Estructuras de datos lineales

Microsoft

Developer Network

Tabla de Contenidos

Capítulo 1. Colecciones y estructuras de datos

Colecciones y estructuras de datos

.NET Framework (current version)

Publicado: octubre de 2016

A menudo, los datos similares pueden controlarse de forma más eficaz si se almacenan y manipulan como si fuesen una colección. Puede utilizar la clase System.Array o las clases de los espacios de nombres System.Collections, System.Collections.Generic, System.Collections.Concurrent y System.Collections.Immutable para agregar, quitar y modificar elementos individuales o un intervalo de elementos de una colección.

Hay dos tipos principales de colecciones: las colecciones genéricas y las colecciones no genéricas. Las colecciones genéricas se agregaron en la versión 2.0 de.NET Framework y son colecciones con seguridad de tipos en tiempo de compilación. Debido a esto, las colecciones genéricas normalmente ofrecen un mejor rendimiento. Las colecciones genéricas aceptan un parámetro de tipo cuando se construyen y no requieren conversiones con el tipo Object al agregar o quitar elementos de la colección. Además, la mayoría de colecciones genéricas son compatibles con aplicaciones de la Tienda Windows. Las colecciones no genéricas almacenan elementos como Object, requieren conversión y la mayoría no son compatibles con el desarrollo de aplicaciones de la Tienda Windows. Sin embargo, puede que vea colecciones no genéricas en código antiguo.

A partir de .NET Framework 4, las colecciones del espacio de nombres System.Collections.Concurrent proporcionan operaciones eficaces y seguras para subprocesos con el fin de obtener acceso a los elementos de la colección desde varios subprocesos. Las clases de colección inmutables en el espacio de nombres System.Collections.Immutable (paquete de NuGet) son intrínsecamente seguras para los subprocesos, ya que las operaciones se realizan en una copia de la colección original, mientras que la colección original no se puede modificar.

Características comunes de las colecciones

Todas las colecciones ofrecen métodos para agregar, quitar o buscar elementos en la colección. Además, todas las colecciones que implementan directa o indirectamente las interfaces ICollection o ICollection<T> comparten estas características:

- **Capacidad para enumerar la colección**

  Las colecciones de .NET Framework implementan System.Collections.IEnumerable o System.Collections.Generic.IEnumerable<T> para permitir procesar una iteración en la colección. Un enumerador puede considerarse como un puntero móvil para cualquier elemento de la colección. La instrucción foreach, in (Referencia de C#) y Instrucción For Each...Next (Visual Basic) usan el enumerador expuesto por el método GetEnumerator y ocultan la complejidad que supone manipular el enumerador. Además, cualquier colección que implementa System.Collections.Generic.IEnumerable<T> se considera un *tipo consultable* y se puede consultar con LINQ. Las consultas LINQ proporcionan un modelo común para acceder a los datos. Por lo general, son más concisas y legibles que los bucles **foreach** estándar y ofrecen capacidad de filtrado, ordenación y agrupación. Las consultas LINQ también pueden mejorar el rendimiento. Para obtener más información, vea LINQ to Objects, Parallel LINQ (PLINQ) y Introduction to LINQ Queries (C#).

- **Capacidad de copiar el contenido de la colección en una matriz**

  Todas las colecciones se pueden copiar en una matriz mediante el método **CopyTo**; sin embargo, el orden de los elementos de la nueva matriz se basa en la secuencia en la que los devuelve el enumerador. La matriz resultante siempre es unidimensional con un límite inferior de cero.

Además, muchas clases de colecciones contienen las siguientes características:

- **Propiedades de capacidad y recuento**

  La capacidad de una colección es el número de elementos que puede contener. El recuento de una colección es el número de elementos que realmente contiene. Algunas colecciones ocultan la capacidad, el recuento, o ambos.

  La mayoría de las colecciones expanden automáticamente su capacidad cuando se alcanza la capacidad actual. La memoria se reasigna y los elementos de la antigua colección se copian en la nueva. Esto reduce el código necesario para utilizar la colección; sin embargo, el

rendimiento de la colección podría verse afectado negativamente. Por ejemplo, en List<T>, si Count es menor que Capacity, el agregar un elemento supone una operación O(1). Si es necesario aumentar la capacidad para alojar el nuevo elemento, agregar un elemento se convierte en una operación O(n), donde n es Count. La mejor manera de evitar el rendimiento deficiente provocado por múltiples reasignaciones es establecer la capacidad inicial el tamaño estimado de la colección.

BitArray es un caso especial; su capacidad es igual que su longitud, que es la misma que su recuento.

- **Límite inferior coherente**

  El límite inferior de una colección es el índice de su primer elemento. Todas las colecciones indizadas en el espacio de nombres System.Collections tienen un límite inferior de cero, lo que significa que están indizadas en 0. De forma predeterminada, Array tiene un límite inferior de cero, pero se puede definir un límite inferior diferente al crear una instancia de la clase **Array** con Array.CreateInstance.

- **Sincronización para el acceso de varios subprocesos** (solo clases System.Collections).

  Los tipos de colecciones no genéricas del espacio de nombres System.Collections proporcionan una seguridad de subprocesos con sincronización; normalmente se exponen a través de los miembros SyncRoot y IsSynchronized. Estas colecciones no son seguras para subprocesos de forma predeterminada. Si necesita un acceso multiproceso escalable y eficaz a una colección, utilice una de las clases del espacio de nombres System.Collections.Concurrent o considere el uso de una colección inmutable. Para obtener más información, consulta Colecciones seguras para subprocesos.

Elegir una colección

En general, debería utilizar colecciones genéricas. En la tabla siguiente se describen algunos escenarios habituales de las colecciones y las clases de colección que puede utilizar en esos escenarios. Si es la primera vez que usa colecciones genéricas, con esta tabla le será más fácil elegir la colección genérica que funciona mejor para su tarea.

Deseo...	Opciones de colección genérica	Opciones de colección no genérica	Opciones de colección de subprocesos o inmutable
Almacenar elementos como pares clave/valor para una consulta rápida por clave	System.Collections.Generic.Dictionary<TKey, TValue>	Hashtable (Colección de pares clave/valor	System.Collections.Concurrent.ConcurrentDictionary<TKey, TValue> System.Collections.Object

		que se organizan en función del código hash de la clave).	Model.ReadOnlyDictionary<TKey, TValue> ImmutableDictionary(TKey, TValue) Clase
Acceso a elementos por índice	System.Collections.Generic.List<T>	System.Array System.Collections.ArrayList	ImmutableList(T) Clase ImmutableArray Clase
Utilizar elementos FIFO (el primero en entrar es el primero en salir)	System.Collections.Generic.Queue<T>	System.Collections.Queue	System.Collections.Concurrent.ConcurrentQueue<T> ImmutableQueue(T) Clase
Utilizar datos LIFO (el último en entrar es el primero en salir)	System.Collections.Generic.Stack<T>	System.Collections.Stack	System.Collections.Concurrent.ConcurrentStack<T> ImmutableStack(T) Clase
Acceso a elementos de forma secuencial	System.Collections.Generic.LinkedList<T>	Sin recomendación	Sin recomendación
Recibir notificaciones cuando se quitan o se agregan elementos a la colección. (implementa INotifyPropertyChanged y System.Collections.Specialized.INotifyCollectionChanged)	System.Collections.ObjectModel.ObservableCollection<T>	Sin recomendación	Sin recomendación
Una colección ordenada	System.Collections.Generic.SortedList<TKey, TValue>	System.Collections.SortedList	ImmutableSortedDictionary(TKey, TValue) Clase

			ImmutableSortedSet(T) Clase
Un conjunto de funciones matemáticas	System.Collections.Generic.HashSet<T>	Sin recomendación	ImmutableHashSet(T) Clase
	System.Collections.Generic.SortedSet<T>		ImmutableSortedSet(T) Clase

Temas relacionados

Título	Descripción
Seleccionar una clase de colección	Describe las diferentes colecciones y le ayuda a seleccionar una para su escenario.
Tipos de colección utilizados normalmente	Describe los tipos de colección genéricos y no genéricos más utilizados, como System.Array, System.Collections.Generic.List<T> y System.Collections.Generic.Dictionary<TKey, TValue>.
Cuándo utilizar colecciones genéricas	Describe el uso de los tipos de colección genéricos.
Comparaciones y ordenaciones en colecciones	Describe el uso de las comparaciones de igualdad y ordenación en las colecciones.
Tipos de colecciones ordenadas	Describe las características y el funcionamiento de colecciones ordenadas.
Tipos de las colecciones Hashtable y Dictionary	Describe las características de los tipos de diccionarios basados en hash genéricos y no genéricos.
Colecciones seguras para	Describe los tipos de colección, como

subprocesos	System.Collections.Concurrent.BlockingCollection<T> y System.Collections.Concurrent.ConcurrentBag<T>, que admiten un acceso simultáneo seguro y eficaz desde varios subprocesos.
System.Collections.Immutable	Presenta las colecciones inalterables y proporciona vínculos a los tipos de colección.

Referencia

System.Array
System.Collections
System.Collections.Concurrent
System.Collections.Generic
System.Collections.Specialized
System.Linq
System.Collections.Immutable

Capítulo 2. Pilas - Stack(T)

Clase Stack<T>

.NET Framework (current version)
Publicado: octubre de 2016

Representa una colección última en entrar, primero en salir (LIFO) de tamaño variable con instancias del mismo tipo especificado.

Espacio de nombres: System.Collections.Generic
Ensamblado: System (en System.dll)

Jerarquía de herencia

System.Object
  System.Collections.Generic.Stack<T>

Sintaxis

C#
```
[SerializableAttribute]
[ComVisibleAttribute(false)]
public class Stack<T> : IEnumerable<T>, IEnumerable, ICollection,
        IReadOnlyCollection<T>
```

Parámetros de tipo

T

Especifica el tipo de elementos de la pila.

Constructores

Nombre	Descripción
Stack<T>()	Inicializa una nueva instancia de la clase Stack<T> que está vacía y tiene la capacidad inicial predeterminada.
Stack<T>(IEnumerable<T>)	Inicializa una nueva instancia de la clase Stack<T> que contiene elementos copiados de la colección especificada y tiene una capacidad suficiente para aceptar el número de elementos copiados.

Stack<T>(Int32)	Inicializa una nueva instancia de la clase Stack<T> que está vacía y tiene la capacidad inicial especificada o la capacidad inicial predeterminada, la que sea mayor.

Propiedades

Nombre	Descripción
Count	Obtiene el número de elementos incluidos en Stack<T>.

Métodos

Nombre	Descripción
Clear()	Quita todos los objetos de la colección Stack<T>.
Contains(T)	Determina si un elemento se encuentra en Stack<T>.
CopyTo(T[], Int32)	Copia Stack<T> en una Array unidimensional existente, a partir del índice especificado de la matriz.
Equals(Object)	Determina si el objeto especificado es igual al objeto actual.(Heredado de Object).
Finalize()	Permite que un objeto intente liberar recursos y realizar otras operaciones de limpieza antes de ser reclamado por el recolector de basura. (Heredado de Object).
GetEnumerator()	Devuelve un enumerador para la colección Stack<T>.
GetHashCode()	Sirve como la función hash predeterminada.(Heredado de Object).
GetType()	Obtiene el Type de la instancia actual.(Heredado de Object).

MemberwiseClone()	Crea una copia superficial del Object actual.(Heredado de Object).
Peek()	Devuelve el objeto situado al principio de Stack<T> sin eliminarlo.
Pop()	Quita y devuelve el objeto situado al principio de Stack<T>.
Push(T)	Inserta un objeto al principio de Stack<T>.
ToArray()	Copia Stack<T> en una nueva matriz.
ToString()	Devuelve una cadena que representa al objeto actual. (Heredado de Object).
TrimExcess()	Establece la capacidad en el número real de elementos de Stack<T>, si este número supone menos del 90 por ciento de la capacidad actual.

Implementaciones de interfaz explícitas

Nombre	Descripción
IEnumerable<T>.GetEnumerator()	Devuelve un enumerador que procesa una iteración en la colección.
ICollection.CopyTo(Array, Int32)	Copia los elementos de ICollection en Array, empezando por un índice determinado de Array.
IEnumerable.GetEnumerator()	Devuelve un enumerador que recorre en iteración una colección.
ICollection.IsSynchronized	Obtiene un valor que indica si el acceso a la interfaz ICollection está sincronizado (es seguro para subprocesos).
ICollection.SyncRoot	Obtiene un objeto que se puede usar para sincronizar el

acceso a ICollection.

Métodos de extensión

Nombre	Descripción
Aggregate<T>(Func<T, T, T>)	Sobrecargado. Aplica una función de acumulador a una secuencia.(Definido por Enumerable).
Aggregate<T, TAccumulate>(TAccumulate, Func<TAccumulate, T, TAccumulate>)	Sobrecargado. Aplica una función de acumulador a una secuencia. El valor de inicialización especificado se utiliza como valor de inicio del acumulador.(Definido por Enumerable).
Aggregate<T, TAccumulate, TResult>(TAccumulate, Func<TAccumulate, T, TAccumulate>, Func<TAccumulate, TResult>)	Sobrecargado. Aplica una función de acumulador a una secuencia. El valor de inicialización especificado se utiliza como valor inicial del acumulador y la función especificada se utiliza para seleccionar el valor resultante.(Definido por Enumerable).
All<T>(Func<T, Boolean>)	Determina si todos los elementos de una secuencia satisfacen una condición.(Definido por Enumerable).
Any<T>()	Sobrecargado. Determina si una secuencia contiene elementos.(Definido por Enumerable).
Any<T>(Func<T, Boolean>)	Sobrecargado. Determina si un elemento de una secuencia satisface una condición.(Definido por Enumerable).
AsEnumerable<T>()	Devuelve la entrada de tipo IEnumerable<T>. (Definido por Enumerable).
AsParallel()	Sobrecargado. Habilita la paralelización de una

	consulta.(Definido por ParallelEnumerable).
AsParallel<T>()	Sobrecargado. Habilita la paralelización de una consulta.(Definido por ParallelEnumerable).
AsQueryable()	Sobrecargado. Convierte un IEnumerable para un IQueryable.(Definido por Queryable).
AsQueryable<T>()	Sobrecargado. Convierte un tipo genérico IEnumerable<T> a un tipo genérico IQueryable<T>.(Definido por Queryable).
Average<T>(Func<T, Decimal>)	Sobrecargado. Calcula el promedio de una secuencia de Decimal valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Average<T>(Func<T, Double>)	Sobrecargado. Calcula el promedio de una secuencia de Double valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Average<T>(Func<T, Int32>)	Sobrecargado. Calcula el promedio de una secuencia de Int32 valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Average<T>(Func<T, Int64>)	Sobrecargado. Calcula el promedio de una secuencia de Int64 valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Average<T>(Func<T, Nullable<Decimal>>)	Sobrecargado. Calcula el promedio de una

	secuencia de nullable Decimal valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Average<T>(Func<T, Nullable<Double>>)	Sobrecargado. Calcula el promedio de una secuencia de nullable Double valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Average<T>(Func<T, Nullable<Int32>>)	Sobrecargado. Calcula el promedio de una secuencia de nullable Int32 valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Average<T>(Func<T, Nullable<Int64>>)	Sobrecargado. Calcula el promedio de una secuencia de nullable Int64 valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Average<T>(Func<T, Nullable<Single>>)	Sobrecargado. Calcula el promedio de una secuencia de nullable Single valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Average<T>(Func<T, Single>)	Sobrecargado. Calcula el promedio de una secuencia de Single valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Cast<TResult>()	Convierte los elementos de un IEnumerable al tipo especificado.(Definido por Enumerable).

