de usuario del sitio web (páginas y controles).

⊟ Mejoras de Visual Web Developer
Las secciones siguientes proporcionan información sobre las mejoras y las nuevas características en Visual Studio 2008 y Visual Web Developer Express.

Nuevas herramientas de diseño CSS y nueva vista Diseño
El diseñador de páginas web permite ahora trabajar en la vista **Diseño**, la vista **Código fuente** o la vista **Dividir**, que muestran al mismo tiempo las vistas **Diseño** y **Código fuente**.

Visual Studio proporciona ahora herramientas que facilitan el trabajo con hojas de estilos en cascada (CSS). Puede diseñar el contenido de estilo y presentación en la vista **Diseño** con nuevas herramientas de interfaz de usuario como la ventana **Propiedades de CSS**. También puede cambiar la posición, el relleno y los márgenes directamente en la vista **Diseño** con herramientas de diseño visuales WYSIWYG.

IntelliSense para JScript y ASP.NET AJAX
Visual Studio 2008 y Visual Web Developer Express proporcionan ahora IntelliSense significativamente mejorado para codificar en ECMAScript (JScript o JavaScript) y para escribir script de cliente para aplicaciones web de estilo AJAX que usan Microsoft AJAX Library. IntelliSense está disponible para el script de cliente en elementos **script** y para los archivos de script de .js de referencia.

Además, IntelliSense muestra comentarios de código XML. Los comentarios de código XML se utilizan para describir detalles de resumen, parámetros y devolución del script de cliente. ASP.NET AJAX también usa comentarios de código XML para proporcionar a IntelliSense tipos y miembros de ASP.NET AJAX. IntelliSense también es compatible con referencias de archivos de script externos que utilicen comentarios de código XML.

Proyectos de aplicaciones Web
Los proyectos de aplicaciones web, lanzados anteriormente como un complemento para Visual Studio 2005, se integran ahora en Visual Studio. Si usa el modelo de proyectos de aplicación web, puede compilar un sitio web en un ensamblado único en la carpeta Bin y definir explícitamente los recursos del proyecto.

El modelo de proyectos de aplicación web usa la misma semántica para proyectos, generaciones y compilaciones que los proyectos web de Visual Studio .NET 2003. Esto permite migrar fácilmente los sitios web de Visual Studio .NET 2003 a la versión actual de Visual Studio.

Los proyectos de aplicación web no reemplazan el tipo de proyecto de sitio web introducido en Visual Studio 2005. En su lugar, proporcionan otro modelo de proyectos que dispone de más opciones para implementar y mantener las aplicaciones web.

Nota:
Los proyectos de aplicación web no se admiten en Visual Web Developer Express.

Aplicaciones web con destinos múltiples
Visual Studio permite ahora destinar una aplicación web a una versión concreta de .NET Framework. Puede usar una instancia de Visual Studio para desarrollar aplicaciones web para .NET Framework versiones 2.0, 3.0 (Windows Vista) y 3.5.

Compatibilidad del diseñador e IntelliSense con LINQ
Un nuevo conjunto de características de Visual Studio 2008 es compatible con Language-Integrated Query (LINQ) y amplía las eficaces posibilidades de consulta en la sintaxis del lenguaje de C# y Visual Basic. LINQ introduce modelos estándar aprendidos con facilidad para consultar y transformar los datos,

y se puede extender para admitir cualquier tipo de origen de datos. El diseñador proporciona una representación visual de clases de datos que permite crear y modificar rápidamente clases que se asignan a los objetos de una base de datos. La compatibilidad de IntelliSense proporciona información para la sintaxis de lenguaje de LINQ y para utilizar el control LinqDataSource en la vista **Código fuente**.

Compatibilidad para crear y usar servicios WCF en un proyecto web

En Visual Studio, puede agregar servicios web ASP.NET (archivos .asmx) y servicios web WCF (archivos .svc) a un proyecto. Las aplicaciones cliente que se escriben en código administrado tienen normalmente acceso a estos servicio web a través de una clase de proxy. Por ejemplo, estas aplicaciones usan la clase de proxy que Visual Studio genera al usar el cuadro de diálogo **Agregar referencia web**. Las aplicaciones AJAX pueden tener acceso a los servicios web desde el explorador mediante clases de proxy que se generan automáticamente en script de cliente. Para obtener más información, consulte Web Services Architectural Overview.

Compatibilidad con controles extensores de ASP.NET AJAX

Los controles extensores de AJAX mejoran las funciones de cliente de los controles de servidor web estándar en las aplicaciones web ASP.NET. Puede proporcionar una experiencia del usuario basada en web más enriquecida si enlaza uno o más extensores a los controles de servidor web como los controles TextBox, los controles Button y los controles Panel.

Visual Studio admite todos los controles extensores de ASP.NET AJAX. Esto incluye los controles extensores que crea y aquéllos que agrega a partir de orígenes como ASP.NET AJAX Control Toolkit, que está disponible en el sitio Web de ASP.NET.

Nota:
ASP.NET AJAX Control Toolkit es una biblioteca admitida por la comunidad y no admitida por Microsoft.

Lo Nuevo en .NET Compact Framework 3.5

.NET Compact Framework versión 3.5 amplía .NET Compact Framework con muchas características nuevas. Este tema proporciona información sobre las principales incorporaciones y modificaciones.

Puede instalar .NET Compact Framework 3.5 en RAM utilizando un archivo CAB.

Nota:
La versión de .NET Compact Framework que se instala mediante un archivo CAB siempre debe ser más reciente que cualquier otra versión guardada en ROM.

Para instalar .NET Compact Framework 3.5 en ROM en dispositivos con Windows Embedded CE, debe obtener la actualización mensual correcta de Platform Builder en el sitio web Windows Embedded CE Updates.

Windows Communication Foundation

.NET Compact Framework 3.5 admite Windows Communication Foundation (WCF), que es el modelo de programación unificado de Microsoft para generar las aplicaciones orientadas a servicios. Los clientes que están ejecutando .NET Compact Framework pueden conectarse a los servicios web de WCF que ya existan en el escritorio. Además, se ha agregado compatibilidad para un nuevo transporte de WCF, el transporte de correo Microsoft Exchange Server, tanto para aplicaciones .NET Compact Framework como para aplicaciones de escritorio.

LINQ

Language-Integrated Query (LINQ) agrega funciones de consulta de uso general a .NET Compact Framework que se aplican a diferentes orígenes de información, como bases de datos relacionales, datos XML y objetos en memoria.

Formularios Windows Forms

La tabla siguiente describe las mejoras realizadas a los controles de los formularios Windows Forms en .NET Compact Framework 3.5.

Tipo	Cambios
TabPage Panel Splitter PictureBox	Ahora, los usuarios pueden agregar gráficos a estos controles.
Control	Ahora, se admiten fuentes ClearType y puede modificar la propiedad BackColor de los controles de sólo lectura.
ComboBox	Ya no se admiten las propiedades SelectionStart y SelectionLength.

SoundPlayer

.NET Compact Framework 3.5 admite SoundPlayer, que permite reproducir varios sonidos. Un dispositivo puede mezclar estos sonidos si el hardware admite esta posibilidad.

Compresión

.NET Compact Framework 3.5 incorpora compatibilidad para las siguientes clases del espacio de nombres System.IO.Compression:

- CompressionMode
- DeflateStream
- GZipStream

Además, se admite la propiedad AutomaticDecompression.

Delegados

.NET Compact Framework 3.5 admite el método CreateDelegate.

Generador de perfiles de CLR de .NET Compact Framework

.NET Compact Framework 3.5 admite el generador de perfiles de CLR, que sólo estaba disponible con la versión completa de .NET Framework. El generador de perfiles de CLR permite ver el montón administrado de un proceso e investigar el comportamiento del recolector de elementos no utilizados. El generador de perfiles de CLR y su documentación asociada están incluidos en las herramientas avanzadas de .NET Compact Framework.

Herramienta de configuración

.NET Compact Framework 3.5 admite la herramienta de configuración, que proporciona información sobre la versión del motor en tiempo de ejecución y funciones administrativas que permiten, por ejemplo, especificar en qué versión de .NET Compact Framework se ejecutará una aplicación. La herramienta de configuración y su documentación asociada están incluidas en las herramientas avanzadas de .NET Compact Framework.

Depuración

Las mejoras realizadas en la depuración de .NET Compact Framework 3.5 son las siguientes:

- Ahora se admiten las evaluaciones de funciones anidadas.
- Ahora, las excepciones no controladas realizan la interrupción en el lugar donde ocurrió la excepción, en lugar del lugar donde se llamó al método Run.

Registro

Se han realizado las mejoras siguientes en las características de registro:

- Ahora, los registros de interoperabilidad incluyen información sobre los objetos cuyas referencias se van a calcular y que están contenidos en estructuras o en tipos de referencia.
- El registro de finalizador incluye información sobre el orden y la temporización del finalizador.
- Los archivos de registro ya no se bloquean mientras la aplicación se está ejecutando. Por consiguiente, puede leer los registros en tiempo de ejecución.
- Las trazas de la pila incluyen la firma de método completa para distinguir las sobrecargas de los métodos.

⊟ Id. de plataforma

.NET Compact Framework 3.5 proporciona información nueva sobre el tipo de plataforma, concretamente si una plataforma es Pocket PC o Smartphone. Para obtener más información sobre los id. de plataforma, vea la enumeración WinCEPlatform.

⊟ Herramientas del motor en tiempo de ejecución

Ahora, la biblioteca de herramientas del motor en tiempo de ejecución proporciona compatibilidad para ejecutar con el emulador las herramientas de diagnóstico del SDK de .NET Compact Framework, como Monitor de rendimiento remoto. Supervisión remota del rendimiento y su documentación asociada están incluidos en las herramientas avanzadas de .NET Compact Framework.

⊟ Nombres seguros

Ahora se admiten nombres seguros con un tamaño mayor de 1,024 bytes.

⊟ Caché de ensamblados global

Las modificaciones realizadas en la arquitectura del ensamblado global mejoran la administración de errores y la integración con Windows Embedded CE versión 6.0.

⊟ Documentación

La documentación de la biblioteca de clases de .NET Compact Framework 3.5 incluye información mejorada sobre la compatibilidad de las plataformas para las sobrecargas.

Lo Nuevo en Windows Presentation Foundation Versión 3.5

En este tema se explican brevemente las principales diferencias entre las versiones 3.0 y 3.5 de Windows Presentation Foundation (WPF).

