Introducción

El examen Microsoft 70-483 se centra en una amplia gama de temas que puede utilizar en su trabajo como desarrollador de C#. Este libro le ayuda a comprender tanto las áreas básicas como las más avanzadas del lenguaje C#. Le muestra cómo usar el lenguaje C# para crear potentes aplicaciones de software. Este libro también le muestra cómo usar las nuevas funciones que se agregaron al lenguaje C#, como la compatibilidad con el código asíncrono. Este libro está dirigido a desarrolladores que tienen cierta experiencia con C# pero desean profundizar sus conocimientos y asegurarse de que estén listos para el examen. Para usar los ejemplos de este libro, debe estar familiarizado con el uso de Visual Studio para crear una aplicación de consola básica.

Este libro cubre todos los objetivos del examen, pero no cubre todas las preguntas del examen. Solo el equipo de examen de Microsoft tiene acceso a las preguntas del examen y Microsoft agrega regularmente nuevas preguntas al examen, lo que hace imposible cubrir preguntas específicas. Debe considerar este libro como un complemento de su experiencia relevante en el mundo real y otros materiales de estudio. Si encuentra un tema en este libro con el que no se siente del todo cómodo, use los enlaces para encontrar más información y tómese el tiempo para investigar y estudiar el tema. La gran información está disponible en MSDN, TechNet, en blogs y en foros.

# Certificaciones de Microsoft

Las certificaciones de Microsoft lo distinguen demostrando su dominio de un amplio conjunto de habilidades y experiencia con los productos y tecnologías actuales de Microsoft. Los exámenes y las certificaciones correspondientes se desarrollan para validar su dominio de competencias críticas a medida que diseña y desarrolla, o implementa y respalda, soluciones con productos y tecnologías de Microsoft tanto locales como en la nube. La certificación aporta una variedad de beneficios para el individuo y para los empleadores y organizaciones.

Quién debería leer este libro

Este libro está dirigido a desarrolladores que desean obtener la certificación para el lenguaje de programación C#. Este libro lo prepara para tomar y aprobar el examen 70-483: Programación en C#. La aprobación exitosa del examen 70-483 también cuenta como crédito para el Desarrollador de soluciones certificado de Microsoft (MCSD): aplicaciones de la Tienda Windows con C#. Los desarrolladores que lean este libro deben tener un conocimiento básico de cómo escribir un programa simple de C#. Puedes crear y ejecutar aplicaciones usando Visual Studio.

# Suposiciones

Debe tener al menos uno o más años de experiencia en la programación de la lógica empresarial / de aplicaciones esencial para una variedad de tipos de aplicaciones y plataformas de hardware / software usando C#. A medida que avance con su aprendizaje a través de este libro y otros recursos de estudio, se volverá competente en el desarrollo de aplicaciones complejas. Podrás usar todas las funciones que ofrece C#. Este libro está enfocado en ayudar a aquellos de ustedes cuyo objetivo es certificarse como desarrollador de C#.

Organización de este libro

Este libro está dividido en cuatro capítulos. Cada capítulo se centra en un dominio de examen diferente relacionado con el Examen 70-483: Programación en C#. Cada capítulo se divide en objetivos específicos del examen que ha sido publicado por Microsoft; se pueden encontrar en la sección "Habilidades que se miden" del examen 70-483: Programación en C#.

El material cubierto por el dominio del examen y los objetivos se han incorporado al libro para que tenga contenido conciso, objetivo por objetivo, junto con escenarios estratégicos del mundo real, experimentos mentales y preguntas de revisión al final del capítulo para proporcionar a los lectores. Con preparación a nivel profesional para el examen.

Convenciones y características de este libro.

Este libro presenta información utilizando convenciones diseñadas para hacer que la información sea legible y fácil de seguir:

■ Cada ejercicio consta de una serie de tareas, presentadas como pasos numerados que enumeran cada acción que debe realizar para completar el ejercicio.

■ Los elementos en la caja con etiquetas como "Nota" proporcionan información adicional o métodos alternativos para completar un paso con éxito.

■ Los elementos en la caja con etiquetas de "Sugerencia de examen" proporcionan información adicional que puede ofrecer sugerencias útiles o información adicional sobre qué esperar del examen.