Concat<T>(IEnumerable<T>)	Concatena dos secuencias.(Definido por Enumerable).
Contains<T>(T)	Sobrecargado. Determina si una secuencia contiene un elemento especificado mediante el comparador de igualdad predeterminado. (Definido por Enumerable).
Contains<T>(T, IEqualityComparer<T>)	Sobrecargado. Determina si una secuencia contiene un elemento especificado mediante el uso de un objeto IEqualityComparer<T>. (Definido por Enumerable).
Count<T>()	Sobrecargado. Devuelve el número de elementos de una secuencia.(Definido por Enumerable).
Count<T>(Func<T, Boolean>)	Sobrecargado. Devuelve un número que representa cuántos elementos de la secuencia especificada satisfacen una condición.(Definido por Enumerable).
DefaultIfEmpty<T>()	Sobrecargado. Devuelve los elementos de la secuencia especificada o el valor predeterminado del parámetro de tipo en una colección singleton si la secuencia está vacía.(Definido por Enumerable).
DefaultIfEmpty<T>(T)	Sobrecargado. Devuelve los elementos de la secuencia especificada o el valor especificado en una colección singleton si la secuencia está vacía. (Definido por Enumerable).
Distinct<T>()	Sobrecargado. Devuelve diversos elementos de una secuencia utilizando el comparador de igualdad predeterminado para comparar valores. (Definido por Enumerable).

Distinct<T>(IEqualityComparer<T>)	Sobrecargado. Devuelve diversos elementos de una secuencia utilizando un IEqualityComparer<T> para comparar valores. (Definido por Enumerable).
ElementAt<T>(Int32)	Devuelve el elemento en un índice especificado en una secuencia.(Definido por Enumerable).
ElementAtOrDefault<T>(Int32)	Devuelve el elemento situado en un índice especificado de una secuencia o un valor predeterminado si el índice está fuera del intervalo.(Definido por Enumerable).
Except<T>(IEnumerable<T>)	Sobrecargado. Proporciona la diferencia de conjunto de dos secuencias utilizando el comparador de igualdad predeterminado para comparar valores.(Definido por Enumerable).
Except<T>(IEnumerable<T>, IEqualityComparer<T>)	Sobrecargado. Proporciona la diferencia de conjuntos de dos secuencias mediante especificado IEqualityComparer<T> para comparar valores.(Definido por Enumerable).
First<T>()	Sobrecargado. Devuelve el primer elemento de una secuencia.(Definido por Enumerable).
First<T>(Func<T, Boolean>)	Sobrecargado. Devuelve el primer elemento de una secuencia que satisface una condición especificada.(Definido por Enumerable).
FirstOrDefault<T>()	Sobrecargado. Devuelve el primer elemento de una secuencia o un valor predeterminado si la secuencia no contiene elementos.(Definido por Enumerable).
FirstOrDefault<T>(Func<T, Boolean>)	Sobrecargado. Devuelve el primer elemento de la secuencia que satisface una condición o un valor

	predeterminado si no se encuentra dicho elemento.(Definido por Enumerable).
GroupBy<T, TKey>(Func<T, TKey>)	Sobrecargado. Agrupa los elementos de una secuencia según una función del selector de claves especificada.(Definido por Enumerable).
GroupBy<T, TKey>(Func<T, TKey>, IEqualityComparer<TKey>)	Sobrecargado. Agrupa los elementos de una secuencia según una función del selector de claves especificada y compara las claves utilizando un comparador especificado.(Definido por Enumerable).
GroupBy<T, TKey, TElement>(Func<T, TKey>, Func<T, TElement>)	Sobrecargado. Agrupa los elementos de una secuencia según una función del selector de claves especificada y proyecta los elementos de cada grupo utilizando una función especificada. (Definido por Enumerable).
GroupBy<T, TKey, TElement>(Func<T, TKey>, Func<T, TElement>, IEqualityComparer<TKey>)	Sobrecargado. Agrupa los elementos de una secuencia según una función del selector de claves. Las claves se comparan utilizando un comparador y los elementos de cada grupo se proyectan utilizando una función especificada. (Definido por Enumerable).
GroupBy<T, TKey, TResult>(Func<T, TKey>, Func<TKey, IEnumerable<T>, TResult>)	Sobrecargado. Agrupa los elementos de una secuencia según una función del selector de claves especificada y crea un valor de resultado a partir de cada grupo y su clave.(Definido por Enumerable).
GroupBy<T, TKey, TResult>(Func<T, TKey>, Func<TKey, IEnumerable<T>, TResult>, IEqualityComparer<TKey>)	Sobrecargado. Agrupa los elementos de una secuencia según una función del selector de claves especificada y crea un valor de resultado a partir de cada grupo y su clave. Las claves se comparan utilizando un comparador especificado. (Definido por Enumerable).

GroupBy<T, TKey, TElement, TResult>(Func<T, TKey>, Func<T, TElement>, Func<TKey, IEnumerable<TElement>, TResult>)	Sobrecargado. Agrupa los elementos de una secuencia según una función del selector de claves especificada y crea un valor de resultado a partir de cada grupo y su clave. Los elementos de cada grupo se proyectan utilizando una función determinada.(Definido por Enumerable).
GroupBy<T, TKey, TElement, TResult>(Func<T, TKey>, Func<T, TElement>, Func<TKey, IEnumerable<TElement>, TResult>, IEqualityComparer<TKey>)	Sobrecargado. Agrupa los elementos de una secuencia según una función del selector de claves especificada y crea un valor de resultado a partir de cada grupo y su clave. Los valores de las claves se comparan utilizando un comparador especificado y los elementos de cada grupo se proyectan utilizando una función especificada. (Definido por Enumerable).
GroupJoin<T, TInner, TKey, TResult>(IEnumerable<TInner>, Func<T, TKey>, Func<TInner, TKey>, Func<T, IEnumerable<TInner>, TResult>)	Sobrecargado. Correlaciona los elementos de dos secuencias basándose en la igualdad de claves y agrupa los resultados. El comparador de igualdad predeterminado se usa para comparar claves. (Definido por Enumerable).
GroupJoin<T, TInner, TKey, TResult>(IEnumerable<TInner>, Func<T, TKey>, Func<TInner, TKey>, Func<T, IEnumerable<TInner>, TResult>, IEqualityComparer<TKey>)	Sobrecargado. Correlaciona los elementos de dos secuencias basándose en la igualdad de claves y agrupa los resultados. Se usa un IEqualityComparer<T> especificado para comparar claves.(Definido por Enumerable).
Intersect<T>(IEnumerable<T>)	Sobrecargado. Proporciona la intersección de conjunto de dos secuencias utilizando el comparador de igualdad predeterminado para comparar valores.(Definido por Enumerable).
Intersect<T>(IEnumerable<T>, IEqualityComparer<T>)	Sobrecargado. Proporciona la intersección de conjuntos de dos secuencias mediante especificado IEqualityComparer<T> para comparar valores.(Definido por Enumerable).
Join<T, TInner, TKey, TResult>(IEnumerable<TInner>, Func<T,	Sobrecargado. Correlaciona los elementos de dos

TKey>, Func<TInner, TKey>, Func<T, TInner, TResult>)	secuencias basadas en claves coincidentes. El comparador de igualdad predeterminado se usa para comparar claves.(Definido por Enumerable).
Join<T, TInner, TKey, TResult>(IEnumerable<TInner>, Func<T, TKey>, Func<TInner, TKey>, Func<T, TInner, TResult>, IEqualityComparer<TKey>)	Sobrecargado. Correlaciona los elementos de dos secuencias basadas en claves coincidentes. Se usa un IEqualityComparer<T> especificado para comparar claves.(Definido por Enumerable).
Last<T>()	Sobrecargado. Devuelve el último elemento de una secuencia.(Definido por Enumerable).
Last<T>(Func<T, Boolean>)	Sobrecargado. Devuelve el último elemento de una secuencia que satisface una condición especificada.(Definido por Enumerable).
LastOrDefault<T>()	Sobrecargado. Devuelve el último elemento de una secuencia o un valor predeterminado si la secuencia no contiene elementos.(Definido por Enumerable).
LastOrDefault<T>(Func<T, Boolean>)	Sobrecargado. Devuelve el último elemento de una secuencia que satisface una condición o un valor predeterminado si no se encuentra dicho elemento.(Definido por Enumerable).
LongCount<T>()	Sobrecargado. Devuelve un Int64 que representa el número total de elementos de una secuencia. (Definido por Enumerable).
LongCount<T>(Func<T, Boolean>)	Sobrecargado. Devuelve un Int64 que representa el número de elementos de una secuencia satisface una condición.(Definido por Enumerable).
Max<T>()	Sobrecargado. Devuelve el valor máximo de una

	secuencia genérica.(Definido por Enumerable).
Max<T>(Func<T, Decimal>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo Decimal valor.(Definido por Enumerable).
Max<T>(Func<T, Double>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo Double valor.(Definido por Enumerable).
Max<T>(Func<T, Int32>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo Int32 valor. (Definido por Enumerable).
Max<T>(Func<T, Int64>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo Int64 valor. (Definido por Enumerable).
Max<T>(Func<T, Nullable<Decimal>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo que acepta valores NULL Decimal valor.(Definido por Enumerable).
Max<T>(Func<T, Nullable<Double>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo que acepta valores NULL Double valor.(Definido por Enumerable).
Max<T>(Func<T, Nullable<Int32>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo que acepta valores NULL Int32 valor.(Definido por

	Enumerable).
Max<T>(Func<T, Nullable<Int64>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo que acepta valores NULL Int64 valor.(Definido por Enumerable).
Max<T>(Func<T, Nullable<Single>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo que acepta valores NULL Single valor.(Definido por Enumerable).
Max<T>(Func<T, Single>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo Single valor.(Definido por Enumerable).
Max<T, TResult>(Func<T, TResult>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia genérica y devuelve el valor máximo resultante.(Definido por Enumerable).
Min<T>()	Sobrecargado. Devuelve el valor mínimo de una secuencia genérica.(Definido por Enumerable).
Min<T>(Func<T, Decimal>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo Decimal valor. (Definido por Enumerable).
Min<T>(Func<T, Double>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo Double valor. (Definido por Enumerable).

Min<T>(Func<T, Int32>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo Int32 valor. (Definido por Enumerable).
Min<T>(Func<T, Int64>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo Int64 valor. (Definido por Enumerable).
Min<T>(Func<T, Nullable<Decimal>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo que acepta valores NULL Decimal valor.(Definido por Enumerable).
Min<T>(Func<T, Nullable<Double>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo que acepta valores NULL Double valor.(Definido por Enumerable).
Min<T>(Func<T, Nullable<Int32>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo que acepta valores NULL Int32 valor.(Definido por Enumerable).
Min<T>(Func<T, Nullable<Int64>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo que acepta valores NULL Int64 valor.(Definido por Enumerable).
Min<T>(Func<T, Nullable<Single>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo que acepta valores NULL Single valor.(Definido por

	Enumerable).
Min<T>(Func<T, Single>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo Single valor. (Definido por Enumerable).
Min<T, TResult>(Func<T, TResult>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia genérica y devuelve el valor mínimo resultante.(Definido por Enumerable).
OfType<TResult>()	Filtra los elementos de un IEnumerable basado en un tipo especificado.(Definido por Enumerable).
OrderBy<T, TKey>(Func<T, TKey>)	Sobrecargado. Ordena los elementos de una secuencia en orden ascendente según una clave. (Definido por Enumerable).
OrderBy<T, TKey>(Func<T, TKey>, IComparer<TKey>)	Sobrecargado. Ordena los elementos de una secuencia en orden ascendente mediante un comparador especificado.(Definido por Enumerable).
OrderByDescending<T, TKey>(Func<T, TKey>)	Sobrecargado. Ordena los elementos de una secuencia en orden descendente según una clave. (Definido por Enumerable).
OrderByDescending<T, TKey>(Func<T, TKey>, IComparer<TKey>)	Sobrecargado. Ordena de manera descendente los elementos de una secuencia utilizando un comparador especificado.(Definido por Enumerable).
Reverse<T>()	Invierte el orden de los elementos de una secuencia.(Definido por Enumerable).

Select<T, TResult>(Func<T, TResult>)	Sobrecargado. Proyecta cada elemento de una secuencia a un nuevo formulario.(Definido por Enumerable).
Select<T, TResult>(Func<T, Int32, TResult>)	Sobrecargado. Proyecta cada elemento de una secuencia en un nuevo formulario mediante la incorporación del índice del elemento.(Definido por Enumerable).
SelectMany<T, TResult>(Func<T, IEnumerable<TResult>>)	Sobrecargado. Proyecta cada elemento de una secuencia en un IEnumerable<T> y reduce las secuencias resultantes en una secuencia.(Definido por Enumerable).
SelectMany<T, TResult>(Func<T, Int32, IEnumerable<TResult>>)	Sobrecargado. Proyecta cada elemento de una secuencia en un IEnumerable<T>, y reduce las secuencias resultantes en una secuencia. El índice de cada elemento de origen se utiliza en el formulario proyectado de ese elemento.(Definido por Enumerable).
SelectMany<T, TCollection, TResult>(Func<T, IEnumerable<TCollection>>, Func<T, TCollection, TResult>)	Sobrecargado. Proyecta cada elemento de una secuencia en IEnumerable<T>, reduce las secuencias resultantes en una única secuencia e invoca una función del selector de resultados en cada elemento.(Definido por Enumerable).
SelectMany<T, TCollection, TResult>(Func<T, Int32, IEnumerable<TCollection>>, Func<T, TCollection, TResult>)	Sobrecargado. Proyecta cada elemento de una secuencia en IEnumerable<T>, reduce las secuencias resultantes en una única secuencia e invoca una función del selector de resultados en cada elemento. El índice de cada elemento de origen se utiliza en el formulario proyectado intermedio de ese elemento.(Definido por Enumerable).
SequenceEqual<T>(IEnumerable<T>)	Sobrecargado. Determina si dos secuencias son iguales al comparar los elementos mediante el comparador de igualdad predeterminado para su

	tipo.(Definido por Enumerable).
SequenceEqual<T>(IEnumerable<T>, IEqualityComparer<T>)	Sobrecargado. Determina si dos secuencias son iguales al comparar sus elementos mediante un objeto IEqualityComparer<T>.(Definido por Enumerable).
Single<T>()	Sobrecargado. Devuelve el único elemento de una secuencia y produce una excepción si no hay exactamente un elemento en la secuencia. (Definido por Enumerable).
Single<T>(Func<T, Boolean>)	Sobrecargado. Devuelve el único elemento de una secuencia que cumpla una condición especificada y produce una excepción si existe más de un elemento de este tipo.(Definido por Enumerable).
SingleOrDefault<T>()	Sobrecargado. Devuelve el único elemento de una secuencia o un valor predeterminado si la secuencia está vacía; Este método produce una excepción si hay más de un elemento en la secuencia.(Definido por Enumerable).
SingleOrDefault<T>(Func<T, Boolean>)	Sobrecargado. Devuelve el único elemento de una secuencia que cumpla la condición especificada, o bien, un valor predeterminado si ese elemento no existe; este método produce una excepción si varios elementos cumplen la condición.(Definido por Enumerable).
Skip<T>(Int32)	Omite un número especificado de elementos de una secuencia y, a continuación, devuelve los elementos restantes.(Definido por Enumerable).
SkipWhile<T>(Func<T, Boolean>)	Sobrecargado. Omite los elementos de una secuencia siempre que una condición especificada sea true y, a continuación, devuelve los elementos

	restantes.(Definido por Enumerable).
SkipWhile<T>(Func<T, Int32, Boolean>)	Sobrecargado. Omite los elementos de una secuencia siempre que una condición especificada sea true y, a continuación, devuelve los elementos restantes. El índice del elemento se usa en la lógica de la función de predicado.(Definido por Enumerable).
Sum<T>(Func<T, Decimal>)	Sobrecargado. Calcula la suma de la secuencia de Decimal valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada. (Definido por Enumerable).
Sum<T>(Func<T, Double>)	Sobrecargado. Calcula la suma de la secuencia de Double valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada. (Definido por Enumerable).
Sum<T>(Func<T, Int32>)	Sobrecargado. Calcula la suma de la secuencia de Int32 valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada. (Definido por Enumerable).
Sum<T>(Func<T, Int64>)	Sobrecargado. Calcula la suma de la secuencia de Int64 valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada. (Definido por Enumerable).
Sum<T>(Func<T, Nullable<Decimal>>)	Sobrecargado. Calcula la suma de la secuencia de nullable Decimal valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).