⊟ Compatibilidad con la versión 3.0

Compatibilidad con versiones anteriores y posteriores

Una aplicación generada con WPF 3.0 se ejecutará en el motor de tiempo de ejecución de WPF 3.5.

Una aplicación generada con WPF 3.5 se ejecutará en el motor de tiempo de ejecución de la versión 3.0 si la aplicación utiliza únicamente las características disponibles en WPF 3.0.

WPF 3.5 define un nuevo espacio de nombres XML, http://schemas.microsoft.com/netfx/2007/xaml/presentation. Al generar una aplicación mediante WPF 3.5, puede utilizar este espacio de nombres o el espacio de nombres definido en WPF 3.0.

Usar como destino un motor de tiempo de ejecución concreto

Las aplicaciones generadas con WPF 3.0 pueden destinarse a cualquier versión del marco de trabajo igual o posterior a la versión en la que se generaron originalmente.

⊟ Aplicaciones

Se han realizado las mejoras siguientes en el modelo de aplicación:

- Compatibilidad completa con complementos para admitir los complementos visuales y no visuales de aplicaciones independientes y Aplicaciones del explorador XAML (XBAPs).
- Las XBAPs se pueden ejecutar ahora en Firefox.
- Las cookies se pueden compartir entre las XBAPs y aplicaciones web del mismo sitio de origen.
- Mejora de la experiencia de XAML IntelliSense para una mayor productividad.
- Compatibilidad de localización expandida.

Complementos visuales y no visuales en WPF

Una aplicación extensible expone la funcionalidad de modo que permite a otras aplicaciones integrarse con su funcionalidad y extenderla. Los complementos son una manera común para que las aplicaciones expongan su extensibilidad. En .NET Framework, un complemento suele ser un ensamblado empaquetado como una biblioteca de vínculos dinámicos (.dll). La aplicación host carga dinámicamente el complemento en tiempo de ejecución para usar y extender los servicios que expone el host. El host y el complemento interactúan entre sí mediante un contrato conocido, que normalmente es una interfaz común publicada por la aplicación host.

Cuando una aplicación admite complementos, los desarrolladores propios y los de otros fabricantes pueden crear complementos para ella. Hay muchos ejemplos de estos tipos de aplicación, entre ellos Office, Visual Studio y Microsoft Windows Media Player. Por ejemplo, la compatibilidad con complementos de Microsoft Windows Media Player permite que otros fabricantes creen descodificadores de DVD y codificadores de MP3.

.NET Framework implementa los bloques de construcción que permiten que las aplicaciones admitan complementos. Sin embargo, el tiempo y la complejidad necesarios para generar esa compatibilidad pueden ser costosos, teniendo en cuenta que un diseño de complemento robusto debe ocuparse de lo siguiente:

- Detección: búsqueda de complementos que se adhieran a los contratos admitidos por las aplicaciones host.
- Activación: carga, ejecución y establecimiento de la comunicación con los complementos.
- Aislamiento: uso de dominios de aplicación o procesos para establecer límites de aislamiento que protejan las aplicaciones frente a posibles problemas de seguridad y ejecución con los complementos.
- Comunicación: los complementos y las aplicaciones host deben poder comunicarse entre sí más allá de los límites de aislamiento llamando a métodos y pasando datos.
- Administración de la duración: carga y descarga de los dominios de aplicación y procesos de una manera limpia y predecible.
- Control de versiones: garantía de que las aplicaciones host y los complementos puedan continuar comunicándose cuando se creen nuevas versiones de cualquiera de ellos.

En lugar de exigirle que resuelva estos problemas, .NET Framework incluye ahora un conjunto de tipos, ubicados en el espacio de nombres System.AddIn, que se conocen como el "modelo de complementos". El modelo de complementos de .NET Framework proporciona funcionalidad para cada uno de los comportamientos de complemento comunes que se han mencionado anteriormente.

En algunos escenarios, sin embargo, también puede ser conveniente permitir que los complementos se integren con las UIs de la aplicación host y las extiendan. WPF extiende el modelo de complementos de .NET Framework para permitir esta compatibilidad, que se genera en torno a un objeto FrameworkElement propiedad de un complemento en las UIs de una aplicación host. Esto permite a los desarrolladores de WPF crear aplicaciones compatibles con los siguientes escenarios comunes:

- Aplicaciones de estilo Messenger que proporcionen servicios adicionales con complementos afines de otros fabricantes.
- Aplicaciones de juegos diseñadas para hospedar juegos de otros fabricantes.
- Aplicaciones lectoras de contenido que hospeden anuncios.
- Aplicaciones mashup que hospeden módulos arbitrarios; por ejemplo, Windows Sidebar.

Por último, los complementos de WPF pueden ser hospedados tanto por aplicaciones independientes como por XBAPs.

Compatibilidad con Firefox para aplicaciones XBAP

Un complemento para WPF 3.5 permite ejecutar XBAPs desde Firefox 2.0. Esta característica no está disponible en WPF 3.0. Entre las características principales se incluyen las siguientes:

- Si Firefox 2.0 es su explorador predeterminado, las XBAPsrespetan la configuración. Es decir, no se utiliza Internet Explorer para las XBAPs si Firefox 2.0 es la opción predeterminada.
- Las características de seguridad disponibles para las XBAPs que se ejecutan en Internet Explorer también están disponibles para las XBAPs que se ejecutan en Firefox 2.0, incluido el recinto de seguridad de confianza parcial. Las características de seguridad adicionales proporcionadas por el explorador son específicas del explorador.

Cookies

Las aplicaciones WPF independientes y las XBAPs pueden crear, obtener y eliminar cookies tanto de sesión como persistentes. En WPF 3.5, las cookies persistentes se pueden compartir entre las XBAPs, los servidores web y los archivos HTML que tienen el mismo sitio de origen.

Mejoras de Visual Studio IntelliSense

Ahora es posible agregar un nuevo elemento XAML mediante el editor XAML de Visual Studio, asignarle un nombre (mediante el atributo **Name**), hacer referencia a él desde código subyacente y ver sus miembros desde el explorador de IntelliSense.

Localización

WPF 3.5 agrega compatibilidad para los siguientes sistemas de escritura:

- Bengalí
- Devanagari
- Gujarati
- Gurmukhi
- Kannada
- Malayalam
- Oriya
- Tamil
- Telugu

Compatibilidad con el Editor de métodos de entrada (IME) para el control TextBox

La clase FrameworkTextComposition tiene ahora las siguientes propiedades:

- CompositionOffset
- CompositionLength
- ResultOffset
- ResultLength

Se utiliza un objeto FrameworkTextComposition como propiedad TextCompositionEventArgs.::.TextComposition cuando el usuario escribe texto en un control TextBox mediante un IME y se produce el evento TextInput, TextInputUpdate o TextInputStart.

⊟ Gráficos

Ahora se pueden almacenar en memoria caché las imágenes que se descargan a través de http en la memoria caché local de archivos temporales de Microsoft Internet Explorer, de modo que las subsiguientes solicitudes de la imagen procedan del disco local en lugar de Internet. En función del

tamaño de las imágenes, ésta puede ser una importante mejora de rendimiento de la red. Se han agregado los siguientes miembros para permitir esta característica:

- BitmapImage..::.UriCachePolicy
- BitmapDecoder..::.Create(Uri, BitmapCreateOptions, BitmapCacheOption, RequestCachePolicy)
- BitmapFrame..::.Create(Uri, RequestCachePolicy)
- BitmapFrame..::.Create(Uri, BitmapCreateOptions, BitmapCacheOption, RequestCachePolicy)

Se ha agregado el evento BitmapSource..::.DecodeFailed para notificar al usuario cuando no se carga una imagen debido a un encabezado dañado.

⊟ Gráficos 3D

Se han agregado las siguientes características nuevas al modelo de objetos 3D.

Compatibilidad con la entrada, el foco y los eventos en 3D

El modelo de objetos 3D admite ahora conceptos de UIElement como la entrada, el foco y los eventos. Las nuevas clases que proporcionan estos servicios son UIElement3D y sus clases derivadas ContainerUIElement3D y ModelUIElement3D.

Contenido 2D interactivo en 3D

La nueva clase Viewport2DVisual3D proporciona la compatibilidad necesaria para colocar contenido 2D interactivo en un objeto 3D.

Nuevos servicios de transformación

Las nuevas clases GeneralTransform3D, GeneralTransform2DTo3D y GeneralTransform3DTo2D permiten las transformaciones entre objetos Visual3D, así como de objetos 2D a objetos 3D y viceversa.

⊟ Enlace de datos

Se han realizado las mejoras siguientes en el enlace de datos:

- Un nuevo mecanismo de depuración facilita la depuración de los enlaces de datos.
- El modelo de datos permite la validación en la capa de negocios proporcionando compatibilidad para la interfaz IDataErrorInfo. Además, el modelo de validación admite ahora el uso de la sintaxis de propiedad para establecer las reglas de validación.
- El modelo de enlaces de datos admite ahora LINQ y XLINQ.

Nuevo mecanismo de depuración

Ahora, los enlaces de datos son más fáciles de depurar. Se puede establecer la nueva propiedad adjunta PresentationTraceSources..::.TraceLevel en un objeto relacionado con el enlace para recibir información sobre el estado de un enlace concreto. PresentationTraceSources es una clase estática en el espacio de nombres System.Diagnostics.

Compatibilidad con IDataErrorInfo

El modelo de validación de datos admite ahora la interfaz IDataErrorInfo para que un objeto de negocios pueda determinar la validez de la entrada. La interfaz define un indizador que toma un nombre de propiedad y devuelve una cadena. Se ha agregado la regla de validación DataErrorValidationRule, que comprueba las excepciones devueltas por el indizador. Para obtener un ejemplo, vea Ejemplo Business Layer Validation.

Sintaxis alternativa para la validación de datos

Las clases Binding y MultiBinding tienen dos propiedades nuevas, **ValidatesOnExceptions** y **ValidatesOnDataErrors**. Estas dos propiedades proporcionan una alternativa a establecer ExceptionValidationRule o DataErrorValidationRule en la sintaxis de elementos.

Compatibilidad con LINQ y XLINQ

Se han realizado mejoras en BindingListCollectionView para proporcionar una mejor compatibilidad con el enlace a una colección de tipo BindingList<(Of <(T>)>) y con LINQ. El comportamiento de los enlaces de datos con un objeto CollectionView sobre una interfaz IEnumerable también se ha mejorado para proporcionar un mayor rendimiento y una mejor compatibilidad con el enlace a los resultados generados por LINQ.