■ El texto que escribe (aparte de los bloques de código) aparece en negrita.

■ Un signo más (+) entre dos nombres de teclas significa que debe presionar esas teclas al mismo tiempo. Por ejemplo, "Presionar Alt + Tabulador" significa que mantienes presionada la tecla Alt mientras presionas la tecla Tabulador.

■ Una barra vertical entre dos o más elementos de menú (por ejemplo, Archivo | Cerrar) significa que debe seleccionar el primer menú o elemento de menú, luego el siguiente, y así sucesivamente.

Expresiones de gratitud

Me gustaría agradecer a las siguientes personas:

■ A Jeff Riley por brindarme la oportunidad de escribir este libro.

■ A Ginny Munroe por ayudarme en todo el proceso de edición. Aprendí mucho de sus comentarios y consejos.

■ A Auri Rahimzadeh por sus habilidades de revisión técnica.

■ A mi esposa, Elise, por tu apoyo.

Y a todas las demás personas que jugaron un papel en preparar este libro. ¡Gracias por su arduo trabajo!

Preparando para el examen

Los exámenes de certificación de Microsoft son una excelente manera de elaborar su currículum y de hacerle saber al mundo su nivel de experiencia. Los exámenes de certificación validan su experiencia en el trabajo y el conocimiento del producto. Si bien no hay sustituciones para la experiencia en el trabajo, la preparación a través del estudio y la práctica pueden ayudarlo a prepararse para el examen. Recomendamos que complete el plan de preparación de su examen utilizando una combinación de materiales de estudio y cursos disponibles. Por ejemplo, puede usar este examen de referencia y otra guía de estudio para su preparación "en casa", y tomar un curso oficial de Microsoft para la experiencia en el aula. Elige la combinación que creas que funciona mejor para ti.

Tenga en cuenta que esta Referencia del examen se basa en información disponible públicamente sobre el examen y la experiencia del autor. Para salvaguardar la integridad del examen, los autores no tienen acceso al examen en vivo.

CAPÍTULO 1

ADMINISTRAR EL FLUJO DEL PROGRAMA

Si pudiera construir solo programas que ejecuten toda su lógica de arriba a abajo, no sería factible construir aplicaciones complejas. Afortunadamente, C# y .NET Framework le ofrecen muchas opciones para crear programas complejos que no tienen un flujo de programas fijo.

Este capítulo comienza analizando cómo crear aplicaciones de multiproceso. Esas aplicaciones pueden escalar bien y seguir respondiendo al usuario mientras realiza su trabajo. También verá la nueva característica del lenguaje async/await que se agregó a C# 5.

Aprenderá sobre las construcciones básicas del lenguaje C# para tomar decisiones y ejecutar un fragmento de código varias veces, según las circunstancias. Estas construcciones forman los bloques de lenguaje básicos de cada aplicación, y los usarás a menudo.

Después de eso, aprenderá a crear aplicaciones que se acoplan de forma flexible mediante el uso de delegados y eventos. Con los eventos, puede crear objetos que pueden notificarse entre ellos cuando algo sucede y que pueden responder a esas notificaciones. Los marcos como ASP.NET, Windows Presentation Foundation (WPF) y WinForms hacen un uso intensivo de los eventos; Entender los eventos a fondo te ayudará a construir grandes aplicaciones.

Desafortunadamente, el flujo de su programa también puede ser interrumpido por errores. Dichos errores pueden ocurrir en áreas que están fuera de su control pero a las que debe responder. A veces usted quiere levantar tal error usted mismo. Aprenderá cómo las excepciones pueden ayudarlo a implementar una sólida estrategia de manejo de errores en sus aplicaciones.

**Objetivos en este capítulo:**

■ Objetivo 1.1: Implementar multiprocesamiento y procesamiento asíncrono.

■ Objetivo 1.2: Gestionar multihilo.

■ Objetivo 1.3: Implementar el flujo del programa

■ Objetivo 1.4: Crear e implementar eventos y devoluciones de llamada

■ Objetivo 1.5. Implementar el manejo de excepciones.