Sum<T>(Func<T, Nullable<Double>>)	Sobrecargado. Calcula la suma de la secuencia de nullable Double valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada. (Definido por Enumerable).
Sum<T>(Func<T, Nullable<Int32>>)	Sobrecargado. Calcula la suma de la secuencia de nullable Int32 valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada. (Definido por Enumerable).
Sum<T>(Func<T, Nullable<Int64>>)	Sobrecargado. Calcula la suma de la secuencia de nullable Int64 valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada. (Definido por Enumerable).
Sum<T>(Func<T, Nullable<Single>>)	Sobrecargado. Calcula la suma de la secuencia de nullable Single valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada. (Definido por Enumerable).
Sum<T>(Func<T, Single>)	Sobrecargado. Calcula la suma de la secuencia de Single valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada. (Definido por Enumerable).
Take<T>(Int32)	Devuelve un número especificado de elementos contiguos desde el principio de una secuencia. (Definido por Enumerable).
TakeWhile<T>(Func<T, Boolean>)	Sobrecargado. Devuelve los elementos de una secuencia siempre que una condición especificada sea true.(Definido por Enumerable).

TakeWhile<T>(Func<T, Int32, Boolean>)	Sobrecargado. Devuelve los elementos de una secuencia siempre que una condición especificada sea true. El índice del elemento se usa en la lógica de la función de predicado.(Definido por Enumerable).
ToArray<T>()	Crea una matriz a partir de un IEnumerable<T>. (Definido por Enumerable).
ToDictionary<T, TKey>(Func<T, TKey>)	Sobrecargado. Crea un Dictionary<TKey, TValue> de un IEnumerable<T> según una función del selector de claves especificada. (Definido por Enumerable).
ToDictionary<T, TKey>(Func<T, TKey>, IEqualityComparer<TKey>)	Sobrecargado. Crea un Dictionary<TKey, TValue> de un IEnumerable<T> según un comparador de función y la clave del selector de claves especificada.(Definido por Enumerable).
ToDictionary<T, TKey, TElement>(Func<T, TKey>, Func<T, TElement>)	Sobrecargado. Crea un Dictionary<TKey, TValue> de un IEnumerable<T> según el selector de claves especificada y las funciones del selector de elementos.(Definido por Enumerable).
ToDictionary<T, TKey, TElement>(Func<T, TKey>, Func<T, TElement>, IEqualityComparer<TKey>)	Sobrecargado. Crea un Dictionary<TKey, TValue> de un IEnumerable<T> según una función del selector de claves especificada, un comparador y una función del selector de elementos.(Definido por Enumerable).
ToList<T>()	Crea un List<T> a partir de un IEnumerable<T>. (Definido por Enumerable).
ToLookup<T, TKey>(Func<T, TKey>)	Sobrecargado. Crea un Lookup<TKey, TElement> de un IEnumerable<T> según una función del selector de claves especificada. (Definido por Enumerable).

ToLookup<T, TKey>(Func<T, TKey>, IEqualityComparer<TKey>)	Sobrecargado. Crea un Lookup<TKey, TElement> de un IEnumerable<T> según un comparador de función y la clave del selector de claves especificada.(Definido por Enumerable).
ToLookup<T, TKey, TElement>(Func<T, TKey>, Func<T, TElement>)	Sobrecargado. Crea un Lookup<TKey, TElement> de un IEnumerable<T> según el selector de claves especificada y las funciones del selector de elementos.(Definido por Enumerable).
ToLookup<T, TKey, TElement>(Func<T, TKey>, Func<T, TElement>, IEqualityComparer<TKey>)	Sobrecargado. Crea un Lookup<TKey, TElement> de un IEnumerable<T> según una función del selector de claves especificada, un comparador y una función del selector de elementos.(Definido por Enumerable).
Union<T>(IEnumerable<T>)	Sobrecargado. Proporciona la unión de conjunto de dos secuencias utilizando el comparador de igualdad predeterminado.(Definido por Enumerable).
Union<T>(IEnumerable<T>, IEqualityComparer<T>)	Sobrecargado. Proporciona la unión de conjuntos de dos secuencias mediante un objeto IEqualityComparer<T>.(Definido por Enumerable).
Where<T>(Func<T, Boolean>)	Sobrecargado. Filtra una secuencia de valores en función de un predicado.(Definido por Enumerable).
Where<T>(Func<T, Int32, Boolean>)	Sobrecargado. Filtra una secuencia de valores en función de un predicado. El índice de cada elemento se usa en la lógica de la función de predicado.(Definido por Enumerable).
Zip<T, TSecond, TResult>(IEnumerable<TSecond>, Func<T, TSecond, TResult>)	Aplica una función especificada a los elementos correspondientes de dos secuencias, una secuencia de los resultados.(Definido por

Enumerable).

Comentarios

Stack<T> se implementa como una matriz.

Pilas y las colas son útiles cuando se necesita almacenamiento temporal de la información; es decir, cuando desee descartar un elemento después de recuperar su valor. Use Queue<T> Si necesita tener acceso a la información en el mismo orden en que se almacena en la colección. Use System.Collections.Generic.Stack<T> Si necesita tener acceso a la información en orden inverso.

Utilice la System.Collections.Concurrent.ConcurrentStack<T> y System.Collections.Concurrent.ConcurrentQueue<T> tipos cuando necesite tener acceso a la colección desde varios subprocesos simultáneamente.

Un uso común de System.Collections.Generic.Stack<T> es conservar los Estados de las variables durante las llamadas a otros procedimientos.

Se pueden realizar tres operaciones principales en una System.Collections.Generic.Stack<T> y sus elementos:

- Push Inserta un elemento en la parte superior de la Stack.
- Pop Quita un elemento de la parte superior de la Stack<T>.
- Peek Devuelve un elemento que está en la parte superior de la Stack<T> pero no lo quita de la Stack<T>.

La capacidad de un Stack<T> es el número de elementos de la Stack<T> puede contener. Cuando se agregan elementos a un Stack<T>, la capacidad aumenta automáticamente según sea necesario mediante la reasignación de la matriz interna. La capacidad puede reducirse llamando a TrimExcess.

Si Count es menor que la capacidad de la pila, Push es una operación o (1). Si es necesario aumentar para alojar el nuevo elemento, la capacidad de Push se convierte en una O (*n*) operación, donde *n* es Count. Pop es una operación o (1).

Stack<T> acepta **null** como un valor válido para la referencia de tipos y permite elementos duplicados.

Ejemplos

En el ejemplo de código siguiente se muestra varios métodos de la Stack<T> clase genérica. El ejemplo de código crea una pila de cadenas con capacidad predeterminada y utiliza el Push

método para insertar cinco cadenas en la pila. Se enumeran los elementos de la pila, que no cambia el estado de la pila. El Pop método se utiliza para extraer la primera cadena de la pila. El Peek método se utiliza para buscar el siguiente elemento en la pila y, a continuación, el Pop método se usa para extraer.

El ToArray método se usa para crear una matriz y copiar los elementos de la pila en él, a continuación, la matriz se pasa a la Stack<T> constructor que toma IEnumerable<T>, crear una copia de la pila con el orden de los elementos invertidos. Se muestran los elementos de la copia.

Dos veces el tamaño de la pila se crea una matriz y la CopyTo método se usa para copiar los elementos de matriz, empezando por el centro de la matriz. El Stack<T> constructor se utiliza de nuevo para crear una copia de la pila con el orden de los elementos invertidos; por lo tanto, los tres elementos null se encuentran al final.

El Contains método se usa para mostrar que la cadena "four" está en la primera copia de la pila, después del cual el Clear método borra la copia y el Count propiedad muestra que la pila está vacía.

C#
```
using System;
using System.Collections.Generic;

class Example
{
    public static void Main()
    {
        Stack<string> numbers = new Stack<string>();
        numbers.Push("one");
        numbers.Push("two");
        numbers.Push("three");
        numbers.Push("four");
        numbers.Push("five");

        // A stack can be enumerated without disturbing its contents.
        foreach( string number in numbers )
        {
            Console.WriteLine(number);
        }

        Console.WriteLine("\nPopping '{0}'", numbers.Pop());
        Console.WriteLine("Peek at next item to destack: {0}",
            numbers.Peek());
        Console.WriteLine("Popping '{0}'", numbers.Pop());

        // Create a copy of the stack, using the ToArray method and the
        // constructor that accepts an IEnumerable<T>.
        Stack<string> stack2 = new Stack<string>(numbers.ToArray());

        Console.WriteLine("\nContents of the first copy:");
        foreach( string number in stack2 )
        {
            Console.WriteLine(number);
        }
```

```
        // Create an array twice the size of the stack and copy the
        // elements of the stack, starting at the middle of the
        // array.
        string[] array2 = new string[numbers.Count * 2];
        numbers.CopyTo(array2, numbers.Count);

        // Create a second stack, using the constructor that accepts an
        // IEnumerable(Of T).
        Stack<string> stack3 = new Stack<string>(array2);

        Console.WriteLine("\nContents of the second copy, with duplicates and
nulls:");
        foreach( string number in stack3 )
        {
            Console.WriteLine(number);
        }

        Console.WriteLine("\nstack2.Contains(\"four\") = {0}",
            stack2.Contains("four"));

        Console.WriteLine("\nstack2.Clear()");
        stack2.Clear();
        Console.WriteLine("\nstack2.Count = {0}", stack2.Count);
    }
}

/* This code example produces the following output:

five
four
three
two
one

Popping 'five'
Peek at next item to destack: four
Popping 'four'

Contents of the first copy:
one
two
three

Contents of the second copy, with duplicates and nulls:
one
two
three




stack2.Contains("four") = False

stack2.Clear()

stack2.Count = 0
 */
```

Información de versión

Plataforma universal de Windows
Disponible desde 8
.NET Framework

Seguridad para subprocesos

Público estático (**Shared** en Visual Basic) de este tipo es seguro para subprocesos. No se garantiza que los miembros de instancias sean seguros para la ejecución de subprocesos.

Un Stack<T> puede admitir varios sistemas de lectura simultáneamente, siempre y cuando no se modifica la colección. Aun así, enumerar una colección no es intrínsecamente un procedimiento seguro para subprocesos. A fin de garantizar la seguridad de los subprocesos, se puede bloquear la colección durante toda la enumeración. Para permitir que varios subprocesos obtengan acceso de lectura y escritura a la colección, debe implementar su propia sincronización.

Ver también

Espacio de nombres System.Collections.Generic
Iteradores (C# y Visual Basic)

Volver al principio

Capítulo 3. Colas - Queue(T)

Clase Queue<T>

.NET Framework (current version)
Publicado: octubre de 2016

Representa una colección de objetos de tipo primero en entrar, primero en salir.

Espacio de nombres: System.Collections.Generic
Ensamblado: System (en System.dll)

Jerarquía de herencia

System.Object
  System.Collections.Generic.Queue<T>

Sintaxis

C#
```
[SerializableAttribute]
[ComVisibleAttribute(false)]
public class Queue<T> : IEnumerable<T>, IEnumerable, ICollection,
        IReadOnlyCollection<T>
```

Parámetros de tipo

T

Especifica el tipo de elementos en la cola.

Constructores

Nombre	Descripción
Queue<T>()	Inicializa una nueva instancia de la clase Queue<T> que está vacía y tiene la capacidad inicial predeterminada.
Queue<T>(IEnumerable<T>)	Inicializa una nueva instancia de la clase Queue<T> que contiene elementos copiados de la colección especificada y tiene una capacidad suficiente para aceptar el número de elementos copiados.
Queue<T>(Int32)	Inicializa una nueva instancia de la clase Queue<T> que está vacía y tiene la capacidad inicial especificada.

Propiedades

Nombre	Descripción
Count	Obtiene el número de elementos incluidos en Queue<T>.

Métodos

Nombre	Descripción
Clear()	Quita todos los objetos de la colección Queue<T>.
Contains(T)	Determina si un elemento se encuentra en Queue<T>.
CopyTo(T[], Int32)	Copia los elementos de Queue<T> en una Array unidimensional existente, a partir del índice especificado de la matriz.
Dequeue()	Quita y devuelve el objeto al comienzo de Queue<T>.
Enqueue(T)	Agrega un objeto al final de Queue<T>.
Equals(Object)	Determina si el objeto especificado es igual al objeto actual.(Heredado de Object).
Finalize()	Permite que un objeto intente liberar recursos y realizar otras operaciones de limpieza antes de ser reclamado por el recolector de basura. (Heredado de Object).
GetEnumerator()	Devuelve un enumerador que recorre en iteración la colección Queue<T>.
GetHashCode()	Sirve como la función hash predeterminada.(Heredado de Object).
GetType()	Obtiene el Type de la instancia actual.(Heredado de Object).
MemberwiseClone()	Crea una copia superficial del Object actual.(Heredado de Object).

Peek()	Devuelve un objeto al principio de Queue<T> sin eliminarlo.
ToArray()	Copia los elementos Queue<T> en una matriz nueva.
ToString()	Devuelve una cadena que representa al objeto actual. (Heredado de Object).
TrimExcess()	Establece la capacidad en el número real de elementos de Queue<T>, si este número supone menos del 90 por ciento de la capacidad actual.

Implementaciones de interfaz explícitas

Nombre	Descripción
IEnumerable<T>.GetEnumerator()	Devuelve un enumerador que recorre en iteración una colección.
ICollection.CopyTo(Array, Int32)	Copia los elementos de ICollection en Array, empezando por un índice determinado de Array.
IEnumerable.GetEnumerator()	Devuelve un enumerador que recorre en iteración una colección.
ICollection.IsSynchronized	Obtiene un valor que indica si el acceso a la interfaz ICollection está sincronizado (es seguro para subprocesos).
ICollection.SyncRoot	Obtiene un objeto que se puede usar para sincronizar el acceso a ICollection.

Métodos de extensión

Nombre	Descripción
Aggregate<T>(Func<T, T, T>)	Sobrecargado. Aplica una función de acumulador a una secuencia.(Definido por Enumerable).