Además, el modelo de enlaces de datos también proporciona compatibilidad con XLINQ.

⊟ Controles

RichTextBox

El control RichTextBox conserva ahora los objetos TextElement personalizados cuando guarda los objetos TextElement y cuando los objetos TextElement participan en operaciones del portapapeles. Las siguientes nuevas API permiten este comportamiento:

- La clase TextRange tiene una nueva sobrecarga Save(Stream, String, Boolean) que acepta un valor booleano que especifica si se deben conservar los objetos TextElement personalizados.

- La clase TextElementEditingBehaviorAttribute permite especificar el comportamiento del objeto TextElement personalizado. Cuando se establecen las propiedades TextElementEditingBehaviorAttribute..::.IsMergeable y TextElementEditingBehaviorAttribute..::.IsTypographicOnly en **false**, un control RichTextBox conserva los límites y el contenido del objeto TextElement personalizado cuando el usuario modifica el contenido del control RichTextBox.

El control RichTextBox tiene una nueva propiedad denominada IsDocumentEnabled. Cuando el valor de IsDocumentEnabled es **true**, los elementos de la interfaz de usuario, como botones e hipervínculos, aceptan los datos proporcionados por el usuario.

TextBoxBase

TextBoxBase tiene una nueva propiedad denominada UndoLimit, que especifica el número máximo de acciones a las que el control hace referencia.

SoundPlayerAction

SoundPlayerAction puede ahora cargar archivos de audio que pueden ser identificados mediante pack identificadores de recursos uniformes (URIs) relativos y absolutos:

- Archivos de recursos: archivos de audio con la acción de compilación Resource.
- Archivos de contenido: archivos de audio con la acción de compilación Content.
- Archivos de sitio de origen: archivos de audio con la acción de compilación None.

Descriptores de acceso set protegidos

Los descriptores de acceso **set** de las siguientes propiedades están ahora protegidos, en lugar de ser internos:

- Thumb..::.IsDragging
- ButtonBase..::.IsPressed

- MenuItem..::.IsPressed
- MenuItem..::.IsHighlighted
- ComboBoxItem..::.IsHighlighted

⊟ Documentos

FlowDocumentPageViewer, FlowDocumentScrollViewer y FlowDocumentReader tienen una nueva propiedad pública denominada **Selection**. La propiedad obtiene el objeto TextSelection que representa el contenido seleccionado en el documento.

⊟ Anotaciones

El marco de trabajo de anotaciones expone ahora las capacidades necesarias para hacer coincidir las anotaciones con los correspondientes objetos anotados. Se ha agregado una nueva interfaz denominada IAnchorInfo. Además, se ha agregado un nuevo método denominado GetAnchorInfo, que devuelve un objeto IAnchorInfo, a la clase AnnotationHelper.

Estas nuevas adiciones permiten la existencia de escenarios en los que se necesite obtener acceso al objeto respecto al que está delimitado el objeto de anotación.

Lo Nuevo en Visual Studio 2008

Este tema contiene información sobre algunas de las características nuevas y mejoras de Visual Studio 2008.

⊟ Entorno de desarrollo integrado (IDE)

Migración de la configuración

Si ha instalado Visual Studio 2005 y Visual Studio 2008 en el mismo equipo, cuando inicie por primera vez Visual Studio 2008, podrá migrar la mayor parte de la configuración de Visual Studio 2005. Los fragmentos de código y los complementos de otro fabricante no se pueden migrar de forma automática y se deben volver a instalar manualmente para su uso en Visual Studio 2008. Si no ha instalado Visual Studio 2005 y Visual Studio 2008 en el mismo equipo, aún puede migrar manualmente la configuración de Visual Studio 2005 para su uso en Visual Studio 2008.

Componentes de la comunidad

Al crear componentes de la comunidad con Visual Studio 2008, puede especificar si desea instalar el componente para usarlo con Visual Studio 2005 y Visual Studio 2008 o si simplemente desea usarlo con Visual Studio 2008 utilizando un nuevo valor para el elemento ContentVersion. Si instala un componente de la comunidad diseñado en Visual Studio 2005, el componente se instalará automáticamente para su uso con Visual Studio 2005 y Visual Studio 2008.

Nota:
Es posible que un componente de la comunidad que se haya creado con Visual Studio 2005 no funcione en Visual Studio 2008 y viceversa, en función del diseño.

Menús Comunidad y Ayuda

El menú **Comunidad** se ha quitado en Visual Studio 2008. Los comandos anteriormente denominados **Formular una pregunta** y **Comprobar estado de la pregunta** se han combinado en un comando nuevo que se llama **Foros de MSDN**, el cual se encuentra en el menú **Ayuda**. El comando **Enviar comentarios** es ahora **Informar de un error**, que también se encuentra en el menú **Ayuda**. El resto de los comandos que estaban en el menú **Comunidad** se han quitado en Visual Studio 2008.

Administración de ventanas y elementos generales del usuario

Se han actualizado varios elementos de la interfaz de usuario (UI). Se incluyen los siguientes:

- **Navegador del IDE**: una interfaz mejorada que facilita pasar de un elemento a otro.
- Se han mejorado los destinos de acoplamiento de las ventanas de herramientas con el fin de que resulte más sencillo acoplar las ventanas.
- Cuadros de diálogo comunes: Visual Studio 2008 utiliza cuadros de diálogo estándar de Windows en lugar de cuadros de diálogo personalizados. Esto hace que la navegación sea más coherente con la de Windows.
- Ahora, puede especificar una fuente personalizada para los elementos IDE no identificados individualmente en la lista **Mostrar valores para** en el Fuentes y colores, Entorno, Opciones (Cuadro de diálogo) mediante la utilización de la nueva opción **Fuente del entorno**.

Compatibilidad con el Diseñador de clases para código de Visual C++
En versiones anteriores de Visual Studio, el Diseñador de clases admitía únicamente los lenguajes administrados (Visual C# y Visual Basic). En Visual Studio 2008, el Diseñador de clases agrega compatibilidad limitada para código nativo de C++, que sólo se puede utilizar para visualización y documentación.

⊟ Proyectos y soluciones

Proyectos de aplicaciones Web
El nuevo modelo de proyecto de aplicaciones Web proporciona la misma semántica de proyectos Web que el modelo de proyectos Web de Visual Studio .NET 2003. Se incluye una estructura basada en archivos de proyecto y un modelo de generación basado en la compilación de todo el código de un proyecto en un solo ensamblado. Además, el nuevo tipo de proyecto admite muchas de las características nuevas de Visual Studio 2005 (como diagramas de clase, desarrollo de pruebas y genéricos) y de ASP.NET versión 2.0 (como páginas maestras, controles de datos, pertenencia e inicio de sesión, administración de funciones, elementos web, personalización, navegación por el sitio y temas).

El modelo de proyecto de aplicaciones web de Visual Studio 2005 quita dos elementos que se requieren para los proyectos web en Visual Studio .NET 2003:

- El uso de Extensiones de servidor de FrontPage. Estas extensiones ya no son necesarias, pero siguen siendo compatibles si el sitio ya las utiliza.
- El uso de una copia local de Internet Information Services (IIS). El nuevo modelo de proyecto es compatible con IIS y con el servidor de desarrollo integrado de ASP.NET.

Utilice proyectos de aplicaciones web cuando tenga que realizar una las siguientes operaciones:

- Migrar aplicaciones grandes de Visual Studio .NET 2003 a Visual Studio 2005.
- Controlar los nombres de los ensamblados de salida.
- Utilizar clases independientes para hacer referencia a clases de páginas y clases de control de usuarios.
- Generar una aplicación Web que incluya varios proyectos Web.
- Agregar pasos previos y posteriores a la generación durante la compilación.

Desarrollo de AJAX
Ahora puede crear aplicaciones web con interfaces de usuario de próxima generación y con componentes de cliente reutilizables que utilicen las nuevas características de Visual Studio 2005. Puede desarrollar las páginas web aplicando un enfoque basado en servidor, basado en cliente o una combinación de ambos, según sus requisitos. Los modelos de programación basados en cliente y en servidor AJAX son compatibles con lo siguiente:

- Controles de servidor compatibles con desarrollo de AJAX basado en servidor. Esto incluye los controles ScriptManager, UpdatePanel, UpdateProgress y Timer. Con estos controles se puede crear un comportamiento de cliente enriquecido, como la representación parcial de páginas y la presentación del progreso de actualización durante las devoluciones de datos asincrónicas, con un breve script de cliente o sin script.

- Microsoft AJAX Library, que es compatible con el desarrollo basado en cliente y orientado a objetos que es independiente del explorador. Además de ser compatible con los controles de servidor habilitados para AJAX, con la biblioteca de clientes podrá desarrollar componentes de cliente personalizados que amplían los elementos DOM o que representan un elemento DOM.

- Clases de servidor que le permiten desarrollar controles de servidor que se asignan a los componentes de cliente personalizados cuyos eventos y propiedades se establecen mediante declaración. Los tipos de servidor que son compatibles con esta funcionalidad incluyen los controles que se derivan de las clases base ExtenderControl o ScriptControl, o bien, que implementan las interfaces IExtenderControl o IScriptControl.

- Compatibilidad para la globalización y localización de scripts. Con la globalización es posible mostrar fechas y números basados en un valor de referencia cultural (configuración regional). Con la localización puede especificar el contenido localizado (texto, imágenes, etc.) para los componentes de cliente de los elementos de la interfaz de usuario o de los mensajes de excepción.

- Obtenga acceso a los servicios web y a los servicios de autenticación de ASP.NET, de administración de funciones y de aplicación de perfiles.

Visual Studio 2008 le permite habilitar en una página, de forma sencilla, las actualizaciones parciales asincrónicas de la misma, lo que evita la sobrecarga de las devoluciones de datos de página completa. Sólo tiene que colocar el marcado y los controles existentes dentro de controles UpdatePanel. Las devoluciones de datos dentro de un control UpdatePanel se convierten en devoluciones de datos asincrónicas y actualizan sólo la parte de la página dentro del panel, lo cual hace que la utilización por parte del usuario sea más fluida. Puede mostrar el progreso de la actualización parcial de la página mediante la utilización de los controles UpdateProgress.