Objetivo 1.1: Implementar multiprocesamiento y procesamiento asíncrono

Las aplicaciones son cada vez más complejas a medida que aumentan las expectativas de los usuarios. Para aprovechar al máximo los sistemas multinúcleo y mantener la capacidad de respuesta, debe crear aplicaciones que utilicen varios subprocesos, a menudo llamados paralelismo.

.NET Framework y el lenguaje C# ofrecen muchas opciones que puede usar para crear aplicaciones de multiproceso.

Este objetivo cubre lo siguiente:

■ Entender los hilos.

■ Usar la biblioteca paralela de tareas.

■ Usar la clase Parallel.

■ Utilice las nuevas palabras clave async y await.

■ Utilizar la Consulta Integrada de Lenguaje Paralelo.

■ Utilizar colecciones concurrentes.

# Entendiendo hilos

Imagine que su computadora tiene solo una unidad central de procesamiento (CPU) que es capaz de ejecutar solo una operación a la vez. Ahora, imagine lo que sucedería si la CPU tiene que trabajar duro para ejecutar una tarea que demore mucho tiempo. Mientras se ejecuta esta operación, todas las demás operaciones se detendrían. Esto significa que toda la máquina se congelaría y parecería que no responde al usuario.

Las cosas se ponen aún peor cuando esa operación de larga duración contiene un error por lo que nunca termina. Debido a que el resto de la máquina no se puede utilizar, lo único que puede hacer es reiniciar la máquina. Para remediar este problema, se utiliza el concepto de un hilo. En las versiones actuales de Windows, cada aplicación se ejecuta en su propio proceso. Un proceso aísla una aplicación de otras aplicaciones dándole su propia memoria virtual y asegurando que los diferentes procesos no puedan influirse entre sí. Cada proceso se ejecuta en su propio hilo. Un hilo es algo así como una CPU virtualizada. Si una aplicación se bloquea o llega a un bucle infinito, solo se ve afectado el proceso de la aplicación.

Windows debe administrar todos los subprocesos para garantizar que puedan hacer su trabajo. Estos gestionan cierto período de tiempo. Una vez finalizado este período, el subproceso se detiene y Windows cambia a otro subproceso. Esto se llama cambio de contexto. En la práctica, esto significa que Windows tiene que hacer algún trabajo para que esto suceda. El hilo actual está utilizando un área determinada de memoria; utiliza registros de la CPU y otros datos de estado, y Windows tiene que asegurarse de que todo el contexto del subproceso se guarda y se restaura en cada conmutador.

Pero, aunque hay ciertos éxitos de rendimiento, el uso de subprocesos garantiza que cada proceso tenga su tiempo de ejecución sin tener que esperar hasta que finalicen todas las demás operaciones. Esto mejora la capacidad de respuesta del sistema y da la ilusión de que una CPU puede ejecutar múltiples tareas a la vez. De esta manera, puede crear una aplicación que use paralelismo, lo que significa que puede ejecutar varios subprocesos en diferentes CPU en paralelo.

Casi cualquier dispositivo que compre hoy tiene una CPU con varios núcleos, que es similar a tener varias CPU. Algunos servidores no solo tienen CPU multinúcleo, sino que también tienen más de una CPU. Para hacer uso de todos estos núcleos, necesitas múltiples hilos. Windows garantiza que esos subprocesos se distribuyan a través de sus núcleos disponibles. De esta manera puede realizar múltiples tareas a la vez y mejorar la escalabilidad.

Debido a la sobrecarga asociada, debe determinar cuidadosamente si necesita multihilo. Pero si desea utilizar subprocesos para escalabilidad o capacidad de respuesta, C# y .NET Framework le ofrecen muchas posibilidades.

# Usando la clase Thread

La clase Thread se puede encontrar en el espacio de nombres System.Threading. Esta clase le permite crear nuevas huellas, administrar su prioridad y obtener su estado.

La clase Thread no es algo que debas usar en tus aplicaciones, excepto cuando tienes necesidades especiales. Sin embargo, cuando utiliza la clase Thread, tiene control sobre todas las opciones de configuración. Por ejemplo, puede especificar la prioridad de su hilo, decirle a Windows que su hilo es de larga duración o configurar otras opciones avanzadas.