Aggregate<T, TAccumulate>(TAccumulate, Func<TAccumulate, T, TAccumulate>)	Sobrecargado. Aplica una función de acumulador a una secuencia. El valor de inicialización especificado se utiliza como valor de inicio del acumulador.(Definido por Enumerable).
Aggregate<T, TAccumulate, TResult>(TAccumulate, Func<TAccumulate, T, TAccumulate>, Func<TAccumulate, TResult>)	Sobrecargado. Aplica una función de acumulador a una secuencia. El valor de inicialización especificado se utiliza como valor inicial del acumulador y la función especificada se utiliza para seleccionar el valor resultante.(Definido por Enumerable).
All<T>(Func<T, Boolean>)	Determina si todos los elementos de una secuencia satisfacen una condición.(Definido por Enumerable).
Any<T>()	Sobrecargado. Determina si una secuencia contiene elementos.(Definido por Enumerable).
Any<T>(Func<T, Boolean>)	Sobrecargado. Determina si un elemento de una secuencia satisface una condición.(Definido por Enumerable).
AsEnumerable<T>()	Devuelve la entrada de tipo IEnumerable<T>. (Definido por Enumerable).
AsParallel()	Sobrecargado. Habilita la paralelización de una consulta.(Definido por ParallelEnumerable).
AsParallel<T>()	Sobrecargado. Habilita la paralelización de una consulta.(Definido por ParallelEnumerable).
AsQueryable()	Sobrecargado. Convierte un IEnumerable para un IQueryable.(Definido por Queryable).
AsQueryable<T>()	Sobrecargado. Convierte un tipo genérico

	IEnumerable<T> a un tipo genérico IQueryable<T>.(Definido por Queryable).
Average<T>(Func<T, Decimal>)	Sobrecargado. Calcula el promedio de una secuencia de Decimal valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Average<T>(Func<T, Double>)	Sobrecargado. Calcula el promedio de una secuencia de Double valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Average<T>(Func<T, Int32>)	Sobrecargado. Calcula el promedio de una secuencia de Int32 valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Average<T>(Func<T, Int64>)	Sobrecargado. Calcula el promedio de una secuencia de Int64 valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Average<T>(Func<T, Nullable<Decimal>>)	Sobrecargado. Calcula el promedio de una secuencia de nullable Decimal valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Average<T>(Func<T, Nullable<Double>>)	Sobrecargado. Calcula el promedio de una secuencia de nullable Double valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).

Average<T>(Func<T, Nullable<Int32>>)	Sobrecargado. Calcula el promedio de una secuencia de nullable Int32 valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Average<T>(Func<T, Nullable<Int64>>)	Sobrecargado. Calcula el promedio de una secuencia de nullable Int64 valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Average<T>(Func<T, Nullable<Single>>)	Sobrecargado. Calcula el promedio de una secuencia de nullable Single valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Average<T>(Func<T, Single>)	Sobrecargado. Calcula el promedio de una secuencia de Single valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Cast<TResult>()	Convierte los elementos de un IEnumerable al tipo especificado.(Definido por Enumerable).
Concat<T>(IEnumerable<T>)	Concatena dos secuencias.(Definido por Enumerable).
Contains<T>(T)	Sobrecargado. Determina si una secuencia contiene un elemento especificado mediante el comparador de igualdad predeterminado. (Definido por Enumerable).
Contains<T>(T, IEqualityComparer<T>)	Sobrecargado. Determina si una secuencia contiene un elemento especificado mediante el uso de un objeto IEqualityComparer<T>.

	(Definido por Enumerable).
Count<T>()	Sobrecargado. Devuelve el número de elementos de una secuencia.(Definido por Enumerable).
Count<T>(Func<T, Boolean>)	Sobrecargado. Devuelve un número que representa cuántos elementos de la secuencia especificada satisfacen una condición.(Definido por Enumerable).
DefaultIfEmpty<T>()	Sobrecargado. Devuelve los elementos de la secuencia especificada o el valor predeterminado del parámetro de tipo en una colección singleton si la secuencia está vacía.(Definido por Enumerable).
DefaultIfEmpty<T>(T)	Sobrecargado. Devuelve los elementos de la secuencia especificada o el valor especificado en una colección singleton si la secuencia está vacía. (Definido por Enumerable).
Distinct<T>()	Sobrecargado. Devuelve diversos elementos de una secuencia utilizando el comparador de igualdad predeterminado para comparar valores. (Definido por Enumerable).
Distinct<T>(IEqualityComparer<T>)	Sobrecargado. Devuelve diversos elementos de una secuencia utilizando un IEqualityComparer<T> para comparar valores. (Definido por Enumerable).
ElementAt<T>(Int32)	Devuelve el elemento en un índice especificado en una secuencia.(Definido por Enumerable).
ElementAtOrDefault<T>(Int32)	Devuelve el elemento situado en un índice especificado de una secuencia o un valor predeterminado si el índice está fuera del

	intervalo.(Definido por Enumerable).
Except<T>(IEnumerable<T>)	Sobrecargado. Proporciona la diferencia de conjunto de dos secuencias utilizando el comparador de igualdad predeterminado para comparar valores.(Definido por Enumerable).
Except<T>(IEnumerable<T>, IEqualityComparer<T>)	Sobrecargado. Proporciona la diferencia de conjuntos de dos secuencias mediante especificado IEqualityComparer<T> para comparar valores.(Definido por Enumerable).
First<T>()	Sobrecargado. Devuelve el primer elemento de una secuencia.(Definido por Enumerable).
First<T>(Func<T, Boolean>)	Sobrecargado. Devuelve el primer elemento de una secuencia que satisface una condición especificada.(Definido por Enumerable).
FirstOrDefault<T>()	Sobrecargado. Devuelve el primer elemento de una secuencia o un valor predeterminado si la secuencia no contiene elementos.(Definido por Enumerable).
FirstOrDefault<T>(Func<T, Boolean>)	Sobrecargado. Devuelve el primer elemento de la secuencia que satisface una condición o un valor predeterminado si no se encuentra dicho elemento.(Definido por Enumerable).
GroupBy<T, TKey>(Func<T, TKey>)	Sobrecargado. Agrupa los elementos de una secuencia según una función del selector de claves especificada.(Definido por Enumerable).
GroupBy<T, TKey>(Func<T, TKey>, IEqualityComparer<TKey>)	Sobrecargado. Agrupa los elementos de una secuencia según una función del selector de claves especificada y compara las claves utilizando un comparador especificado.(Definido

	por Enumerable).
GroupBy<T, TKey, TElement>(Func<T, TKey>, Func<T, TElement>)	Sobrecargado. Agrupa los elementos de una secuencia según una función del selector de claves especificada y proyecta los elementos de cada grupo utilizando una función especificada. (Definido por Enumerable).
GroupBy<T, TKey, TElement>(Func<T, TKey>, Func<T, TElement>, IEqualityComparer<TKey>)	Sobrecargado. Agrupa los elementos de una secuencia según una función del selector de claves. Las claves se comparan utilizando un comparador y los elementos de cada grupo se proyectan utilizando una función especificada. (Definido por Enumerable).
GroupBy<T, TKey, TResult>(Func<T, TKey>, Func<TKey, IEnumerable<T>, TResult>)	Sobrecargado. Agrupa los elementos de una secuencia según una función del selector de claves especificada y crea un valor de resultado a partir de cada grupo y su clave.(Definido por Enumerable).
GroupBy<T, TKey, TResult>(Func<T, TKey>, Func<TKey, IEnumerable<T>, TResult>, IEqualityComparer<TKey>)	Sobrecargado. Agrupa los elementos de una secuencia según una función del selector de claves especificada y crea un valor de resultado a partir de cada grupo y su clave. Las claves se comparan utilizando un comparador especificado. (Definido por Enumerable).
GroupBy<T, TKey, TElement, TResult>(Func<T, TKey>, Func<T, TElement>, Func<TKey, IEnumerable<TElement>, TResult>)	Sobrecargado. Agrupa los elementos de una secuencia según una función del selector de claves especificada y crea un valor de resultado a partir de cada grupo y su clave. Los elementos de cada grupo se proyectan utilizando una función determinada.(Definido por Enumerable).
GroupBy<T, TKey, TElement, TResult>(Func<T, TKey>, Func<T, TElement>, Func<TKey, IEnumerable<TElement>, TResult>, IEqualityComparer<TKey>)	Sobrecargado. Agrupa los elementos de una secuencia según una función del selector de claves especificada y crea un valor de resultado a partir de cada grupo y su clave. Los valores de las

	claves se comparan utilizando un comparador especificado y los elementos de cada grupo se proyectan utilizando una función especificada. (Definido por Enumerable).
GroupJoin<T, TInner, TKey, TResult>(IEnumerable<TInner>, Func<T, TKey>, Func<TInner, TKey>, Func<T, IEnumerable<TInner>, TResult>)	Sobrecargado. Correlaciona los elementos de dos secuencias basándose en la igualdad de claves y agrupa los resultados. El comparador de igualdad predeterminado se usa para comparar claves. (Definido por Enumerable).
GroupJoin<T, TInner, TKey, TResult>(IEnumerable<TInner>, Func<T, TKey>, Func<TInner, TKey>, Func<T, IEnumerable<TInner>, TResult>, IEqualityComparer<TKey>)	Sobrecargado. Correlaciona los elementos de dos secuencias basándose en la igualdad de claves y agrupa los resultados. Se usa un IEqualityComparer<T> especificado para comparar claves.(Definido por Enumerable).
Intersect<T>(IEnumerable<T>)	Sobrecargado. Proporciona la intersección de conjunto de dos secuencias utilizando el comparador de igualdad predeterminado para comparar valores.(Definido por Enumerable).
Intersect<T>(IEnumerable<T>, IEqualityComparer<T>)	Sobrecargado. Proporciona la intersección de conjuntos de dos secuencias mediante especificado IEqualityComparer<T> para comparar valores.(Definido por Enumerable).
Join<T, TInner, TKey, TResult>(IEnumerable<TInner>, Func<T, TKey>, Func<TInner, TKey>, Func<T, TInner, TResult>)	Sobrecargado. Correlaciona los elementos de dos secuencias basadas en claves coincidentes. El comparador de igualdad predeterminado se usa para comparar claves.(Definido por Enumerable).
Join<T, TInner, TKey, TResult>(IEnumerable<TInner>, Func<T, TKey>, Func<TInner, TKey>, Func<T, TInner, TResult>, IEqualityComparer<TKey>)	Sobrecargado. Correlaciona los elementos de dos secuencias basadas en claves coincidentes. Se usa un IEqualityComparer<T> especificado para comparar claves.(Definido por Enumerable).
Last<T>()	Sobrecargado. Devuelve el último elemento de

	una secuencia.(Definido por Enumerable).
Last<T>(Func<T, Boolean>)	Sobrecargado. Devuelve el último elemento de una secuencia que satisface una condición especificada.(Definido por Enumerable).
LastOrDefault<T>()	Sobrecargado. Devuelve el último elemento de una secuencia o un valor predeterminado si la secuencia no contiene elementos.(Definido por Enumerable).
LastOrDefault<T>(Func<T, Boolean>)	Sobrecargado. Devuelve el último elemento de una secuencia que satisface una condición o un valor predeterminado si no se encuentra dicho elemento.(Definido por Enumerable).
LongCount<T>()	Sobrecargado. Devuelve un Int64 que representa el número total de elementos de una secuencia. (Definido por Enumerable).
LongCount<T>(Func<T, Boolean>)	Sobrecargado. Devuelve un Int64 que representa el número de elementos de una secuencia satisface una condición.(Definido por Enumerable).
Max<T>()	Sobrecargado. Devuelve el valor máximo de una secuencia genérica.(Definido por Enumerable).
Max<T>(Func<T, Decimal>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo Decimal valor.(Definido por Enumerable).
Max<T>(Func<T, Double>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo Double

	valor.(Definido por Enumerable).
Max<T>(Func<T, Int32>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo Int32 valor. (Definido por Enumerable).
Max<T>(Func<T, Int64>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo Int64 valor. (Definido por Enumerable).
Max<T>(Func<T, Nullable<Decimal>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo que acepta valores NULL Decimal valor.(Definido por Enumerable).
Max<T>(Func<T, Nullable<Double>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo que acepta valores NULL Double valor.(Definido por Enumerable).
Max<T>(Func<T, Nullable<Int32>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo que acepta valores NULL Int32 valor.(Definido por Enumerable).
Max<T>(Func<T, Nullable<Int64>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo que acepta valores NULL Int64 valor.(Definido por Enumerable).
Max<T>(Func<T, Nullable<Single>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una

	secuencia y devuelve el valor máximo que acepta valores NULL Single valor.(Definido por Enumerable).
Max<T>(Func<T, Single>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo Single valor.(Definido por Enumerable).
Max<T, TResult>(Func<T, TResult>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia genérica y devuelve el valor máximo resultante.(Definido por Enumerable).
Min<T>()	Sobrecargado. Devuelve el valor mínimo de una secuencia genérica.(Definido por Enumerable).
Min<T>(Func<T, Decimal>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo Decimal valor. (Definido por Enumerable).
Min<T>(Func<T, Double>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo Double valor. (Definido por Enumerable).
Min<T>(Func<T, Int32>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo Int32 valor. (Definido por Enumerable).
Min<T>(Func<T, Int64>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo Int64 valor. (Definido por Enumerable).

Min<T>(Func<T, Nullable<Decimal>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo que acepta valores NULL Decimal valor.(Definido por Enumerable).
Min<T>(Func<T, Nullable<Double>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo que acepta valores NULL Double valor.(Definido por Enumerable).
Min<T>(Func<T, Nullable<Int32>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo que acepta valores NULL Int32 valor.(Definido por Enumerable).
Min<T>(Func<T, Nullable<Int64>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo que acepta valores NULL Int64 valor.(Definido por Enumerable).
Min<T>(Func<T, Nullable<Single>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo que acepta valores NULL Single valor.(Definido por Enumerable).
Min<T>(Func<T, Single>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo Single valor. (Definido por Enumerable).
Min<T, TResult>(Func<T, TResult>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia genérica y devuelve el valor mínimo

	resultante.(Definido por Enumerable).
OfType<TResult>()	Filtra los elementos de un IEnumerable basado en un tipo especificado.(Definido por Enumerable).
OrderBy<T, TKey>(Func<T, TKey>)	Sobrecargado. Ordena los elementos de una secuencia en orden ascendente según una clave. (Definido por Enumerable).
OrderBy<T, TKey>(Func<T, TKey>, IComparer<TKey>)	Sobrecargado. Ordena los elementos de una secuencia en orden ascendente mediante un comparador especificado.(Definido por Enumerable).
OrderByDescending<T, TKey>(Func<T, TKey>)	Sobrecargado. Ordena los elementos de una secuencia en orden descendente según una clave. (Definido por Enumerable).
OrderByDescending<T, TKey>(Func<T, TKey>, IComparer<TKey>)	Sobrecargado. Ordena de manera descendente los elementos de una secuencia utilizando un comparador especificado.(Definido por Enumerable).
Reverse<T>()	Invierte el orden de los elementos de una secuencia.(Definido por Enumerable).
Select<T, TResult>(Func<T, TResult>)	Sobrecargado. Proyecta cada elemento de una secuencia a un nuevo formulario.(Definido por Enumerable).
Select<T, TResult>(Func<T, Int32, TResult>)	Sobrecargado. Proyecta cada elemento de una secuencia en un nuevo formulario mediante la incorporación del índice del elemento.(Definido por Enumerable).
SelectMany<T, TResult>(Func<T, IEnumerable<TResult>>)	Sobrecargado. Proyecta cada elemento de una

	secuencia en un IEnumerable<T> y reduce las secuencias resultantes en una secuencia.(Definido por Enumerable).
SelectMany<T, TResult>(Func<T, Int32, IEnumerable<TResult>>)	Sobrecargado. Proyecta cada elemento de una secuencia en un IEnumerable<T>, y reduce las secuencias resultantes en una secuencia. El índice de cada elemento de origen se utiliza en el formulario proyectado de ese elemento.(Definido por Enumerable).
SelectMany<T, TCollection, TResult>(Func<T, IEnumerable<TCollection>>, Func<T, TCollection, TResult>)	Sobrecargado. Proyecta cada elemento de una secuencia en IEnumerable<T>, reduce las secuencias resultantes en una única secuencia e invoca una función del selector de resultados en cada elemento.(Definido por Enumerable).
SelectMany<T, TCollection, TResult>(Func<T, Int32, IEnumerable<TCollection>>, Func<T, TCollection, TResult>)	Sobrecargado. Proyecta cada elemento de una secuencia en IEnumerable<T>, reduce las secuencias resultantes en una única secuencia e invoca una función del selector de resultados en cada elemento. El índice de cada elemento de origen se utiliza en el formulario proyectado intermedio de ese elemento.(Definido por Enumerable).
SequenceEqual<T>(IEnumerable<T>)	Sobrecargado. Determina si dos secuencias son iguales al comparar los elementos mediante el comparador de igualdad predeterminado para su tipo.(Definido por Enumerable).
SequenceEqual<T>(IEnumerable<T>, IEqualityComparer<T>)	Sobrecargado. Determina si dos secuencias son iguales al comparar sus elementos mediante un objeto IEqualityComparer<T>.(Definido por Enumerable).
Single<T>()	Sobrecargado. Devuelve el único elemento de una secuencia y produce una excepción si no hay exactamente un elemento en la secuencia.