Diseñador de proyectos

Compatibilidad del Diseñador de proyectos con las aplicaciones de Windows Presentation Foundation (WPF)

Las aplicaciones de Windows Presentation Foundation (WPF) se han agregado a Visual Studio 2008. Hay cuatro tipos de proyecto de WPF:

- Aplicación de WPF (.xaml, .exe)

- Aplicación de explorador de WPF (.exe, .xbap)

- Biblioteca de controles personalizados de WPF (.dll)

- Biblioteca de controles de usuario de WPF (.dll)

Cuando un proyecto de WPF se carga en el IDE, la interfaz de usuario de las páginas del Diseñador de proyectos le permite especificar las propiedades particulares de las aplicaciones de WPF.

Compatibilidad del Diseñador de proyectos con los proyectos de aplicaciones web

Los proyectos de aplicaciones web se han agregado a Visual Studio en Visual Studio 2005 Service Pack 1 y también se incluyen en Visual Studio 2008. El nuevo modelo de proyecto de aplicaciones web proporciona la misma semántica para proyectos de aplicaciones web que el modelo de proyectos web de

Visual Studio .NET 2003, con la excepción de que ha sido actualizado con características de Visual Studio 2005 y de ASP.NET 2.0. El Diseñador de proyectos de Visual Studio es compatible con los proyectos de aplicaciones web, con las restricciones siguientes:

- En la página **Configuración**, los proyectos de aplicación Web sólo pueden ser de ámbito de aplicación.
- En la página **Firma**, la opción de firma de manifiestos está deshabilitada porque los proyectos de aplicaciones web no usan la implementación ClickOnce.

Compatibilidad del Diseñador de proyectos con las versiones de .NET Framework
La compatibilidad con .NET Framework permite usar el código con una versión específica de .NET Framework:

- .NET Framework 2.0, que se incluía con Visual Studio 2005.
- .NET Framework 3.0, que se incluye con Windows Vista.
- .NET Framework 3.5, que se incluye con Visual Studio 2008.

Debido a esta nueva compatibilidad, los cuadros de diálogo **Configuración de compilador avanzada** (Visual Basic) y **Configuración de generación avanzada** (C#) cuentan con una nueva lista desplegable **Marco de trabajo de destino** que le permite especificar estos sistemas operativos.

Implementación
Implementación ClickOnce
La implementación ClickOnce se ha mejorado con las siguientes características nuevas:

- ClickOnce admite la implementación de aplicaciones de explorador web de WPF. Las aplicaciones de explorador web de WPF están hospedadas en un explorador web y, por tanto, requieren una configuración de implementación y seguridad especial. Cuando se generan e implementan estas aplicaciones, Visual Studio proporcionará la interfaz de usuario adecuada y los valores predeterminados.
- ClickOnce ofrece a los fabricantes independientes de software (ISV) la opción de volver a firmar el manifiesto de la aplicación con el nombre de la compañía, el nombre de la aplicación y la dirección URL de implementación/soporte de su cliente. Cuando los usuarios finales instalan la aplicación, sigue apareciendo la marca comercial original de la compañía del ISV en el cuadro de diálogo donde se pregunta si se desea confiar en la aplicación.
- Puede generar e implementar aplicaciones de Visual Studio Tools para Office utilizando la página Publicar del Diseñador de proyectos o el Asistente para publicación.
- ClickOnce admite la generación de manifiestos bajo Control de cuentas de usuario (UAC) en Windows Vista.
- ClickOnce admite la implementación de los complementos y la documentación de Office cuando se utiliza Visual Studio Tools para Office.

- ClickOnce ofrece compatibilidad mejorada con los exploradores de otros fabricantes. Las versiones anteriores admitían las instalación en exploradores de otros fabricantes mediante el uso de complementos, que a veces producía problemas. En esta versión, un usuario puede instalar un archivo de ClickOnce directamente utilizando el comando Run.

- Puede asociar extensiones de nombre de archivo a una aplicación ClickOnce para que la aplicación se pueda iniciar directamente desde el tipo de archivo asociado.

- ClickOnce ofrece compatibilidad mejorada para cambiar la ubicación de implementación de una aplicación y controlar la expiración de certificados.

- Por razones de seguridad, las aplicaciones ClickOnce siempre se instalan y se ejecutan usuario por usuario. Una aplicación que solicita privilegios de administrador a UAC de Windows Vista produce un error durante la instalación.

Implementación de Windows Installer
La implementación de Windows Installer se ha actualizado para Windows Vista y para las versiones más recientes de .NET Framework:

- Se ha actualizado Windows Installer de forma que la instalación en Windows Vista se efectúa sin problemas, incluso cuando se ejecuta bajo UAC.

- La condición de inicio de .NET Framework admite destinar aplicaciones para las nuevas versiones 3.0 y 3.5 de .NET Framework.

Nota:
Al abrir un proyecto de Visual Studio existente en Visual Studio 2008, la propiedad **Version** de las condiciones de inicio de .NET Framework en el proyecto existente se cambia por la versión actual. Debe volver a cambiar la propiedad **Version** al valor adecuado.

Edición

Nuevas herramientas de Diseño CSS y nueva vista Diseño
Visual Studio 2008 presenta una edición de CSS enriquecida con diversas herramientas nuevas para que resulte más fácil que nunca trabajar con hojas de estilos en cascada (CSS). Gran parte del trabajo de preparación del diseño y de los contenidos se puede realizar en la vista Diseño con la cuadrícula Propiedades de CSS, los paneles Aplicar estilos y Administrar estilos y la herramienta Aplicación de estilo directo. También puede cambiar la posición, relleno y márgenes en la vista Diseño con herramientas de diseño visuales WYSIWYG.

IntelliSense para Jscript y ASP.NET AJAX
IntelliSense se ha mejorado significativamente y ahora admite la creación de JScript y el scripting de AJAX en ASP.NET. El script de cliente que se incluye en una página web mediante la utilización de etiquetas <script> tiene ahora la ventaja de contar con IntelliSense, al igual que los archivos de script .js.

Además, IntelliSense muestra comentarios de código XML. Los comentarios de código XML se utilizan para describir el resumen, el parámetro y los detalles de devolución del script de cliente. ASP.NET AJAX también utiliza comentarios de código XML para proporcionar a IntelliSense tipos y miembros de

ASP.NET AJAX. IntelliSense también es compatible con referencias de archivos de script externos que utilicen comentarios de código XML.

Compatibilidad del Explorador de objetos y de la función Buscar símbolo con todas las versiones de .NET Framework

Ahora, puede especificar que el Examinador de objetos sólo muestre información para una versión única de .NET Framework o de .NET Compact Framework. Además, las búsquedas de Buscar símbolo, Buscar y reemplazar (Ventana) pueden restringirse a una versión única de .NET Framework o de .NET Compact Framework.

WPF Designer

Con Windows Presentation Foundation (WPF) Designer podrá crear aplicaciones de WPF y controles personalizados en el IDE. WPF Designer combina edición en tiempo real de XAML con una experiencia mejorada de diseño gráfico. Las siguientes características son nuevas en WPF Designer:

- Con la Vista dividida puede ajustar los objetos en el diseñador gráfico y ver inmediatamente los cambios en el código XAML subyacente. Igualmente, los cambios en el código XAML se reflejan de inmediato en el diseñador gráfico.
- En la ventana Esquema del documento puede ver y moverse a través del XAML manteniendo la selección de elementos completamente sincronizada con el diseñador, el esquema de documento, el editor XAML y la ventana **Propiedades**.
- IntelliSense en el editor XAML habilita la entrada rápida de código. Ahora, IntelliSense admite los tipos que haya definido.
- Se pueden agregar líneas de cuadrícula a las cuadrículas del diseñador para facilitar la posición de los controles.
- Las líneas de ajuste le permiten alinear fácilmente controles y texto.
- Ahora, el diseñador admite la carga de tipos que haya definido. Esto incluye los controles personalizados y controles de usuario.
- Puede cancelar la carga de archivos XAML grandes.
- La extensibilidad en tiempo de diseño admite el modo de diseño y los editores de propiedades.

⊟ Datos

- El Diseñador relacional de objetos (Diseñador R/O) ayuda a los programadores a crear y editar los objetos de LINQ to SQL que realizan asignaciones entre una aplicación y una base de datos. El Diseñador relacional de objetos crea DataContext, clases de entidad y métodos DataContext que usa LINQ to SQL para comunicarse con la base de datos remota y controlar los datos que se usan en la aplicación.
- La compatibilidad de n niveles para los conjuntos de datos con tipo proporciona mejoras para el Diseñador de DataSet que ayudan a separar código **TableAdapter** y código del conjunto de datos con tipo en proyectos adicionales.

- Las funciones de actualización jerárquica se han incorporado en el **Diseñador de DataSet** para permitir guardar datos en varias tablas relacionadas. Se ha agregado un nuevo objeto **TableAdapterManager** a los conjuntos de datos con tipo que proporciona código generado que incluye la lógica de almacenamiento necesaria para mantener la integridad referencial. En lugar de llamar al método **Update** para cada **TableAdapter**, llame al método TableAdapterManager.UpdateAll.

- El almacenamiento en caché de bases de datos local incorpora una base de datos de SQL Server Compact 3.5 y de Microsoft Synchronization Services para ADO.NET en una aplicación y prepara la aplicación para que sincronice los datos periódicamente con una base de datos remota en un servidor. El almacenamiento en caché de bases de datos permite a las aplicaciones reducir el número de viajes de ida y vuelta entre la aplicación y un servidor de base de datos. Esto puede aumentar el rendimiento cuando se trabaja con datos que no se modifican con frecuencia o cuando las aplicaciones no siempre pueden conectarse a la base de datos remota.

- Microsoft SQL Server Compact 3.5 es una base de datos compacta que se puede implementar en equipos de escritorio, en dispositivos inteligentes y en Tablet PC. SQL Server Compact 3.5 es una base de datos local que se incorpora sin esfuerzo en las aplicaciones y se implementa fácilmente.

⊟ Language-Integrated Query (LINQ)

Language-Integrated Query (LINQ) es un nuevo conjunto de características de Visual Studio 2008 que amplía las eficaces posibilidades de consulta en la sintaxis del lenguaje de C# y Visual Basic. LINQ introduce patrones sencillos y fáciles de aprender para consultar y transformar datos, y se puede ampliar para admitir potencialmente cualquier tipo de origen de datos. Visual Studio 2008 incluye ensamblados de proveedor LINQ que habilitan las operaciones de consulta integradas en el idioma de colecciones de .NET Framework (LINQ a Objects), bases de datos SQL (LINQ a SQL), conjuntos de datos de ADO.NET (LINQ a ADO.NET) y documentos XML (LINQ a XML).

Los operadores de consulta estándares son los métodos que incluyen las funciones de la consulta en el modelo LINQ.