El Listado 1-1 muestra un ejemplo del uso de la clase Thread para ejecutar un método en otro hilo. La clase de consola sincroniza el uso de la secuencia de salida para que pueda escribir desde múltiples subprocesos. La sincronización es el mecanismo para garantizar que dos subprocesos no ejecuten una parte específica de su programa al mismo tiempo. En el caso de una aplicación de consola, esto significa que no hay dos hilos que puedan escribir datos en la pantalla al mismo tiempo. Si un subproceso está trabajando con la secuencia de salida, otros deberán esperar antes de que finalice.

*[LISTADO 1-1 Creando un hilo con la clase Thread]*

Como puede ver, ambos hilos se ejecutan e imprimen su mensaje en la consola. Se llama al método Thread.Join en el subproceso principal para que espere hasta que finalice el otro subproceso.

¿Por qué el Thread.Sleep(0)? Se utiliza para indicar a Windows que este hilo está terminado. En lugar de esperar a que finalice la totalidad del segmento de tiempo del hilo, cambiará inmediatamente a otro hilo.

Tanto tu proceso como tu hilo tienen una prioridad. Asignar una prioridad baja es útil para aplicaciones como un protector de pantalla. Tal aplicación no debe competir con otras aplicaciones por el tiempo de CPU. Un subproceso de prioridad más alta debe usarse solo cuando sea absolutamente necesario. A un nuevo hilo se le asigna una prioridad de Normal, lo cual está bien para casi todos los escenarios.

Otra cosa que es importante saber acerca de los hilos es la diferencia entre los hilos de primer plano y de fondo. Los hilos de primer plano se pueden usar para mantener viva una aplicación. Solo cuando todos los hilos de primer plano terminan, el Common Language Runtime (CLR) cierra su aplicación. Luego se terminan los hilos de fondo.

El listado 1-2 muestra esta diferencia en acción.

*[LISTADO 1-2 Usando un hilo de fondo]*

El constructor Thread tiene otra sobrecarga que toma una instancia de un delegado ParameterizedThreadStart. Esta sobrecarga se puede usar si desea pasar algunos datos a través del método de inicio de su hilo a su método de trabajo, como muestra el Listado 1-3.

*[LISTADO 1-3 Usando el ParameterizedThreadStart]*

En este caso, el valor 5 se pasa a ThreadMethod como un objeto. Puede convertirlo en el tipo esperado para usarlo en su método.

Para detener un hilo, puedes usar el método Thread.Abort. Sin embargo, debido a que este método es ejecutado por otro hilo, puede ocurrir en cualquier momento. Cuando sucede, se lanza una excepción ThreadAbort en el subproceso de destino. Esto puede potencialmente dejar un estado corrupto y hacer que su aplicación sea inutilizable.

Una mejor manera de detener un hilo es mediante el uso de una variable compartida a la que pueden acceder tanto su destino como su hilo de llamada. El listado 1-4 muestra un ejemplo.

[*LISTADO 1-4 Deteniendo un hilo*]

En este caso, el hilo se inicializa con una expresión lambda (que a su vez es solo una versión abreviada de un delegado). El hilo sigue corriendo hasta que *stopped* se convierte en *true*. Después de eso, el método *t.Join* hace que la aplicación de la consola espere hasta que el hilo finalice la ejecución.

Un hilo tiene su propia pila de llamadas que almacena todos los métodos que se ejecutan. Las variables locales se almacenan en la pila de llamadas y son privadas para el hilo.

Un hilo también puede tener sus propios datos que no son una variable local. Al marcar un campo con el atributo *ThreadStatic*, cada subproceso obtiene su propia copia de un campo (consulte el Listado 1-5).

[*LISTADO 1-5 Uso del atributo ThreadStatic*]

Con el atributo *ThreadStaticAttribute* aplicado, el valor máximo de *\_field* se convierte en 10. Si lo elimina, puede ver que ambos subprocesos acceden al mismo valor y se convierte en 20.

Si desea usar datos locales en un hilo e inicializarlos para cada hilo, puede usar la clase *ThreadLocal<T>*. Esta clase lleva a un delegado a un método que inicializa el valor. El listado 1-6 muestra un ejemplo.

[*LISTADO 1-6 Usando ThreadLocal<T>*]