	(Definido por Enumerable).
Single<T>(Func<T, Boolean>)	Sobrecargado. Devuelve el único elemento de una secuencia que cumpla una condición especificada y produce una excepción si existe más de un elemento de este tipo.(Definido por Enumerable).
SingleOrDefault<T>()	Sobrecargado. Devuelve el único elemento de una secuencia o un valor predeterminado si la secuencia está vacía; Este método produce una excepción si hay más de un elemento en la secuencia.(Definido por Enumerable).
SingleOrDefault<T>(Func<T, Boolean>)	Sobrecargado. Devuelve el único elemento de una secuencia que cumpla la condición especificada, o bien, un valor predeterminado si ese elemento no existe; este método produce una excepción si varios elementos cumplen la condición.(Definido por Enumerable).
Skip<T>(Int32)	Omite un número especificado de elementos de una secuencia y, a continuación, devuelve los elementos restantes.(Definido por Enumerable).
SkipWhile<T>(Func<T, Boolean>)	Sobrecargado. Omite los elementos de una secuencia siempre que una condición especificada sea true y, a continuación, devuelve los elementos restantes.(Definido por Enumerable).
SkipWhile<T>(Func<T, Int32, Boolean>)	Sobrecargado. Omite los elementos de una secuencia siempre que una condición especificada sea true y, a continuación, devuelve los elementos restantes. El índice del elemento se usa en la lógica de la función de predicado.(Definido por Enumerable).
Sum<T>(Func<T, Decimal>)	Sobrecargado. Calcula la suma de la secuencia de Decimal valores que se obtienen mediante la

	invocación de una función de transformación en cada elemento de la secuencia de entrada. (Definido por Enumerable).
Sum<T>(Func<T, Double>)	Sobrecargado. Calcula la suma de la secuencia de Double valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada. (Definido por Enumerable).
Sum<T>(Func<T, Int32>)	Sobrecargado. Calcula la suma de la secuencia de Int32 valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada. (Definido por Enumerable).
Sum<T>(Func<T, Int64>)	Sobrecargado. Calcula la suma de la secuencia de Int64 valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada. (Definido por Enumerable).
Sum<T>(Func<T, Nullable<Decimal>>)	Sobrecargado. Calcula la suma de la secuencia de nullable Decimal valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Sum<T>(Func<T, Nullable<Double>>)	Sobrecargado. Calcula la suma de la secuencia de nullable Double valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada. (Definido por Enumerable).
Sum<T>(Func<T, Nullable<Int32>>)	Sobrecargado. Calcula la suma de la secuencia de nullable Int32 valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.

	(Definido por Enumerable).
Sum<T>(Func<T, Nullable<Int64>>)	Sobrecargado. Calcula la suma de la secuencia de nullable Int64 valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada. (Definido por Enumerable).
Sum<T>(Func<T, Nullable<Single>>)	Sobrecargado. Calcula la suma de la secuencia de nullable Single valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada. (Definido por Enumerable).
Sum<T>(Func<T, Single>)	Sobrecargado. Calcula la suma de la secuencia de Single valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada. (Definido por Enumerable).
Take<T>(Int32)	Devuelve un número especificado de elementos contiguos desde el principio de una secuencia. (Definido por Enumerable).
TakeWhile<T>(Func<T, Boolean>)	Sobrecargado. Devuelve los elementos de una secuencia siempre que una condición especificada sea true.(Definido por Enumerable).
TakeWhile<T>(Func<T, Int32, Boolean>)	Sobrecargado. Devuelve los elementos de una secuencia siempre que una condición especificada sea true. El índice del elemento se usa en la lógica de la función de predicado.(Definido por Enumerable).
ToArray<T>()	Crea una matriz a partir de un IEnumerable<T>. (Definido por Enumerable).

ToDictionary<T, TKey>(Func<T, TKey>)	Sobrecargado. Crea un Dictionary<TKey, TValue> de un IEnumerable<T> según una función del selector de claves especificada. (Definido por Enumerable).
ToDictionary<T, TKey>(Func<T, TKey>, IEqualityComparer<TKey>)	Sobrecargado. Crea un Dictionary<TKey, TValue> de un IEnumerable<T> según un comparador de función y la clave del selector de claves especificada.(Definido por Enumerable).
ToDictionary<T, TKey, TElement>(Func<T, TKey>, Func<T, TElement>)	Sobrecargado. Crea un Dictionary<TKey, TValue> de un IEnumerable<T> según el selector de claves especificada y las funciones del selector de elementos.(Definido por Enumerable).
ToDictionary<T, TKey, TElement>(Func<T, TKey>, Func<T, TElement>, IEqualityComparer<TKey>)	Sobrecargado. Crea un Dictionary<TKey, TValue> de un IEnumerable<T> según una función del selector de claves especificada, un comparador y una función del selector de elementos.(Definido por Enumerable).
ToList<T>()	Crea un List<T> a partir de un IEnumerable<T>. (Definido por Enumerable).
ToLookup<T, TKey>(Func<T, TKey>)	Sobrecargado. Crea un Lookup<TKey, TElement> de un IEnumerable<T> según una función del selector de claves especificada. (Definido por Enumerable).
ToLookup<T, TKey>(Func<T, TKey>, IEqualityComparer<TKey>)	Sobrecargado. Crea un Lookup<TKey, TElement> de un IEnumerable<T> según un comparador de función y la clave del selector de claves especificada.(Definido por Enumerable).
ToLookup<T, TKey, TElement>(Func<T, TKey>, Func<T, TElement>)	Sobrecargado. Crea un Lookup<TKey, TElement> de un IEnumerable<T> según el selector de claves especificada y las funciones del

	selector de elementos.(Definido por Enumerable).
ToLookup<T, TKey, TElement>(Func<T, TKey>, Func<T, TElement>, IEqualityComparer<TKey>)	Sobrecargado. Crea un Lookup<TKey, TElement> de un IEnumerable<T> según una función del selector de claves especificada, un comparador y una función del selector de elementos.(Definido por Enumerable).
Union<T>(IEnumerable<T>)	Sobrecargado. Proporciona la unión de conjunto de dos secuencias utilizando el comparador de igualdad predeterminado.(Definido por Enumerable).
Union<T>(IEnumerable<T>, IEqualityComparer<T>)	Sobrecargado. Proporciona la unión de conjuntos de dos secuencias mediante un objeto IEqualityComparer<T>.(Definido por Enumerable).
Where<T>(Func<T, Boolean>)	Sobrecargado. Filtra una secuencia de valores en función de un predicado.(Definido por Enumerable).
Where<T>(Func<T, Int32, Boolean>)	Sobrecargado. Filtra una secuencia de valores en función de un predicado. El índice de cada elemento se usa en la lógica de la función de predicado.(Definido por Enumerable).
Zip<T, TSecond, TResult>(IEnumerable<TSecond>, Func<T, TSecond, TResult>)	Aplica una función especificada a los elementos correspondientes de dos secuencias, una secuencia de los resultados.(Definido por Enumerable).

Comentarios

Esta clase implementa una cola genérica como una matriz circular. Objetos almacenados en un Queue<T> se insertan en un extremo y se quitan del otro. Las colas y las pilas son útiles cuando se necesita almacenamiento temporal de la información; es decir, cuando desee descartar un elemento después de recuperar su valor. Use Queue<T> Si necesita tener acceso a la información

en el mismo orden en que se almacena en la colección. Use Stack<T> Si necesita tener acceso a la información en orden inverso. Use ConcurrentQueue<T> o ConcurrentStack<T> Si necesita tener acceso a la colección desde varios subprocesos simultáneamente.

Se pueden realizar tres operaciones principales en una Queue<T> y sus elementos:

- Enqueue Agrega un elemento al final de la Queue<T>.
- Dequeue Quita el elemento más antiguo desde el principio de la Queue<T>.
- Peek Peek devuelve el elemento más antiguo que está al principio de la Queue<T> pero no lo quita de la Queue<T>.

La capacidad de un Queue<T> es el número de elementos de la Queue<T> puede contener. Cuando se agregan elementos a un Queue<T>, la capacidad aumenta automáticamente según sea necesario mediante la reasignación de la matriz interna. La capacidad puede reducirse llamando a TrimExcess.

Queue<T> acepta **null** como un valor válido para la referencia de tipos y permite elementos duplicados.

Ejemplos

En el ejemplo de código siguiente se muestra varios métodos de la Queue<T> clase genérica. El ejemplo de código crea una cola de cadenas con capacidad predeterminada y utiliza el Enqueue método para poner en cola cinco cadenas. Se enumeran los elementos de la cola, que no cambia el estado de la cola. El Dequeue método se utiliza para la primera cadena de la cola. El Peek método se utiliza para buscar el elemento siguiente en la cola y, a continuación, el Dequeue método se usa para quitarlo de la cola.

El ToArray método se usa para crear una matriz y copiar los elementos de la cola a la, a continuación, la matriz se pasa a la Queue<T> constructor que toma IEnumerable<T>, crear una copia de la cola. Se muestran los elementos de la copia.

Dos veces el tamaño de la cola se crea una matriz y la CopyTo método se usa para copiar los elementos de matriz, empezando por el centro de la matriz. El Queue<T> constructor se utiliza de nuevo para crear una segunda copia de la cola que contiene tres elementos null al principio.

El Contains método se usa para mostrar que la cadena "four" está en la primera copia de la cola, después del cual el Clear método borra la copia y el Count propiedad muestra que la cola está vacía.

C#
```
using System;
```

```
using System.Collections.Generic;

class Example
{
    public static void Main()
    {
        Queue<string> numbers = new Queue<string>();
        numbers.Enqueue("one");
        numbers.Enqueue("two");
        numbers.Enqueue("three");
        numbers.Enqueue("four");
        numbers.Enqueue("five");

        // A queue can be enumerated without disturbing its contents.
        foreach( string number in numbers )
        {
            Console.WriteLine(number);
        }

        Console.WriteLine("\nDequeuing '{0}'", numbers.Dequeue());
        Console.WriteLine("Peek at next item to dequeue: {0}",
            numbers.Peek());
        Console.WriteLine("Dequeuing '{0}'", numbers.Dequeue());

        // Create a copy of the queue, using the ToArray method and the
        // constructor that accepts an IEnumerable<T>.
        Queue<string> queueCopy = new Queue<string>(numbers.ToArray());

        Console.WriteLine("\nContents of the first copy:");
        foreach( string number in queueCopy )
        {
            Console.WriteLine(number);
        }

        // Create an array twice the size of the queue and copy the
        // elements of the queue, starting at the middle of the
        // array.
        string[] array2 = new string[numbers.Count * 2];
        numbers.CopyTo(array2, numbers.Count);

        // Create a second queue, using the constructor that accepts an
        // IEnumerable(Of T).
        Queue<string> queueCopy2 = new Queue<string>(array2);

        Console.WriteLine("\nContents of the second copy, with duplicates and
nulls:");
        foreach( string number in queueCopy2 )
        {
            Console.WriteLine(number);
        }

        Console.WriteLine("\nqueueCopy.Contains(\"four\") = {0}",
            queueCopy.Contains("four"));

        Console.WriteLine("\nqueueCopy.Clear()");
        queueCopy.Clear();
        Console.WriteLine("\nqueueCopy.Count = {0}", queueCopy.Count);
    }
```

```
}

/* This code example produces the following output:

one
two
three
four
five

Dequeuing 'one'
Peek at next item to dequeue: two
Dequeuing 'two'

Contents of the copy:
three
four
five

Contents of the second copy, with duplicates and nulls:

three
four
five

queueCopy.Contains("four") = True

queueCopy.Clear()

queueCopy.Count = 0
 */
```

Seguridad para subprocesos

Público estático (**Shared** en Visual Basic) de este tipo es seguro para subprocesos. No se garantiza que los miembros de instancias sean seguros para la ejecución de subprocesos.

Un Queue<T> puede admitir varios sistemas de lectura simultáneamente, siempre y cuando no se modifica la colección. Aun así, enumerar una colección no es intrínsecamente un procedimiento seguro para subprocesos. A fin de garantizar la seguridad de los subprocesos, se puede bloquear la colección durante toda la enumeración. Para permitir que varios subprocesos obtengan acceso de lectura y escritura a la colección, debe implementar su propia sincronización.

Ver también

Espacio de nombres System.Collections.Generic
Volver al principio

Capítulo 4. Listas - List(T)

Clase List<T>

.NET Framework (current version)
Publicado: octubre de 2016

Representa una lista de objetos fuertemente tipados a la que se puede obtener acceso por índice. Proporciona métodos para buscar, ordenar y manipular listas.

Para examinar el código fuente de .NET Framework para este tipo, vea la Reference Source.

Espacio de nombres: System.Collections.Generic
Ensamblado: mscorlib (en mscorlib.dll)

Jerarquía de herencia

System.Object
  System.Collections.Generic.List<T>

System.Data.Services.ExpandSegmentCollection
System.Workflow.Activities.OperationParameterInfoCollection
System.Workflow.Activities.WorkflowRoleCollection
System.Workflow.ComponentModel.ActivityCollection
System.Workflow.ComponentModel.Design.ActivityDesignerGlyphCollection
System.Workflow.Runtime.Tracking.ActivityTrackingLocationCollection
System.Workflow.Runtime.Tracking.ActivityTrackPointCollection
System.Workflow.Runtime.Tracking.ExtractCollection
System.Workflow.Runtime.Tracking.TrackingAnnotationCollection
System.Workflow.Runtime.Tracking.TrackingConditionCollection
System.Workflow.Runtime.Tracking.UserTrackingLocationCollection
System.Workflow.Runtime.Tracking.UserTrackPointCollection
System.Workflow.Runtime.Tracking.WorkflowTrackPointCollection

Sintaxis

C#

```
[SerializableAttribute]
public class List<T> : IList<T>, ICollection<T>, IEnumerable<T>,
        IEnumerable, IList, ICollection, IReadOnlyList<T>,
IReadOnlyCollection<T>
```

Parámetros de tipo

T

Tipo de elementos en la lista.

Constructores

Nombre	Descripción
List<T>()	Inicializa una nueva instancia de la clase List<T> que está vacía y tiene la capacidad inicial predeterminada.
List<T>(IEnumerable<T>)	Inicializa una nueva instancia de la clase List<T> que contiene elementos copiados de la colección especificada y tiene una capacidad suficiente para aceptar el número de elementos copiados.
List<T>(Int32)	Inicializa una nueva instancia de la clase List<T> que está vacía y tiene la capacidad inicial especificada.