⊟ Servicios de aplicaciones cliente

Los servicios de aplicaciones cliente son nuevos en .NET Framework 3.5 y permiten a las aplicaciones basadas en Windows (incluidos los formularios Windows Forms y las aplicaciones de Windows Presentation Foundation) tener fácilmente acceso a los servicios de inicio de sesión de ASP.NET, las funciones y los perfiles. Con estos servicios podrá autenticar a los usuarios y recuperar las funciones de usuario y la configuración de aplicaciones de un servidor compartido.

Puede habilitar los servicios de la aplicación cliente si especifica y configura los proveedores de servicios del cliente en el Diseñador de proyectos de Visual Studio o en su archivo de configuración de la aplicación. Estos proveedores se acoplan al modelo de extensibilidad web y le permiten tener acceso a los servicios web a través del inicio de sesión de .NET Framework, las funciones y las API de configuración. Los servicios de la aplicación cliente también admiten una conectividad ocasional mediante el almacenamiento y recuperación de información acerca del usuario en una caché de datos local cuando la aplicación está sin conexión.

⊟ Informes

Visual Studio 2008 proporciona nuevas características y mejoras para la elaboración de informes.

Nuevos proyectos de informes
Visual Studio 2008 incluye dos nuevas plantillas de proyecto para crear aplicaciones de informes. La plantilla Aplicación de informes se encuentra disponible en el cuadro de diálogo Nuevo proyecto y la plantilla Sitio web de informes de ASP.NET en el cuadro de diálogo Nuevo sitio Web. Al crear un nuevo proyecto de Aplicación de informes, Visual Studio proporciona un informe (.rdlc) y un formulario (.vb/.cs) con un control ReportViewer enlazado al informe. Para un ASP.NET proyecto Sitio web de informes, Visual Studio crearán un sitio web que contiene un informe (.rdlc), una página predeterminada de ASP.NET (.aspx) con un control ReportViewer enlazado al informe y un archivo de configuración web (.config).

Al crear un proyecto de informe, se iniciará un nuevo Asistente para informes. A continuación, podrá utilizar el asistente para generar el informe, o de forma alternativa, cierre el asistente y genere manualmente el informe.

Asistente para informes
Visual Studio 2008 presenta un Asistente para informes, el cual le guiará a través de los pasos para crear un informe básico. Seleccionará un origen de datos para el informe, definirá un conjunto de datos, seleccionará un tipo de informe (tabular o matriz) y aplicará un estilo al informe. Una vez completado el asistente, podrá mejorar el informe con el Diseñador de informes.

El Asistente para informes se inicia automáticamente cuando crea un proyecto de Aplicación de informes o un Sitio web de proyectos de ASP.NET.

Mejoras en el editor de expresiones
Ahora, el editor de expresiones proporciona expresiones de muestra que puede utilizar en los informes. Puede copiar las expresiones de muestra en su informe y utilizarlas tal cual o modificarlas para que se ajusten a sus necesidades.

Impresión de ReportViewer
El control RSClientPrint está ahora disponible cuando el control ReportViewer de ASP.NET se configura para procesamiento local. De esta forma podrá imprimir informes que ha procesado el control y que son independientes de un servidor de informes.

Compresión para PDF
Los controles ReportViewer ahora comprimirán informes que se representan o se exportan a formato PDF cuando se configuran para procesamiento local.

⊟ MSBuild

Elección de una versión concreta de .NET Framework
MSBuild le permite ahora generar proyectos para versiones específicas de .NET Framework. Esta funcionalidad nueva es compatible con diversas funciones nuevas de API.

Capacidades multiprocesador
Ahora MSBuild reconoce cuando un sistema está utilizando varios procesadores, bien procesadores en núcleo, bien procesadores independientes. MSBuild se sirve de todos los procesadores disponibles para reducir el tiempo total de compilación de los proyectos.

Registro mejorado

El registro de eventos de compilación se ha actualizado para administrar compilaciones con procesadores múltiples. MSBuild admite ahora el modelo de registro distribuido además del modelo de registro central y presenta una tecnología nueva conocida como "registradores de reenvío".

Definiciones de elementos

El nuevo elemento de archivo de proyecto de ItemDefinitionGroup le permite definir un conjunto de definiciones de elementos que constituyen los valores de los metadatos predeterminados globales que se aplican a todos los elementos en el proyecto.

Ubicación de ensamblado y cambios de nombre

Los nombres de archivo y las ubicaciones de los ensamblados de MSBuild se han actualizado en Visual Studio 2008. Se ha anexado "v3.5" a los nombres de archivo de ciertos ensamblados.

Lo Nuevo en Visual Studio Team System (VSTS)

Visual Studio Team System incluye muchas características nuevas y mejoradas, que se resumen en este tema.

⊟ Team Foundation

Varios componentes de Team Foundation tienen nuevas características y mejoras para Visual Studio Team System 2008 Team Foundation Server.

Team Foundation Build

Definiciones de compilación

Las definiciones de compilación reemplazan a los tipos de compilación de Microsoft Visual Studio 2005 Team System. A diferencia de los tipos de compilación, puede utilizar la interfaz de usuario de Team Explorer para modificar las definiciones de compilación. Las definiciones de compilación también son compatibles con el área de trabajo en el control de versiones. Ahora puede especificar rutas de acceso locales y almacenar los archivos de compilación en cualquier ubicación que especifique en el control de versiones.

Integración continua de compilaciones

Puede especificar un desencadenador para una compilación al crear una nueva definición de compilación o modificar una existente. Puede usar compilaciones a petición, compilaciones acumuladas e integración continua, en la que cada protección inicia una compilación. También puede definir cuánto tiempo desea esperar entre las compilaciones al definir compilaciones acumuladas.

Compilaciones programadas

Ahora puede ejecutar compilaciones según una programación, aunque no haya ningún cambio.

Agentes de compilación

Los agentes de compilación se pueden denominar independientemente del nombre de equipo de la compilación. Cada agente de compilación se puede conectar a un equipo de compilación a través de dos puertos: un puerto interactivo y el puerto predeterminado utilizado para ejecutar compilaciones.

HTTPS y Capa de sockets seguros (SSL) para la compilación

Ahora puede configurar Team Foundation Build para que exija el uso de HTTPS y SSL.

Nuevas propiedades para personalizar Team Foundation Build

Team System 2008 Team Foundation Server incluye nuevas propiedades para personalizar compilaciones. Estas propiedades incluyen la personalización del comportamiento de las compilaciones de C++, *SkipInvalidConfigurations*, *CustomizableOutDir* y *CustomizablePublishDir*.

Nuevas tareas y destinos para personalizar Team Foundation Build

Team Foundation Build incluye una serie de nuevos destinos que se pueden invalidar para personalizar el proceso de compilación.

Control de código fuente de Team Foundation

Destruir

Ahora puede destruir o eliminar permanentemente de Control de versiones de Team Foundation archivos que están bajo control de código fuente.

Obtener la última versión al desproteger

Ahora puede habilitar Control de versiones de Team Foundation para que recupere automáticamente la última versión de un archivo al desprotegerlo.

Anotar archivos
Ahora puede anotar los archivos de código fuente. En el código fuente puede ver, línea por línea, la información sobre qué cambios se realizaron, quién los hizo y cuándo se efectuaron.

Comparar carpetas
Ahora puede comparar dos carpetas de servidor, dos carpetas locales o una carpeta de servidor y una carpeta local mediante el control de código fuente. Puede ver las diferencias, por ejemplo elementos que faltan y elementos a los que se agregó o de los que se eliminó algo o con cambios que han entrado en conflicto.

Seguimiento de elementos de trabajo de Team Foundation
El rendimiento de la mayoría de las operaciones de seguimiento de elementos de trabajo sometidas a una carga intensa ha mejorado significativamente. Comparado con Visual Studio 2005 Team Foundation Server, el rendimiento se ha duplicado. Ahora se tarda menos en completar las operaciones individuales. Se ha reducido el uso de la CPU en el servidor de nivel de los datos de Team Foundation. Las grandes organizaciones pueden admitir en sus servidores existentes más usuarios de seguimiento de elementos de trabajo que con Visual Studio 2005 Team Foundation Server.

Visual Studio Team System 2008 Team Foundation Server es más escalable. La escalabilidad ha mejorado significativamente los tiempos de respuesta de la mayoría de las operaciones de seguimiento de elementos de trabajo cuando el servidor está en condiciones de carga. Esto es especialmente cierto para los equipos de más de 500 personas. Las grandes organizaciones deberían ser capaces de admitir en sus servidores existentes más usuarios de seguimiento de elementos de trabajo que con Visual Studio 2005 Team Foundation Server.

Administración de Team Foundation Server
Agregar un gran número de usuarios a Visual Studio Team System 2008 Team Foundation Server es mucho más confiable y es menos probable que se produzcan largos retrasos u otros problemas. Mientras que el número total de usuarios admitidos no ha cambiado, la sincronización de usuarios entre Active Directory y Visual Studio Team System 2008 Team Foundation Server se completa mucho más rápido.

⊟ Edición para arquitectura
Visual Studio Team System Architecture contiene nuevas características y mejoras en las siguientes áreas de Visual Studio Team System 2008:

Diseñar sistemas de aplicaciones con un enfoque descendente
Ahora puede usar un enfoque descendente para diseñar sistemas de aplicaciones comenzando con el Diseñador de sistemas. Puede empezar con una nueva solución de diseño de sistemas o puede continuar con una solución existente. Puede agregar directamente sistemas, aplicaciones y extremos como miembros a la definición del sistema. Puede agregar extremos directamente al límite de la definición del sistema y delegar su comportamiento en los miembros más adelante. Puede cambiar el nombre de los miembros y de sus definiciones subyacentes al mismo tiempo. Puede reparar miembros de sistemas de aplicaciones que pierdan sus definiciones.

Conformar los extremos de servicios web .NET para archivos WSDL

Ahora puede conformar las operaciones de un extremo de proveedor de servicios web .NET existente para un archivo WSDL.

Generar proyectos de aplicación web ASP.NET para aplicaciones ASP.NET
Ahora puede seleccionar la plantilla **Aplicación web ASP.NET** para implementar una aplicación ASP.NET. Esta acción genera el tipo de proyecto correspondiente para la aplicación.

Guardar, importar y exportar prototipos personalizados
Ahora puede guardar o instalar prototipos personalizados sólo para uso personal o para todos los usuarios del equipo. Ahora puede instalar prototipos personalizados importándolos en lugar de editar el Registro.