Propiedades

Nombre	Descripción
Capacity	Obtiene o establece el número total de elementos que puede contener la estructura de datos interna sin cambiar el tamaño.
Count	Obtiene el número de elementos incluidos en List<T>.
Item[Int32]	Obtiene o establece el elemento en el índice especificado.

Métodos

Nombre	Descripción
Add(T)	Agrega un objeto al final de List<T>.
AddRange(IEnumerable<T>)	Agrega los elementos de la colección especificada al final de List<T>.
AsReadOnly()	Devuelve un contenedor de ReadOnlyCollection<T> de solo lectura para la colección actual.
BinarySearch(T)	Busca la List<T> completa ordenada para un elemento usando el comparador predeterminado y devuelve el índice de base cero del elemento.
BinarySearch(T, IComparer<T>)	Busca la List<T> completa ordenada para un elemento usando el comparador especificado y devuelve el índice de base cero del elemento.
BinarySearch(Int32, Int32, T, IComparer<T>)	Busca un elemento en un intervalo de elementos del objeto List<T> ordenado usando el comparador especificado y devuelve el índice de base cero del elemento.

Clear()	Quita todos los elementos de List<T>.
Contains(T)	Determina si un elemento se encuentra en List<T>.
ConvertAll<TOutput>(Converter<T, TOutput>)	Convierte en otro tipo los elementos incluidos en la List<T> actual y devuelve una lista que contiene los elementos convertidos.
CopyTo(T[])	Copia toda la List<T> en una matriz unidimensional compatible, empezando en el principio de la matriz de destino.
CopyTo(T[], Int32)	Copia la totalidad de List<T> en una matriz unidimensional compatible, empezando por el índice especificado de la matriz de destino.
CopyTo(Int32, T[], Int32, Int32)	Copia un intervalo de elementos de List<T> en una matriz unidimensional compatible, empezando en el índice especificado de la matriz de destino.
Equals(Object)	Determina si el objeto especificado es igual al objeto actual.(Heredado de Object).
Exists(Predicate<T>)	Determina si List<T> contiene elementos que cumplen las condiciones definidas por el predicado especificado.
Finalize()	Permite que un objeto intente liberar recursos y realizar otras operaciones de limpieza antes de ser reclamado por el recolector de basura. (Heredado de Object).
Find(Predicate<T>)	Busca un elemento que coincida con las condiciones definidas por el predicado especificado y devuelve la primera aparición en toda la matriz List<T>.
FindAll(Predicate<T>)	Recupera todos los elementos que coinciden con las

	condiciones definidas por el predicado especificado.
FindIndex(Int32, Int32, Predicate<T>)	Busca un elemento que coincida con las condiciones definidas por el predicado especificado y devuelve el índice de base cero de la primera aparición en el intervalo de elementos de la matriz List<T> que comienza en el índice especificado y contiene el número especificado de elementos.
FindIndex(Int32, Predicate<T>)	Busca un elemento que coincida con las condiciones definidas por el predicado especificado y devuelve el índice de base cero de la primera aparición en el intervalo de elementos de la matriz List<T> que va desde el índice especificado hasta el último elemento.
FindIndex(Predicate<T>)	Busca un elemento que coincida con las condiciones definidas por el predicado especificado y devuelve el índice de base cero de la primera aparición en toda la matriz List<T>.
FindLast(Predicate<T>)	Busca un elemento que coincida con las condiciones definidas por el predicado especificado y devuelve la última aparición en toda la matriz List<T>.
FindLastIndex(Int32, Int32, Predicate<T>)	Busca un elemento que coincida con las condiciones definidas por el predicado especificado y devuelve el índice de base cero de la última aparición en el intervalo de elementos de la matriz List<T> que contiene el número especificado de elementos y termina en el índice especificado.
FindLastIndex(Int32, Predicate<T>)	Busca un elemento que coincida con las condiciones definidas por el predicado especificado y devuelve el índice de base cero de la última aparición en el intervalo de elementos de la matriz List<T> que va desde el primer elemento hasta el índice especificado.
FindLastIndex(Predicate<T>)	Busca un elemento que coincida con las condiciones

	definidas por el predicado especificado y devuelve el índice de base cero de la última aparición en toda la matriz List<T>.
ForEach(Action<T>)	Realiza la acción especificada en cada elemento de List<T>.
GetEnumerator()	Devuelve un enumerador que recorre en iteración la colección List<T>.
GetHashCode()	Sirve como la función hash predeterminada.(Heredado de Object).
GetRange(Int32, Int32)	Crea una copia superficial de un intervalo de elementos en la List<T> de origen.
GetType()	Obtiene el Type de la instancia actual.(Heredado de Object).
IndexOf(T)	Busca el objeto especificado y devuelve el índice de base cero de la primera aparición en todo el objeto List<T>.
IndexOf(T, Int32)	Busca el objeto especificado y devuelve el índice de base cero de la primera aparición dentro del intervalo de elementos de List<T> que abarca desde el índice especificado hasta el último elemento.
IndexOf(T, Int32, Int32)	Busca el objeto especificado y devuelve el índice de base cero de la primera aparición dentro del intervalo de elementos de List<T> que comienza en el índice especificado y contiene el número especificado de elementos.
Insert(Int32, T)	Inserta un elemento en List<T>, en el índice especificado.

InsertRange(Int32, IEnumerable<T>)	Inserta los elementos de una colección en List<T> en el índice especificado.
LastIndexOf(T)	Busca el objeto especificado y devuelve el índice de base cero de la última aparición en toda la List<T>.
LastIndexOf(T, Int32)	Busca el objeto especificado y devuelve el índice de base cero de la última aparición dentro del intervalo de elementos de List<T> que abarca desde el primer elemento hasta el último índice especificado.
LastIndexOf(T, Int32, Int32)	Busca el objeto especificado y devuelve el índice de base cero de la última aparición dentro del intervalo de elementos de List<T> que contiene el número de elementos especificado y termina en el índice determinado.
MemberwiseClone()	Crea una copia superficial del Object actual.(Heredado de Object).
Remove(T)	Quita la primera aparición de un objeto específico de la interfaz List<T>.
RemoveAll(Predicate<T>)	Quita todos los elementos que cumplen las condiciones definidas por el predicado especificado.
RemoveAt(Int32)	Quita el elemento situado en el índice especificado de List<T>.
RemoveRange(Int32, Int32)	Quita todos los elementos de List<T>.
Reverse()	Invierte el orden de los elementos en la List<T> completa.
Reverse(Int32, Int32)	Invierte el orden de los elementos en el intervalo

	especificado.
Sort()	Ordena los elementos de toda la List<T> utilizando el comparador predeterminado.
Sort(Comparison<T>)	Ordena los elementos de toda la List<T> utilizando el System.Comparison<T> especificado.
Sort(IComparer<T>)	Ordena los elementos en la List<T> completa usando el comparador especificado.
Sort(Int32, Int32, IComparer<T>)	Ordena los elementos en un intervalo de elementos de la matriz List<T> usando el comparador especificado.
ToArray()	Copia los elementos de List<T> en una nueva matriz.
ToString()	Devuelve una cadena que representa al objeto actual. (Heredado de Object).
TrimExcess()	Establece la capacidad en el número real de elementos que hay en List<T>, si dicho número es inferior a un valor umbral.
TrueForAll(Predicate<T>)	Determina si cada elemento de List<T> cumple las condiciones que define el predicado especificado.

Implementaciones de interfaz explícitas

Nombre	Descripción
IEnumerable<T>.GetEnumerator()	Devuelve un enumerador que recorre en iteración una colección.
ICollection.CopyTo(Array, Int32)	Copia los elementos de ICollection en Array, empezando por

	un índice determinado de Array.
IEnumerable.GetEnumerator()	Devuelve un enumerador que recorre en iteración una colección.
IList.Add(Object)	Agrega un elemento a IList.
IList.Contains(Object)	Determina si IList contiene un valor específico.
IList.IndexOf(Object)	Determina el índice de un elemento específico de IList.
IList.Insert(Int32, Object)	Inserta un elemento en la interfaz IList, en el índice especificado.
IList.Remove(Object)	Quita la primera aparición de un objeto específico de la interfaz IList.
ICollection<T>.IsReadOnly	Obtiene un valor que indica si ICollection<T> es de solo lectura.
ICollection.IsSynchronized	Obtiene un valor que indica si el acceso a la interfaz ICollection está sincronizado (es seguro para subprocesos).
ICollection.SyncRoot	Obtiene un objeto que se puede usar para sincronizar el acceso a ICollection.
IList.IsFixedSize	Obtiene un valor que indica si la interfaz IList tiene un tamaño fijo.
IList.IsReadOnly	Obtiene un valor que indica si IList es de solo lectura.
IList.Item[Int32]	Obtiene o establece el elemento en el índice especificado.

Métodos de extensión

Nombre	Descripción
Aggregate<T>(Func<T, T, T>)	Sobrecargado. Aplica una función de acumulador a una secuencia.(Definido por Enumerable).
Aggregate<T, TAccumulate>(TAccumulate, Func<TAccumulate, T, TAccumulate>)	Sobrecargado. Aplica una función de acumulador a una secuencia. El valor de inicialización especificado se utiliza como valor de inicio del acumulador.(Definido por Enumerable).
Aggregate<T, TAccumulate, TResult>(TAccumulate, Func<TAccumulate, T, TAccumulate>, Func<TAccumulate, TResult>)	Sobrecargado. Aplica una función de acumulador a una secuencia. El valor de inicialización especificado se utiliza como valor inicial del acumulador y la función especificada se utiliza para seleccionar el valor resultante.(Definido por Enumerable).
All<T>(Func<T, Boolean>)	Determina si todos los elementos de una secuencia satisfacen una condición.(Definido por Enumerable).
Any<T>()	Sobrecargado. Determina si una secuencia contiene elementos.(Definido por Enumerable).
Any<T>(Func<T, Boolean>)	Sobrecargado. Determina si un elemento de una secuencia satisface una condición.(Definido por Enumerable).
AsEnumerable<T>()	Devuelve la entrada de tipo IEnumerable<T>. (Definido por Enumerable).
AsParallel()	Sobrecargado. Habilita la paralelización de una

	consulta.(Definido por ParallelEnumerable).
AsParallel<T>()	Sobrecargado. Habilita la paralelización de una consulta.(Definido por ParallelEnumerable).
AsQueryable()	Sobrecargado. Convierte un IEnumerable para un IQueryable.(Definido por Queryable).
AsQueryable<T>()	Sobrecargado. Convierte un tipo genérico IEnumerable<T> a un tipo genérico IQueryable<T>.(Definido por Queryable).
Average<T>(Func<T, Decimal>)	Sobrecargado. Calcula el promedio de una secuencia de Decimal valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Average<T>(Func<T, Double>)	Sobrecargado. Calcula el promedio de una secuencia de Double valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Average<T>(Func<T, Int32>)	Sobrecargado. Calcula el promedio de una secuencia de Int32 valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Average<T>(Func<T, Int64>)	Sobrecargado. Calcula el promedio de una secuencia de Int64 valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Average<T>(Func<T, Nullable<Decimal>>)	Sobrecargado. Calcula el promedio de una

	secuencia de nullable Decimal valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Average<T>(Func<T, Nullable<Double>>)	Sobrecargado. Calcula el promedio de una secuencia de nullable Double valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Average<T>(Func<T, Nullable<Int32>>)	Sobrecargado. Calcula el promedio de una secuencia de nullable Int32 valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Average<T>(Func<T, Nullable<Int64>>)	Sobrecargado. Calcula el promedio de una secuencia de nullable Int64 valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Average<T>(Func<T, Nullable<Single>>)	Sobrecargado. Calcula el promedio de una secuencia de nullable Single valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Average<T>(Func<T, Single>)	Sobrecargado. Calcula el promedio de una secuencia de Single valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).
Cast<TResult>()	Convierte los elementos de un IEnumerable al tipo especificado.(Definido por Enumerable).

Concat<T>(IEnumerable<T>)	Concatena dos secuencias.(Definido por Enumerable).
Contains<T>(T)	Sobrecargado. Determina si una secuencia contiene un elemento especificado mediante el comparador de igualdad predeterminado. (Definido por Enumerable).
Contains<T>(T, IEqualityComparer<T>)	Sobrecargado. Determina si una secuencia contiene un elemento especificado mediante el uso de un objeto IEqualityComparer<T>. (Definido por Enumerable).
Count<T>()	Sobrecargado. Devuelve el número de elementos de una secuencia.(Definido por Enumerable).
Count<T>(Func<T, Boolean>)	Sobrecargado. Devuelve un número que representa cuántos elementos de la secuencia especificada satisfacen una condición.(Definido por Enumerable).
DefaultIfEmpty<T>()	Sobrecargado. Devuelve los elementos de la secuencia especificada o el valor predeterminado del parámetro de tipo en una colección singleton si la secuencia está vacía.(Definido por Enumerable).
DefaultIfEmpty<T>(T)	Sobrecargado. Devuelve los elementos de la secuencia especificada o el valor especificado en una colección singleton si la secuencia está vacía. (Definido por Enumerable).
Distinct<T>()	Sobrecargado. Devuelve diversos elementos de una secuencia utilizando el comparador de igualdad predeterminado para comparar valores. (Definido por Enumerable).

Distinct<T>(IEqualityComparer<T>)	Sobrecargado. Devuelve diversos elementos de una secuencia utilizando un IEqualityComparer<T> para comparar valores. (Definido por Enumerable).
ElementAt<T>(Int32)	Devuelve el elemento en un índice especificado en una secuencia.(Definido por Enumerable).
ElementAtOrDefault<T>(Int32)	Devuelve el elemento situado en un índice especificado de una secuencia o un valor predeterminado si el índice está fuera del intervalo.(Definido por Enumerable).
Except<T>(IEnumerable<T>)	Sobrecargado. Proporciona la diferencia de conjunto de dos secuencias utilizando el comparador de igualdad predeterminado para comparar valores.(Definido por Enumerable).
Except<T>(IEnumerable<T>, IEqualityComparer<T>)	Sobrecargado. Proporciona la diferencia de conjuntos de dos secuencias mediante especificado IEqualityComparer<T> para comparar valores.(Definido por Enumerable).
First<T>()	Sobrecargado. Devuelve el primer elemento de una secuencia.(Definido por Enumerable).
First<T>(Func<T, Boolean>)	Sobrecargado. Devuelve el primer elemento de una secuencia que satisface una condición especificada.(Definido por Enumerable).
FirstOrDefault<T>()	Sobrecargado. Devuelve el primer elemento de una secuencia o un valor predeterminado si la secuencia no contiene elementos.(Definido por Enumerable).
FirstOrDefault<T>(Func<T, Boolean>)	Sobrecargado. Devuelve el primer elemento de la secuencia que satisface una condición o un valor

	predeterminado si no se encuentra dicho elemento.(Definido por Enumerable).
GroupBy<T, TKey>(Func<T, TKey>)	Sobrecargado. Agrupa los elementos de una secuencia según una función del selector de claves especificada.(Definido por Enumerable).
GroupBy<T, TKey>(Func<T, TKey>, IEqualityComparer<TKey>)	Sobrecargado. Agrupa los elementos de una secuencia según una función del selector de claves especificada y compara las claves utilizando un comparador especificado.(Definido por Enumerable).
GroupBy<T, TKey, TElement>(Func<T, TKey>, Func<T, TElement>)	Sobrecargado. Agrupa los elementos de una secuencia según una función del selector de claves especificada y proyecta los elementos de cada grupo utilizando una función especificada. (Definido por Enumerable).
GroupBy<T, TKey, TElement>(Func<T, TKey>, Func<T, TElement>, IEqualityComparer<TKey>)	Sobrecargado. Agrupa los elementos de una secuencia según una función del selector de claves. Las claves se comparan utilizando un comparador y los elementos de cada grupo se proyectan utilizando una función especificada. (Definido por Enumerable).
GroupBy<T, TKey, TResult>(Func<T, TKey>, Func<TKey, IEnumerable<T>, TResult>)	Sobrecargado. Agrupa los elementos de una secuencia según una función del selector de claves especificada y crea un valor de resultado a partir de cada grupo y su clave.(Definido por Enumerable).
GroupBy<T, TKey, TResult>(Func<T, TKey>, Func<TKey, IEnumerable<T>, TResult>, IEqualityComparer<TKey>)	Sobrecargado. Agrupa los elementos de una secuencia según una función del selector de claves especificada y crea un valor de resultado a partir de cada grupo y su clave. Las claves se comparan utilizando un comparador especificado. (Definido por Enumerable).

GroupBy<T, TKey, TElement, TResult>(Func<T, TKey>, Func<T, TElement>, Func<TKey, IEnumerable<TElement>, TResult>)	Sobrecargado. Agrupa los elementos de una secuencia según una función del selector de claves especificada y crea un valor de resultado a partir de cada grupo y su clave. Los elementos de cada grupo se proyectan utilizando una función determinada.(Definido por Enumerable).
GroupBy<T, TKey, TElement, TResult>(Func<T, TKey>, Func<T, TElement>, Func<TKey, IEnumerable<TElement>, TResult>, IEqualityComparer<TKey>)	Sobrecargado. Agrupa los elementos de una secuencia según una función del selector de claves especificada y crea un valor de resultado a partir de cada grupo y su clave. Los valores de las claves se comparan utilizando un comparador especificado y los elementos de cada grupo se proyectan utilizando una función especificada. (Definido por Enumerable).
GroupJoin<T, TInner, TKey, TResult>(IEnumerable<TInner>, Func<T, TKey>, Func<TInner, TKey>, Func<T, IEnumerable<TInner>, TResult>)	Sobrecargado. Correlaciona los elementos de dos secuencias basándose en la igualdad de claves y agrupa los resultados. El comparador de igualdad predeterminado se usa para comparar claves. (Definido por Enumerable).
GroupJoin<T, TInner, TKey, TResult>(IEnumerable<TInner>, Func<T, TKey>, Func<TInner, TKey>, Func<T, IEnumerable<TInner>, TResult>, IEqualityComparer<TKey>)	Sobrecargado. Correlaciona los elementos de dos secuencias basándose en la igualdad de claves y agrupa los resultados. Se usa un IEqualityComparer<T> especificado para comparar claves.(Definido por Enumerable).
Intersect<T>(IEnumerable<T>)	Sobrecargado. Proporciona la intersección de conjunto de dos secuencias utilizando el comparador de igualdad predeterminado para comparar valores.(Definido por Enumerable).
Intersect<T>(IEnumerable<T>, IEqualityComparer<T>)	Sobrecargado. Proporciona la intersección de conjuntos de dos secuencias mediante especificado IEqualityComparer<T> para comparar valores.(Definido por Enumerable).
Join<T, TInner, TKey, TResult>(IEnumerable<TInner>, Func<T,	Sobrecargado. Correlaciona los elementos de dos

TKey>, Func<TInner, TKey>, Func<T, TInner, TResult>)	secuencias basadas en claves coincidentes. El comparador de igualdad predeterminado se usa para comparar claves.(Definido por Enumerable).
Join<T, TInner, TKey, TResult>(IEnumerable<TInner>, Func<T, TKey>, Func<TInner, TKey>, Func<T, TInner, TResult>, IEqualityComparer<TKey>)	Sobrecargado. Correlaciona los elementos de dos secuencias basadas en claves coincidentes. Se usa un IEqualityComparer<T> especificado para comparar claves.(Definido por Enumerable).
Last<T>()	Sobrecargado. Devuelve el último elemento de una secuencia.(Definido por Enumerable).
Last<T>(Func<T, Boolean>)	Sobrecargado. Devuelve el último elemento de una secuencia que satisface una condición especificada.(Definido por Enumerable).
LastOrDefault<T>()	Sobrecargado. Devuelve el último elemento de una secuencia o un valor predeterminado si la secuencia no contiene elementos.(Definido por Enumerable).
LastOrDefault<T>(Func<T, Boolean>)	Sobrecargado. Devuelve el último elemento de una secuencia que satisface una condición o un valor predeterminado si no se encuentra dicho elemento.(Definido por Enumerable).
LongCount<T>()	Sobrecargado. Devuelve un Int64 que representa el número total de elementos de una secuencia. (Definido por Enumerable).
LongCount<T>(Func<T, Boolean>)	Sobrecargado. Devuelve un Int64 que representa el número de elementos de una secuencia satisface una condición.(Definido por Enumerable).
Max<T>()	Sobrecargado. Devuelve el valor máximo de una

	secuencia genérica.(Definido por Enumerable).
Max<T>(Func<T, Decimal>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo Decimal valor.(Definido por Enumerable).
Max<T>(Func<T, Double>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo Double valor.(Definido por Enumerable).
Max<T>(Func<T, Int32>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo Int32 valor. (Definido por Enumerable).
Max<T>(Func<T, Int64>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo Int64 valor. (Definido por Enumerable).
Max<T>(Func<T, Nullable<Decimal>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo que acepta valores NULL Decimal valor.(Definido por Enumerable).
Max<T>(Func<T, Nullable<Double>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo que acepta valores NULL Double valor.(Definido por Enumerable).
Max<T>(Func<T, Nullable<Int32>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo que acepta valores NULL Int32 valor.(Definido por

	Enumerable).
Max<T>(Func<T, Nullable<Int64>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo que acepta valores NULL Int64 valor.(Definido por Enumerable).
Max<T>(Func<T, Nullable<Single>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo que acepta valores NULL Single valor.(Definido por Enumerable).
Max<T>(Func<T, Single>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el valor máximo Single valor.(Definido por Enumerable).
Max<T, TResult>(Func<T, TResult>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia genérica y devuelve el valor máximo resultante.(Definido por Enumerable).
Min<T>()	Sobrecargado. Devuelve el valor mínimo de una secuencia genérica.(Definido por Enumerable).
Min<T>(Func<T, Decimal>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo Decimal valor. (Definido por Enumerable).
Min<T>(Func<T, Double>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo Double valor. (Definido por Enumerable).

Min<T>(Func<T, Int32>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo Int32 valor. (Definido por Enumerable).
Min<T>(Func<T, Int64>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo Int64 valor. (Definido por Enumerable).
Min<T>(Func<T, Nullable<Decimal>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo que acepta valores NULL Decimal valor.(Definido por Enumerable).
Min<T>(Func<T, Nullable<Double>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo que acepta valores NULL Double valor.(Definido por Enumerable).
Min<T>(Func<T, Nullable<Int32>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo que acepta valores NULL Int32 valor.(Definido por Enumerable).
Min<T>(Func<T, Nullable<Int64>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo que acepta valores NULL Int64 valor.(Definido por Enumerable).
Min<T>(Func<T, Nullable<Single>>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo que acepta valores NULL Single valor.(Definido por

	Enumerable).
Min<T>(Func<T, Single>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia y devuelve el mínimo Single valor. (Definido por Enumerable).
Min<T, TResult>(Func<T, TResult>)	Sobrecargado. Invoca una función de transformación en cada elemento de una secuencia genérica y devuelve el valor mínimo resultante.(Definido por Enumerable).
OfType<TResult>()	Filtra los elementos de un IEnumerable basado en un tipo especificado.(Definido por Enumerable).
OrderBy<T, TKey>(Func<T, TKey>)	Sobrecargado. Ordena los elementos de una secuencia en orden ascendente según una clave. (Definido por Enumerable).
OrderBy<T, TKey>(Func<T, TKey>, IComparer<TKey>)	Sobrecargado. Ordena los elementos de una secuencia en orden ascendente mediante un comparador especificado.(Definido por Enumerable).
OrderByDescending<T, TKey>(Func<T, TKey>)	Sobrecargado. Ordena los elementos de una secuencia en orden descendente según una clave. (Definido por Enumerable).
OrderByDescending<T, TKey>(Func<T, TKey>, IComparer<TKey>)	Sobrecargado. Ordena de manera descendente los elementos de una secuencia utilizando un comparador especificado.(Definido por Enumerable).
Reverse<T>()	Invierte el orden de los elementos de una secuencia.(Definido por Enumerable).

Select<T, TResult>(Func<T, TResult>)	Sobrecargado. Proyecta cada elemento de una secuencia a un nuevo formulario.(Definido por Enumerable).
Select<T, TResult>(Func<T, Int32, TResult>)	Sobrecargado. Proyecta cada elemento de una secuencia en un nuevo formulario mediante la incorporación del índice del elemento.(Definido por Enumerable).
SelectMany<T, TResult>(Func<T, IEnumerable<TResult>>)	Sobrecargado. Proyecta cada elemento de una secuencia en un IEnumerable<T> y reduce las secuencias resultantes en una secuencia.(Definido por Enumerable).
SelectMany<T, TResult>(Func<T, Int32, IEnumerable<TResult>>)	Sobrecargado. Proyecta cada elemento de una secuencia en un IEnumerable<T>, y reduce las secuencias resultantes en una secuencia. El índice de cada elemento de origen se utiliza en el formulario proyectado de ese elemento.(Definido por Enumerable).
SelectMany<T, TCollection, TResult>(Func<T, IEnumerable<TCollection>>, Func<T, TCollection, TResult>)	Sobrecargado. Proyecta cada elemento de una secuencia en IEnumerable<T>, reduce las secuencias resultantes en una única secuencia e invoca una función del selector de resultados en cada elemento.(Definido por Enumerable).
SelectMany<T, TCollection, TResult>(Func<T, Int32, IEnumerable<TCollection>>, Func<T, TCollection, TResult>)	Sobrecargado. Proyecta cada elemento de una secuencia en IEnumerable<T>, reduce las secuencias resultantes en una única secuencia e invoca una función del selector de resultados en cada elemento. El índice de cada elemento de origen se utiliza en el formulario proyectado intermedio de ese elemento.(Definido por Enumerable).
SequenceEqual<T>(IEnumerable<T>)	Sobrecargado. Determina si dos secuencias son iguales al comparar los elementos mediante el comparador de igualdad predeterminado para su

	tipo.(Definido por Enumerable).
SequenceEqual<T>(IEnumerable<T>, IEqualityComparer<T>)	Sobrecargado. Determina si dos secuencias son iguales al comparar sus elementos mediante un objeto IEqualityComparer<T>.(Definido por Enumerable).
Single<T>()	Sobrecargado. Devuelve el único elemento de una secuencia y produce una excepción si no hay exactamente un elemento en la secuencia. (Definido por Enumerable).
Single<T>(Func<T, Boolean>)	Sobrecargado. Devuelve el único elemento de una secuencia que cumpla una condición especificada y produce una excepción si existe más de un elemento de este tipo.(Definido por Enumerable).
SingleOrDefault<T>()	Sobrecargado. Devuelve el único elemento de una secuencia o un valor predeterminado si la secuencia está vacía; Este método produce una excepción si hay más de un elemento en la secuencia.(Definido por Enumerable).
SingleOrDefault<T>(Func<T, Boolean>)	Sobrecargado. Devuelve el único elemento de una secuencia que cumpla la condición especificada, o bien, un valor predeterminado si ese elemento no existe; este método produce una excepción si varios elementos cumplen la condición.(Definido por Enumerable).
Skip<T>(Int32)	Omite un número especificado de elementos de una secuencia y, a continuación, devuelve los elementos restantes.(Definido por Enumerable).
SkipWhile<T>(Func<T, Boolean>)	Sobrecargado. Omite los elementos de una secuencia siempre que una condición especificada sea true y, a continuación, devuelve los elementos

	restantes.(Definido por Enumerable).
SkipWhile<T>(Func<T, Int32, Boolean>)	Sobrecargado. Omite los elementos de una secuencia siempre que una condición especificada sea true y, a continuación, devuelve los elementos restantes. El índice del elemento se usa en la lógica de la función de predicado.(Definido por Enumerable).
Sum<T>(Func<T, Decimal>)	Sobrecargado. Calcula la suma de la secuencia de Decimal valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada. (Definido por Enumerable).
Sum<T>(Func<T, Double>)	Sobrecargado. Calcula la suma de la secuencia de Double valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada. (Definido por Enumerable).
Sum<T>(Func<T, Int32>)	Sobrecargado. Calcula la suma de la secuencia de Int32 valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada. (Definido por Enumerable).
Sum<T>(Func<T, Int64>)	Sobrecargado. Calcula la suma de la secuencia de Int64 valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada. (Definido por Enumerable).
Sum<T>(Func<T, Nullable<Decimal>>)	Sobrecargado. Calcula la suma de la secuencia de nullable Decimal valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada.(Definido por Enumerable).

Sum<T>(Func<T, Nullable<Double>>)	Sobrecargado. Calcula la suma de la secuencia de nullable Double valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada. (Definido por Enumerable).
Sum<T>(Func<T, Nullable<Int32>>)	Sobrecargado. Calcula la suma de la secuencia de nullable Int32 valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada. (Definido por Enumerable).
Sum<T>(Func<T, Nullable<Int64>>)	Sobrecargado. Calcula la suma de la secuencia de nullable Int64 valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada. (Definido por Enumerable).
Sum<T>(Func<T, Nullable<Single>>)	Sobrecargado. Calcula la suma de la secuencia de nullable Single valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada. (Definido por Enumerable).
Sum<T>(Func<T, Single>)	Sobrecargado. Calcula la suma de la secuencia de Single valores que se obtienen mediante la invocación de una función de transformación en cada elemento de la secuencia de entrada. (Definido por Enumerable).
Take<T>(Int32)	Devuelve un número especificado de elementos contiguos desde el principio de una secuencia. (Definido por Enumerable).
TakeWhile<T>(Func<T, Boolean>)	Sobrecargado. Devuelve los elementos de una secuencia siempre que una condición especificada sea true.(Definido por Enumerable).

TakeWhile<T>(Func<T, Int32, Boolean>)	Sobrecargado. Devuelve los elementos de una secuencia siempre que una condición especificada sea true. El índice del elemento se usa en la lógica de la función de predicado.(Definido por Enumerable).
ToArray<T>()	Crea una matriz a partir de un IEnumerable<T>. (Definido por Enumerable).
ToDictionary<T, TKey>(Func<T, TKey>)	Sobrecargado. Crea un Dictionary<TKey, TValue> de un IEnumerable<T> según una función del selector de claves especificada. (Definido por Enumerable).
ToDictionary<T, TKey>(Func<T, TKey>, IEqualityComparer<TKey>)	Sobrecargado. Crea un Dictionary<TKey, TValue> de un IEnumerable<T> según un comparador de función y la clave del selector de claves especificada.(Definido por Enumerable).
ToDictionary<T, TKey, TElement>(Func<T, TKey>, Func<T, TElement>)	Sobrecargado. Crea un Dictionary<TKey, TValue> de un IEnumerable<T> según el selector de claves especificada y las funciones del selector de elementos.(Definido por Enumerable).
ToDictionary<T, TKey, TElement>(Func<T, TKey>, Func<T, TElement>, IEqualityComparer<TKey>)	Sobrecargado. Crea un Dictionary<TKey, TValue> de un IEnumerable<T> según una función del selector de claves especificada, un comparador y una función del selector de elementos.(Definido por Enumerable).
ToList<T>()	Crea un List<T> a partir de un IEnumerable<T>. (Definido por Enumerable).
ToLookup<T, TKey>(Func<T, TKey>)	Sobrecargado. Crea un Lookup<TKey, TElement> de un IEnumerable<T> según una función del selector de claves especificada. (Definido por Enumerable).