Nota:
Aún es necesario editar el Registro para instalar archivos .sdmdocument de los prototipos de aplicación creados con el Kit de desarrollo de software (SDK) del modelo de definición del sistema (SDM).

Ahora puede exportar los prototipos personalizados que desea compartir con otros usuarios.

Seleccionar entre varias versiones de .NET Framework
Ahora puede seleccionar .NET Framework 2.0, 3.0 ó 3.5 para aplicaciones de Office, Windows y ASP.NET.

Seleccionar entre varias versiones de Office
Ahora puede seleccionar plantillas de proyecto de Office 2003 u Office 2007 para las aplicaciones de Office.

⊟ Edición de la base de datos

Visual Studio Team System Database ahora se integra en la instalación de Visual Studio Team System. Ya no tiene que instalarlo por separado al instalar el conjunto de aplicaciones completo.

Especificar opciones de tabla y de índice
Ahora puede especificar opciones en las definiciones de tabla y de índice, como el formato de almacenamiento vardecimal que es una novedad de Microsoft SQL Server 2005.

⊟ Edición para programadores

Análisis de código
Las herramientas de análisis del código realizan comprobaciones extensas para buscar defectos de código, que se presentan como advertencias en la ventana de errores.

El análisis de código se ha mejorado con las siguientes características:

Extensión y mejora de las reglas
El análisis de código tiene más de 20 nuevas reglas. Las reglas ofrecen ahora mayor exactitud, en particular en lo referente a su denominación.

Corrector ortográfico compatible con diccionarios personalizados
Puede utilizar el corrector ortográfico para cadenas de recursos, así como para nombres de clases, métodos y propiedades. Puede usar un diccionario personalizado para comprobar palabras no estándar.

Mejor control de la supresión en la lista de errores
Puede suprimir los problemas de análisis de código desde la ventana de errores en el nivel de proyecto o en el código fuente.

Opción de supresión automática del código generado
Puede suprimir automáticamente los mensajes de error del código generado. Esto resulta particularmente útil para el código generado por el diseñador.

Mejoras de la directiva de análisis de código
Al copiar la configuración del servidor en el proyecto, ahora tiene la posibilidad de reemplazar la selección local o combinar las reglas de directiva con las reglas de proyecto locales. Asimismo, ahora tiene información más detallada sobre las infracciones de directivas. Esto permite determinar el origen de la infracción.

Métricas de código
Las métricas de código son un conjunto de medidas de software que proporcionan a los programadores una mejor visión del código que están desarrollando. Gracias a las métricas de código, los programadores pueden entender qué tipos y métodos se deben rehacer o probar más a fondo. Además, los equipos de desarrollo identifican los riesgos potenciales, entienden el estado actual de un proyecto y siguen el progreso durante el desarrollo del software.

Herramientas de generación de perfiles
Las herramientas de generación de perfiles de Visual Studio Developer Edition permiten a los programadores medir, evaluar e identificar problemas relacionados con el rendimiento en el código.

Se han agregado las siguientes características a las herramientas de generación de perfiles:

Compatibilidad con 64 bits
El generador de perfiles ahora incluye compatibilidad con aplicaciones de 64 bits que se ejecutan en un sistema operativo y hardware de 64 bits y aplicaciones de 32 bits que se ejecutan en un sistema operativo y hardware de 64 bits.

Pilas de asignación completas
El generador de perfiles tiene pilas de llamadas completas para la asignación. Esto resulta útil para la asignación que se produce en código que no es de usuario, pero que se origina indirectamente a causa de las acciones del usuario. Mediante la pila completa, puede ver exactamente qué partes del código provocan indirectamente la asignación.

Puede recopilar datos de asignación si configura los valores en la página de propiedades de la sesión de rendimiento. Use la vista de asignación en el informe de rendimiento para ver los resultados.

Datos de muestreo en el nivel de línea
Las herramientas de generación de perfiles ahora incluyen un puntero de instrucciones y vistas de línea en los informes de rendimiento. Asimismo, la vista de módulos ahora incluye información de línea.

Reducción de ruido de informes
Puede configurar informes de rendimiento para la reducción de ruido. Esto limita la cantidad de datos en la vista Árbol de llamadas y en la vista Asignación. Al utilizar la reducción de ruido, los problemas de rendimiento destacan más. Esto resulta útil al analizar informes de rendimiento.

Control en tiempo de ejecución

Las herramientas de generación de perfiles incluyen un control en tiempo de ejecución. El control en tiempo de ejecución se inicia automáticamente con el generador de perfiles. Se puede detener y reanudar para el registro de datos de rendimiento. Además, puede usar el control en tiempo de ejecución para iniciar la aplicación con el registro en pausa. Esto permite omitir la recolección de datos al inicio de la aplicación. Al usar el control en tiempo de ejecución, puede insertar manualmente anotaciones en los datos de rendimiento cuando ocurran eventos de interés en la duración de la aplicación. Puede filtrar los datos en sus anotaciones más adelante.

Análisis filtrado

Ahora puede filtrar los informes de rendimiento por marca de tiempo, proceso, subproceso y marcas. Puede usar el botón Mostrar consulta para obtener el análisis filtrado. Asimismo, puede usar la opción /summaryfile del comando VSPerfReport.

Comparar informes

Ahora el generador de perfiles admite la comparación de informes. Puede comparar un informe si usa el Explorador de rendimiento o /diff en las opciones del comando VSPerfReport.

Mejora de la compatibilidad con el contador de chips

Las herramientas de generación de perfiles proporcionan nuevos nombres de contadores de chips más descriptivos (por ejemplo: "Líneas no ejecutadas en L2", "Líneas no ejecutadas en ITLB", "Bifurcaciones mal previstas"). Puede modificar los archivos xml para seguir configurando los contadores destinados a una arquitectura concreta.

Compatibilidad con el contador de Windows

Ahora el generador de perfiles obtiene contadores de Windows (por ejemplo, "% tiempo de procesador", "% tiempo de disco", "Bytes/sec del disco", "Errores de página/seg"). Puede usar el nodo de contadores de Windows en la página de propiedades de la sesión de rendimiento o la opción /wincounter del comando VSPerfCmd. En la vista Marcas se muestran los contadores. Puede usar los contadores como extremos de filtrado.

Archivos de informe comprimidos

Con las herramientas de generación de perfiles podrá generar pequeños archivos de informe comprimidos que se abren con rapidez. Esto se debe a que estos archivos, que se crean a partir de los informes completos, ya se han analizado. Puede hacer clic con el botón secundario en el informe en el Explorador de rendimiento y elegir **Guardar analizados** o usar la opción /summaryfile del comando VSPerfReport.

Ruta de acceso activa

Ahora el generador de perfiles tiene la capacidad de expandir automáticamente la ruta de acceso de código más costosa en el árbol de llamadas y en la vista de asignación del informe de rendimiento.

Copiar datos de vista de informe a HTML

El generador de perfiles incluye compatibilidad con informes enriquecidos en el Portapapeles. Puede copiar y pegar datos enriquecidos (tablas con encabezados y valores) desde los informes de rendimiento.

Compatibilidad con Windows Communications Foundation

Las herramientas de generación de perfiles ahora son compatibles con Windows Communications Foundation (WCF).

Integración de pruebas web y pruebas de carga en Visual Studio Team Suite
Puede crear sesiones de rendimiento para pruebas web y pruebas de carga desde Vista de pruebas y Resultados de pruebas.

Nota:
Esta característica sólo se aplica a Visual Studio Team System.

Edición de prueba

Visual Studio Team System Test contiene nuevas características y mejoras en las siguientes áreas de Visual Studio Team System 2008 Test:

Probar los métodos del código
Ahora es más fácil y rápido crear y ejecutar pruebas unitarias para más tipos de código de producción.

Usar pruebas unitarias en Visual Studio Professional Edition
Los programadores que usan Visual Studio Professional Edition ahora pueden crear y ejecutar dos tipos de pruebas: unitarias y por orden. Una prueba unitaria se usa para validar que un método concreto del código de producción funciona correctamente, probar las regresiones o realizar pruebas relacionadas (buddy) o pruebas de humo. Las pruebas por orden ejecutan otras pruebas en un orden especificado.

Ejecutar pruebas unitarias con mayor facilidad
Los nuevos menús y combinaciones de teclas permiten a los programadores de las pruebas unitarias iniciar ejecuciones de prueba y seleccionar las pruebas para ejecutarlas más rápidamente. Asimismo, ahora puede generar pruebas desde un archivo binario, sin necesidad de tener acceso al código fuente del producto. Puede generar pruebas para tipos de datos genéricos como valores devueltos y parámetros de método.

Usar la herencia entre las clases de prueba
Ahora, las clases de prueba pueden heredar miembros de otras clases de prueba. Esto permite a los programadores crear inicializaciones o pruebas en una clase de prueba base, que heredarán todas las demás clases de prueba derivadas. Esta característica elimina el código de pruebas duplicado. Esto ofrece a los programadores más opciones para personalizar correctamente sus pruebas unitarias.

Ejecutar pruebas unitarias en dispositivos
Visual Studio dispone de un conjunto de herramientas para probar aplicaciones Smart Device de C# y Visual Basic. Estas herramientas proporcionan un subconjunto de la funcionalidad de Test Edition.

Crear adaptadores host
Normalmente, las pruebas se ejecutan en el entorno predeterminado que proporciona Herramientas para pruebas Team System. Para ejecutar pruebas en un entorno diferente, use un adaptador host. Puede usar el SDK de Visual Studio para crear nuevos adaptadores host. Descargue el SDK de Visual Studio en el sitio relacionado .

Mejora del enlace de datos de pruebas unitarias
Ahora puede usar un asistente para enlazar con facilidad una prueba unitaria a un origen de datos, por ejemplo a archivos CSV y XML.

Sitios web de pruebas web
Visual Studio Team System 2008 Test proporciona más control para crear pruebas web.

Llamar a una prueba web desde otra

- Puede insertar una llamada a una prueba web desde una segunda prueba web.

Mejora del enlace de datos de pruebas web
Test Edition ahora incluye compatibilidad integrada con archivos csv y xml. Un nuevo asistente facilita el proceso de enlace de datos. Asimismo, puede ver una vista previa de los datos antes de completar el proceso.

Mejora de las características de las pruebas web
Test Edition ahora incluye compatibilidad con reglas de validación de nivel de prueba. Puede crear reglas de validación en el nivel de prueba. Estas nuevas reglas se pueden aplicar a todas las solicitudes individuales de la prueba. Puede detener una prueba web si se produce un error en ella. Asimismo, puede validar el valor devuelto por un código de estado HTTP previsto.

En Test Edition ahora puede extraer solicitudes de las pruebas web para crear pruebas web nuevas. También puede insertar llamadas a otras pruebas web. Esto significa que puede crear componentes de pruebas web y reutilizar las pruebas y solicitudes web.

Pruebas de carga
Ahora puede usar opciones de modelos de carga más realistas para ejecutar pruebas de carga. Asimismo, puede organizar los datos devueltos de un modo más eficaz y flexible.

Controlar los modelos de carga
Las pruebas de carga ahora ofrecen opciones de modelos de carga adicionales. Estas opciones permiten crear pruebas de carga que reproducen con mayor precisión el uso real esperado de una aplicación o sitio web. Ahora, puede crear modelos de uso basados en el número de pruebas ejecutadas, la cantidad de tiempo dedicada a cada prueba o el ritmo al que los usuarios las ejecutan.

Mejora de las vistas del analizador de prueba de carga

- El analizador de prueba de carga de Test Edition incluye una nueva vista de resumen que muestra los indicadores y resultados clave en una única página que se puede imprimir y exportar. A su vez, cuatro nuevos gráficos integrados muestran la información clave. Se pueden mostrar hasta cuatro gráficos al mismo tiempo. Estas mejoras permiten ver hasta cuatro tablas simultáneamente.

Mejora de la administración del repositorio de resultados de pruebas de carga
Test Edition incluye un nuevo cuadro de diálogo de administración de repositorio que permite tener acceso directamente al repositorio de resultados de pruebas de carga. Ahora puede abrir, importar, exportar y eliminar fácilmente los resultados de las pruebas de carga.

Esquema publicado para los archivos XML
Cuando trabaja con Test Edition, los datos se crean y almacenan en archivos XML. Estos archivos incluyen lo siguiente:

- Archivo de metadatos de prueba. Este tipo de archivo tiene la extensión .vsmdi. Los archivos de metadatos de prueba almacenan información sobre las pruebas de la solución.

- Archivo de resultados de pruebas. Este tipo de archivo tiene la extensión .trx. Cuando se ejecutan pruebas, Visual Studio guarda automáticamente los resultados de las pruebas en un archivo trx.

- Pruebas manuales en formato de texto. Este tipo de archivo tiene la extensión .mtx. Cuando se crea una prueba manual en formato de texto, se guarda en el proyecto de prueba como un archivo de este tipo.

En Team System 2008 Test, todos los archivos XML que usa Test Edition se definen mediante un nuevo XSD denominado TestTypes.xsd. Cualquier modificación que realice en cualquiera de estos archivos, ya sea manualmente o mediante programación, debe dar lugar a un código XML que cumpla el esquema definido en este XSD. De igual forma, todos los archivos que cree con estas extensiones deberán cumplir el esquema definido en el XSD. De lo contrario, Test Edition no podrá utilizarlos.

Los proyectos de prueba creados en Visual Studio 2005 contienen archivos XML. Al abrir un proyecto de prueba de Visual Studio 2005, el asistente para la actualización de proyectos de Visual Studio 2008 le pide permiso para convertir los archivos al nuevo formato. Para usar los archivos en Team System 2008 Test, debe permitir que Visual Studio los convierta. Si decide no convertir o actualizar uno o más archivos, Visual Studio no podrá abrir el proyecto de prueba. De igual forma, si agrega un archivo existente con el formato anterior a un proyecto de prueba, se le pedirá que permita a Visual Studio actualizar el formato de archivo. Si responde afirmativamente, Visual Studio convierte el archivo y éste queda disponible como parte del proyecto de prueba. Si responde negativamente, se cancela la solicitud de agregar los archivos.

Este lanzamiento proporciona las siguientes ventajas:

- Reglas de validación de pruebas web mejoradas.

  Ahora tiene más flexibilidad para aplicar reglas de validación y usar sus resultados para controlar el flujo de programa de la prueba web.

- Mejor control de los modelos de carga.

  Ahora dispone de métodos más flexibles para controlar los modelos de carga en las pruebas de carga que ejecute.

- Mejora de las vistas del analizador de prueba de carga.

  Los nuevos gráficos integrados y funciones de presentación facilitan y agilizan la comprensión de los resultados de las pruebas de carga.

- Mejora de la administración del repositorio de resultados de pruebas de carga.

  Ahora resulta más fácil tener acceso al repositorio de resultados de pruebas de carga.

- Archivo XML esquematizado para los resultados de pruebas.

  Ahora puede trabajar mediante programación con los resultados de pruebas que se almacenan automáticamente en formato XML en un archivo .trx (resultados de pruebas).

Lo Nuevo en Visual Studio Tools para Office (VSTO)

Microsoft Visual Studio Tools para Microsoft Office System (versión 3.0) incluye nuevas características destinadas a ayudarle a realizar las tareas siguientes:

- Personalizar las aplicaciones de Microsoft Office mediante la creación de complementos
- Personalizar documentos de Word y Excel
- Crear flujos de trabajo de SharePoint
- Crear paneles de tareas personalizados
- Personalizar la cinta de opciones
- Estructurar documentos mediante los controles de contenido de Word
- Extender los formularios de Outlook con áreas de formulario personalizadas
- Importar áreas de formulario diseñadas en Outlook
- Proteger e implementar las soluciones de Microsoft Office

⊟ Personalizar las aplicaciones de Microsoft Office mediante la creación de complementos

Los complementos de nivel de aplicación constituyen un mecanismo que permiten incorporar características propias a las aplicaciones de Microsoft Office. El código que escriba estará disponible en la propia aplicación, con independencia de los documentos que estén abiertos.

Visual Studio Tools para Office incluye las características siguientes que simplifican el desarrollo de los complementos:

- Crear complementos utilizando plantillas de proyecto de numerosas aplicaciones de Microsoft Office 2003 y Microsoft Office System 2007.

- Programar complementos utilizando un nuevo modelo de programación menos complejo que el que se utiliza en los complementos COM.

Llamar a código de un complemento desde otras soluciones de Office

Puede exponer un objeto del complemento a otras soluciones de Microsoft Office, como por ejemplo, otros complementos o el código de VBA de los documentos. Esto resulta útil si el complemento proporciona un servicio que desea habilitar para que lo usen otras soluciones de Office.

Utilizar ClickOnce para implementar complementos de nivel de aplicación

Puede utilizar las tecnologías ClickOnce para implementar complementos en aplicaciones de Microsoft Office System 2007.

Visual Studio Tools para Office incluye las características de implementación siguientes:

- Utilizar el **Asistente para publicación** para implementar los complementos.

- Permitir que las soluciones descarguen e instalen automáticamente las actualizaciones cuando se cargue el complemento.

- Cargar y ejecutar los complementos instalados cuando el usuario no esté conectado a una red.

Cargar de forma segura los complementos de nivel de aplicación

Los complementos de las aplicaciones de Microsoft Office System 2007 deben pasar un conjunto de comprobaciones de seguridad antes de poder cargarse en los equipos cliente.

⊟ Personalizar documentos de Word y Excel

Las personalizaciones de nivel de documento constituyen un mecanismo que permite incorporar características propias a un documento o libro concreto. Las características de las personalizaciones que cree sólo estarán disponibles en el documento o libro asociado.

Visual Studio Tools para Office incluye las características siguientes que simplifican el desarrollo de las personalizaciones de nivel de documento en Word 2007 y Excel 2007:

- Crear personalizaciones de documentos y libros en los formatos XML abiertos de Office admitidos en Word 2007 y Excel 2007 o en los formatos de archivo binario admitidos en Microsoft Office 2003 y versiones anteriores
- Diseñar los documentos y plantillas en Visual Studio y escribir el código en el mismo entorno.
- Agregar controles de formularios Windows Forms al documento o plantilla.
- Agregar controles host al documento o plantilla. Los controles host extienden algunos de los objetos integrados en Word y Excel. Estos objetos exponen eventos y tienen capacidad para enlazar datos.

Llamar a miembros en una personalización de nivel de documento desde código de VBA

Puede configurar un proyecto de nivel de documento para que el código de VBA incluido en el documento pueda llamar a miembros públicos del ensamblado del proyecto.

⊟ Crear flujos de trabajo de SharePoint

Diseñe flujos de trabajo de SharePoint que ayuden a mover los documentos que están almacenados en Microsoft Office SharePoint Server 2007 a través de un proceso eficaz.

Visual Studio Tools para Office introduce las características siguientes que simplifican el desarrollo de flujos de trabajo de SharePoint:

- Crear proyectos de flujo de trabajo de SharePoint mediante las plantillas **Flujo de trabajo secuencial de SharePoint 2007** y **Flujo de trabajo de equipo de estado de SharePoint 2007**.
- Presionar F5 para depurar la lógica del flujo de trabajo. Visual Studio Tools para Office asocia automáticamente el flujo de trabajo a una biblioteca de documentos predeterminada de un sitio web local de SharePoint e inicia una instancia del flujo de trabajo.

Configurar los valores de depuración de proyecto de SharePoint a través de un asistente

Cuando cree un proyecto, utilice el asistente **Nuevo flujo de trabajo de Office SharePoint** para establecer los valores de depuración. El asistente incluye las opciones siguientes:

- Especificar el sitio, la biblioteca y las listas de SharePoint que pretenden utilizarse al depurar el flujo de trabajo de SharePoint.
- Especificar las acciones que inician el flujo de trabajo.

⊟ Crear paneles de tareas personalizados

Utilice Visual Studio Tools para Office para crear paneles de tareas personalizados. Los paneles de tareas son paneles de interfaz que normalmente están anclados acoplados en un lado de una ventana de una aplicación de Microsoft Office. Los paneles de tareas personalizados le ofrecen una manera de crear su propio panel de tareas y proporcionar a los usuarios una interfaz conocida para tener acceso a las características de su solución.

Puede crear paneles de tareas personalizados para los complementos de nivel de aplicación de algunas aplicaciones de Microsoft Office System 2007.

⊟ Personalizar la cinta de opciones

Puede personalizar la cinta de opciones de las aplicaciones siguientes:

- Microsoft Office Excel 2007
- Microsoft Office Outlook 2007
- Microsoft Office PowerPoint 2007
- Microsoft Office Word 2007

Personalizar la cinta de opciones mediante su diseñador

Visual Studio Tools para Office incluye las características siguientes para simplificar el proceso de personalización de la cinta de opciones:

- Agregar rápidamente una cinta personalizable a un proyecto de Office utilizando la plantilla de elementos **Cinta (diseñador visual)**.
- Crear fichas personalizadas visualmente utilizando el diseñador de la cinta de opciones:
    - Arrastrar controles hasta la superficie del diseñador de la cinta de opciones.
    - Ajustar el diseño y el aspecto del control.
    - Hacer doble clic en controles para abrir los controladores de eventos.
- Establecer las propiedades de los controles utilizando la ventana **Propiedades**.
- Agregar código personalizado a los controladores de eventos utilizando Visual C# o Visual Basic y aprovechar la comprobación de tipos y la tecnología IntelliSense.

También puede utilizar el diseñador de la cinta de opciones para agregar los controles al menú que se abre al hacer clic en el botón **Microsoft Office**.

Personalizar la cinta de opciones mediante su código XML

Puede utilizar el código XML de la cinta de opciones para realizar personalizaciones avanzadas de la cinta de opciones que no pueden realizarse a través de su diseñador. Las características siguientes también están habilitadas:

- Agregar rápidamente una cinta de opciones personalizable a cualquier proyecto de Visual Studio Tools para Office mediante la plantilla de elementos **Cinta (XML)**.
- Exportar las cintas de opciones que creó utilizando el elemento **Cinta (diseñador visual)** a un elemento de **Cinta (XML)**.

⊟ Estructurar documentos mediante los controles de contenido de Word

Utilice los controles de contenido para crear documentos de Word 2007 estructurados. Un control de contenido define un área que sólo puede contener un tipo específico de contenido, como texto, fechas o imágenes. Puede utilizar controles de contenido para limitar los mecanismos a través de los cuales los usuarios pueden interactuar con las áreas de un documento.

Visual Studio Tools para Office incluye las características siguientes que simplifican el desarrollo de controles de contenido:

- Programar con nuevas clases administradas para cada control de contenido proporcionado por Word 2007.
- Administrar eventos de usuario para cada control de contenido.
- Enlazar controles de contenido a objetos de elementos XML personalizados del documento, a campos de base de datos o a objetos administrados.
- Agregar controles de contenido a los documentos utilizando el diseñador:
    - Arrastrar los controles de contenido hasta la superficie del documento.
    - Establecer las propiedades de los controles utilizando la ventana **Propiedades**.
    - Hacer doble clic en los controles para crear controladores de eventos predeterminados.
- Agregar mediante programación controles de contenido a documentos en tiempo de ejecución.

⊟ Extender los formularios de Outlook con áreas de formulario personalizadas

Utilizar Visual Studio Tools para Office para diseñar áreas de formulario que extienden un formulario de Microsoft Office Outlook estándar o personalizado.

Visual Studio Tools para Office incluye las características siguientes que simplifican el desarrollo de áreas de formulario:

- Agregar rápidamente áreas de formulario al proyecto utilizando la plantilla de elementos **Área de formulario**.
- Definir el diseño del área de formulario y elegir el tipo de formulario que se va a extender mediante el Asistente para áreas de formulario.
- Desarrollar visualmente áreas de un formulario utilizando el Diseñador de áreas de formulario:
    - Arrastrar y colocar controles administrados en la superficie del Diseñador de áreas de formulario.
    - Ajustar el diseño y el aspecto del control.
    - Hacer doble clic en controles para abrir los controladores de eventos.
    - Agregar código personalizado utilizando Visual C# o Visual Basic y aprovechar la comprobación de tipos y la tecnología IntelliSense.

- Depurar el proyecto utilizando herramientas que inicien Outlook automáticamente y generen los archivos y las opciones de configuración que Outlook debe tener para buscar y ejecutar un área de formulario.

⊟ Importar áreas de formulario diseñadas en Outlook

Utilice el asistente **Nueva área de formulario de Outlook** para importar áreas de formulario diseñadas en Microsoft Office Outlook. Cuando realice sus diseños en Outlook, puede utilizar campos y controles nativos de Outlook que no están disponibles en el **Cuadro de herramientas** de Visual Studio.

También puede volver utilizar las áreas de formularios que ya están desarrolladas en otros proyectos de Outlook.

- Después de importar el área del formulario, puede agregar código para administrar los eventos de control.

⊟ Proteger e implementar las soluciones de Microsoft Office

Puede utilizar las tecnologías ClickOnce para proteger e implementar todas las soluciones de Visual Studio Tools para Office para Microsoft Office System 2007, incluidas las personalizaciones de nivel de documento y los complementos de nivel de aplicación.

Visual Studio Tools para Office incluye las características de implementación siguientes:

- Utilizar el **Asistente para publicación** para publicar e implementar personalizaciones y complementos.
- Permitir que las soluciones busquen actualizaciones automáticamente a intervalos regulares, descarguen e instalen las actualizaciones o reviertan a versiones anteriores.
- Cargar y ejecutar las soluciones de Office instaladas cuando el usuario no está conectado a una red.

Cargar con seguridad las soluciones de Microsoft Office

El modelo de seguridad de ClickOnce se ha diseñado para que sea compatible con versiones futuras del motor en tiempo de ejecución de Visual Studio Tools para Office, Microsoft .NET Framework y Microsoft Office.

Visual Studio Tools para Office incluye las características de seguridad siguientes:

- Para Microsoft Office System 2007, protección de las soluciones de Office con un modelo de seguridad independiente de la versión basado en ClickOnce.
- Tomar decisiones de seguridad utilizando el Centro de confianza de Microsoft Office, firmando el manifiesto de implementación con certificados, mostrando el mensaje de ClickOnce relativo a la confianza o agregando entradas mediante programación a la lista de inclusión de usuarios.
- Establecer la directiva de seguridad individualmente para cada usuario de un equipo.

Novedades en ADO.NET

Las siguientes características son nuevas en ADO.NET.

⊟ Language-Integrated Query (LINQ)

Language-Integrated Query (LINQ) es una innovación que aporta capacidad de consulta directamente en los lenguajes de programación de .NET Framework 3.0. Las operaciones de consulta se expresan en el propio lenguaje y no como literales de cadena incrustados en el código de aplicación. LINQ se integra en varios aspectos del acceso a datos de .NET Framework.

⊟ LINQ to DataSet

LINQ to DataSet proporciona capacidad LINQ en para datos desconectados almacenados en un objeto DataSet.

⊟ LINQ to SQL

LINQ to SQL admite consultas en un modelo de objetos asignado a las estructuras de datos de una base de datos de Microsoft SQL Server sin utilizar un modelo conceptual intermedio. Cada tabla se representa mediante una clase distinta, acoplando de manera precisa el modelo de objetos al esquema de la base de datos. LINQ to SQL convierte las consultas de Language-Integrated Query del modelo de objetos a Transact-SQL y las envía a la base de datos para su ejecución. Cuando la base de datos devuelve los resultados, LINQ to SQL los vuelve a traducir a objetos.

⊟ Nuevas características en SqlClient para SQL Server 2008

La futura versión de SQL Server 2008 contiene características compatibles con el proveedor de datos de .NET Framework para SQL Server (System.Data.SqlClient).

Lo Nuevo en Visual Database Tools

Puede utilizar las herramientas Visual Database Tools para crear y mantener bases de datos, así como para diseñar las partes de manipulación de datos de las aplicaciones de base de datos de Visual Studio.

⊟ SQL Server Compact Edition

Ya puede conectarse a una base de datos de SQL Server Compact Edition (SQL Server CE). Mediante este enfoque, puede crear y mantener bases de datos de SQL Server CE del mismo modo que crea y mantiene otras bases de datos de SQL Server.

Lo Nuevo en Datos en Visual Studio 2008

Esta versión de Visual Studio incluye las nuevas características siguientes para desarrollar aplicaciones que tienen acceso a datos:

- La tecnología Language-Integrated Query (LINQ) presenta importantes avances en lenguaje de programación sobre Visual Studio 2005. LINQ to SQL aplica la tecnología LINQ a las bases de datos relacionales.

- El Diseñador relacional de objetos (Diseñador R/O) ayuda a los programadores a crear y editar los objetos de LINQ to SQL que realizan asignaciones entre una aplicación y una base de datos. El Diseñador relacional de objetos crea el DataContext, las clases de entidad y los métodos DataContext que usa LINQ to SQL para comunicarse con la base de datos remota y controlar los datos que se usan en la aplicación.

    - Para abrir Diseñador relacional de objetos, agregue un elemento **LINQ a clases SQL** a un proyecto.

- La compatibilidad de n niveles para los conjuntos de datos con tipo proporciona mejoras para el **Diseñador de DataSet** que ayudan a separar código **TableAdapter** y código del conjunto de datos con tipo en proyectos adicionales.

    - Para separar código **TableAdapter** y código del conjunto de datos con tipo en proyectos adicionales, establezca la propiedad **DataSet Project** en el **Diseñador de DataSet**.

- Las funciones de actualización jerárquica se han incorporado en el **Diseñador de DataSet** para permitir guardar datos en varias tablas relacionadas. Se ha agregado un nuevo objeto **TableAdapterManager** a los conjuntos de datos con tipo que proporciona código generado que incluye la lógica necesaria de almacenamiento para mantener la integridad referencial. En lugar de llamar al método **Update** para cada **TableAdapter**, llame al método TableAdapterManager.UpdateAll.

    - De forma predeterminada, las actualizaciones jerárquicas están habilitadas para los conjuntos de datos que se crean en esta versión de Visual Studio. Puede controlar las actualizaciones jerárquicas configurando la propiedad **Actualización jerárquica** en el **Diseñador de DataSet**.

- El almacenamiento local en caché de bases de datos incorpora una base de datos de SQL Server Compact 3.5 y Microsoft Synchronization Services para ADO.NET en una aplicación y prepara la aplicación para que sincronice los datos periódicamente con una base de datos remota en un servidor. El almacenamiento en caché de bases de datos permite a las aplicaciones reducir el número de viajes de ida y vuelta entre la aplicación y un servidor de base de datos. Esto puede aumentar el rendimiento cuando se trabaja con datos que no se modifican con frecuencia o cuando las aplicaciones no siempre pueden conectarse a la base de datos remota.

    - Configure el almacenamiento en caché de bases de datos agregando una **Caché de base de datos local** a un proyecto.

- Microsoft SQL Server Compact 3.5 es una base de datos compacta que se puede implementar en equipos de escritorio, Smart Device y Tablet PC. SQL Server Compact 3.5 es una base de datos local que se incorpora en las aplicaciones y se implementa con facilidad.