ToLookup<T, TKey>(Func<T, TKey>, IEqualityComparer<TKey>)	Sobrecargado. Crea un Lookup<TKey, TElement> de un IEnumerable<T> según un comparador de función y la clave del selector de claves especificada.(Definido por Enumerable).
ToLookup<T, TKey, TElement>(Func<T, TKey>, Func<T, TElement>)	Sobrecargado. Crea un Lookup<TKey, TElement> de un IEnumerable<T> según el selector de claves especificada y las funciones del selector de elementos.(Definido por Enumerable).
ToLookup<T, TKey, TElement>(Func<T, TKey>, Func<T, TElement>, IEqualityComparer<TKey>)	Sobrecargado. Crea un Lookup<TKey, TElement> de un IEnumerable<T> según una función del selector de claves especificada, un comparador y una función del selector de elementos.(Definido por Enumerable).
Union<T>(IEnumerable<T>)	Sobrecargado. Proporciona la unión de conjunto de dos secuencias utilizando el comparador de igualdad predeterminado.(Definido por Enumerable).
Union<T>(IEnumerable<T>, IEqualityComparer<T>)	Sobrecargado. Proporciona la unión de conjuntos de dos secuencias mediante un objeto IEqualityComparer<T>.(Definido por Enumerable).
Where<T>(Func<T, Boolean>)	Sobrecargado. Filtra una secuencia de valores en función de un predicado.(Definido por Enumerable).
Where<T>(Func<T, Int32, Boolean>)	Sobrecargado. Filtra una secuencia de valores en función de un predicado. El índice de cada elemento se usa en la lógica de la función de predicado.(Definido por Enumerable).
Zip<T, TSecond, TResult>(IEnumerable<TSecond>, Func<T, TSecond, TResult>)	Aplica una función especificada a los elementos correspondientes de dos secuencias, una secuencia de los resultados.(Definido por

Enumerable).

Comentarios

Nota

Para ver el código fuente de .NET Framework para este tipo, consulte el Reference Source. Puede examinar el código fuente en línea, descargue la referencia para verla sin conexión y recorrer (incluidas las revisiones y actualizaciones) durante la depuración; see instructions.

La List<T> clase es el equivalente genérico de la ArrayList clase. Implementa el IList<T> interfaz genérica mediante una matriz cuyo tamaño aumenta dinámicamente cuando es necesario.

Puede agregar elementos a un List<T> utilizando el Add o AddRange métodos.

La List<T> clase usa un comparador de igualdad y un comparador de orden.

- Métodos como Contains, IndexOf, LastIndexOf, y Remove utilizan un comparador de igualdad para los elementos de lista. El comparador de igualdad predeterminado para el tipo *T* se determina como sigue. Si tipo *T* implementa el IEquatable<T> interfaz genérica, el comparador de igualdad es el Equals(T) método de dicha interfaz; en caso contrario, el comparador de igualdad predeterminado es Object.Equals(Object).

- Métodos como BinarySearch y Sort utilizan un comparador de orden para los elementos de lista. El comparador predeterminado para el tipo *T* se determina como sigue. Si tipo *T* implementa el IComparable<T> interfaz genérica, el comparador predeterminado es la CompareTo(T) método de dicha interfaz; en caso contrario, si tipo *T* implementa la no genérica IComparable interfaz, el comparador predeterminado es la CompareTo(Object) método de dicha interfaz. Si tipo *T* no implementa ninguna interfaz, entonces no hay ningún comparador predeterminado y debe proporcionarse explícitamente un delegado de comparación o comparador.

El List<T> no se garantiza que se va a ordenar. Debe ordenar la List<T> antes de realizar operaciones (como BinarySearch) que requieren el List<T> esté ordenada.

Pueden tener acceso a los elementos de esta colección utilizando un índice entero. Índices de esta colección son de base cero.

Para grandes List<T> objetos, puede aumentar la capacidad máxima de 2 millones de elementos en un sistema de 64 bits estableciendo la **enabled** atributo del elemento de configuración a **true** en el entorno de tiempo de ejecución.

List<T> acepta **null** como un valor válido para la referencia de tipos y permite elementos duplicados.

Para obtener una versión inmutable de la List<T> de clases, consulte ImmutableList<T>.

Consideraciones sobre el rendimiento

Decidir si utilizar el List<T> o ArrayList (clase), que tienen una funcionalidad similar, recuerde que la List<T> clase funciona mejor en la mayoría de los casos y seguridad de tipos. Si se utiliza un tipo de referencia de tipo *T* de la List<T> (clase), el comportamiento de las dos clases es idéntico. Sin embargo, si se utiliza un tipo de valor de tipo *T*, debe tener en cuenta los problemas de implementación y la conversión boxing.

Si se utiliza un tipo de valor de tipo *T*, el compilador genera una implementación de la List<T> específicamente para ese tipo de valor. Es decir, un elemento de lista de un List<T> objeto no tiene que aplicar la conversión boxing antes de que el elemento se puede utilizar, y después de unos 500 elementos de lista se crean la memoria que se guarda no conversión boxing de los elementos de lista es mayor que la memoria utilizada para generar la implementación de la clase.

Compruebe el tipo de valor utilizado para el tipo *T* implementa el IEquatable<T> interfaz genérica. Si no, los métodos como Contains debe llamar a la Object.Equals(Object) método, que los cuadros del elemento de lista afectado. Si el tipo de valor implementa la IComparable interfaz y posee el código fuente, también implementan la IComparable<T> interfaz genérica para evitar el BinarySearch y Sort métodos de conversión boxing de los elementos de lista. Si no dispone del código fuente, pasar un IComparer<T> de objeto para el BinarySearch y Sort métodos

Es una ventaja de usar la implementación específica del tipo de la List<T> clase en lugar de utilizar la ArrayList clase o escribir una colección de contenedor fuertemente tipado usted mismo. El motivo es que la propia implementación debe hacer lo que .NET Framework ya realiza por usted, y common language runtime puede compartir código de lenguaje intermedio de Microsoft y los metadatos, que su implementación no puede hacer.

Consideraciones de F

La List<T> clase se utiliza con poca frecuencia en el código de F #. En su lugar, Lists (F#), que son listas inmutables vinculadas individualmente, se suele preferir. Una lista de F # proporciona una serie ordenada e inmutable de valores y se puede usar en el desarrollo de estilo funcional. Cuando se utiliza en F #, laList<T> clase normalmente se conoce por el ResizeArray<'T> escriba abreviatura para evitar conflictos con listas de F #

Ejemplos

En el ejemplo siguiente se muestra cómo agregar, quitar e insertar un objeto comercial simple en un List<T>.

C#
```
using System;
```

```
using System.Collections.Generic;
// Simple business object. A PartId is used to identify the type of part
// but the part name can change.
public class Part : IEquatable<Part>
    {
        public string PartName { get; set; }

        public int PartId { get; set; }

        public override string ToString()
        {
            return "ID: " + PartId + "   Name: " + PartName;
        }
        public override bool Equals(object obj)
        {
            if (obj == null) return false;
            Part objAsPart = obj as Part;
            if (objAsPart == null) return false;
            else return Equals(objAsPart);
        }
        public override int GetHashCode()
        {
            return PartId;
        }
        public bool Equals(Part other)
        {
            if (other == null) return false;
            return (this.PartId.Equals(other.PartId));
        }
    // Should also override == and != operators.

    }
public class Example
{
    public static void Main()
    {
        // Create a list of parts.
        List<Part> parts = new List<Part>();

        // Add parts to the list.
        parts.Add(new Part() {PartName="crank arm", PartId=1234});
        parts.Add(new Part() { PartName = "chain ring", PartId = 1334 });
        parts.Add(new Part() { PartName = "regular seat", PartId = 1434 });
        parts.Add(new Part() { PartName = "banana seat", PartId = 1444 });
        parts.Add(new Part() { PartName = "cassette", PartId = 1534 });
        parts.Add(new Part() { PartName = "shift lever", PartId = 1634 });

        // Write out the parts in the list. This will call the overridden
ToString method
        // in the Part class.
        Console.WriteLine();
        foreach (Part aPart in parts)
        {
            Console.WriteLine(aPart);
        }
```

```
        // Check the list for part #1734. This calls the IEquitable.Equals
method
        // of the Part class, which checks the PartId for equality.
        Console.WriteLine("\nContains(\"1734\"): {0}",
        parts.Contains(new Part {PartId=1734, PartName="" }));

        // Insert a new item at position 2.
        Console.WriteLine("\nInsert(2, \"1834\")");
        parts.Insert(2, new Part() { PartName = "brake lever", PartId =
1834 });


        //Console.WriteLine();
        foreach (Part aPart in parts)
        {
            Console.WriteLine(aPart);
        }

        Console.WriteLine("\nParts[3]: {0}", parts[3]);

        Console.WriteLine("\nRemove(\"1534\")");

        // This will remove part 1534 even though the PartName is different,
        // because the Equals method only checks PartId for equality.
        parts.Remove(new Part(){PartId=1534, PartName="cogs"});

        Console.WriteLine();
        foreach (Part aPart in parts)
        {
            Console.WriteLine(aPart);
        }
        Console.WriteLine("\nRemoveAt(3)");
        // This will remove the part at index 3.
        parts.RemoveAt(3);

        Console.WriteLine();
        foreach (Part aPart in parts)
        {
            Console.WriteLine(aPart);
        }

            /*

             ID: 1234   Name: crank arm
             ID: 1334   Name: chain ring
             ID: 1434   Name: regular seat
             ID: 1444   Name: banana seat
             ID: 1534   Name: cassette
             ID: 1634   Name: shift lever

             Contains("1734"): False

             Insert(2, "1834")
             ID: 1234   Name: crank arm
             ID: 1334   Name: chain ring
             ID: 1834   Name: brake lever
             ID: 1434   Name: regular seat
             ID: 1444   Name: banana seat
```

```
            ID: 1534   Name: cassette
            ID: 1634   Name: shift lever

            Parts[3]: ID: 1434   Name: regular seat

            Remove("1534")

            ID: 1234   Name: crank arm
            ID: 1334   Name: chain ring
            ID: 1834   Name: brake lever
            ID: 1434   Name: regular seat
            ID: 1444   Name: banana seat
            ID: 1634   Name: shift lever

            RemoveAt(3)

            ID: 1234   Name: crank arm
            ID: 1334   Name: chain ring
            ID: 1834   Name: brake lever
            ID: 1444   Name: banana seat
            ID: 1634   Name: shift lever


        */

    }
}
```

En el ejemplo siguiente se muestra varias propiedades y métodos de la List<T> clase genérica de tipo cadena. (Para obtener un ejemplo de un List<T> de tipos complejos, vea la Contains método.)

El constructor predeterminado se utiliza para crear una lista de cadenas con la capacidad predeterminada. El Capacity se muestra la propiedad y, a continuación, el Add método se usa para agregar varios elementos. Se enumeran los elementos y el Capacity propiedad se muestra de nuevo, junto con el Count propiedad, para mostrar que ha aumentado la capacidad según sea necesario.

El Contains método se utiliza para comprobar la presencia de un elemento en la lista, el Insert método se utiliza para insertar un nuevo elemento en mitad de la lista y se volverá a mostrar el contenido de la lista.

El valor predeterminado Item propiedad (el indizador en C#) se utiliza para recuperar un elemento, el Remove método se utiliza para quitar la primera instancia del elemento duplicado agregado anteriormente y se muestra el contenido nuevo. El Remove método siempre quita la primera instancia que encuentra.

El TrimExcess método se utiliza para reducir la capacidad para que coincida con el recuento y el Capacity y Count se muestran las propiedades. Si la capacidad no utilizada hubiera sido menos del 10% de la capacidad total, la lista no habría tamaño ha cambiado.

Por último, el Clear método se utiliza para quitar todos los elementos de la lista y el Capacity y Count se muestran las propiedades.

C#
```
using System;
using System.Collections.Generic;

public class Example
{
    public static void Main()
    {
        List<string> dinosaurs = new List<string>();

        Console.WriteLine("\nCapacity: {0}", dinosaurs.Capacity);

        dinosaurs.Add("Tyrannosaurus");
        dinosaurs.Add("Amargasaurus");
        dinosaurs.Add("Mamenchisaurus");
        dinosaurs.Add("Deinonychus");
        dinosaurs.Add("Compsognathus");
        Console.WriteLine();
        foreach(string dinosaur in dinosaurs)
        {
            Console.WriteLine(dinosaur);
        }

        Console.WriteLine("\nCapacity: {0}", dinosaurs.Capacity);
        Console.WriteLine("Count: {0}", dinosaurs.Count);

        Console.WriteLine("\nContains(\"Deinonychus\"): {0}",
            dinosaurs.Contains("Deinonychus"));

        Console.WriteLine("\nInsert(2, \"Compsognathus\")");
        dinosaurs.Insert(2, "Compsognathus");

        Console.WriteLine();
        foreach(string dinosaur in dinosaurs)
        {
            Console.WriteLine(dinosaur);
        }

        // Shows accessing the list using the Item property.
        Console.WriteLine("\ndinosaurs[3]: {0}", dinosaurs[3]);

        Console.WriteLine("\nRemove(\"Compsognathus\")");
        dinosaurs.Remove("Compsognathus");

        Console.WriteLine();
        foreach(string dinosaur in dinosaurs)
        {
            Console.WriteLine(dinosaur);
        }

        dinosaurs.TrimExcess();
        Console.WriteLine("\nTrimExcess()");
        Console.WriteLine("Capacity: {0}", dinosaurs.Capacity);
        Console.WriteLine("Count: {0}", dinosaurs.Count);
```

```
        dinosaurs.Clear();
        Console.WriteLine("\nClear()");
        Console.WriteLine("Capacity: {0}", dinosaurs.Capacity);
        Console.WriteLine("Count: {0}", dinosaurs.Count);
    }
}

/* This code example produces the following output:

Capacity: 0

Tyrannosaurus
Amargasaurus
Mamenchisaurus
Deinonychus
Compsognathus

Capacity: 8
Count: 5

Contains("Deinonychus"): True

Insert(2, "Compsognathus")

Tyrannosaurus
Amargasaurus
Compsognathus
Mamenchisaurus
Deinonychus
Compsognathus

dinosaurs[3]: Mamenchisaurus

Remove("Compsognathus")

Tyrannosaurus
Amargasaurus
Mamenchisaurus
Deinonychus
Compsognathus

TrimExcess()
Capacity: 5
Count: 5

Clear()
Capacity: 5
Count: 0
 */
```

Seguridad para subprocesos

Público estático (**Shared** en Visual Basic) de este tipo es seguro para subprocesos. No se garantiza que los miembros de instancias sean seguros para la ejecución de subprocesos.

Es seguro realizar varias operaciones de lectura en un List<T>, pero se pueden producir problemas si se modifica la colección mientras se está leyendo. Para garantizar la seguridad de subprocesos, bloquear la colección durante una lectura o la operación de escritura. Para habilitar una colección para tener acceso a varios subprocesos para leer y escribir, debe implementar su propia sincronización. Para las colecciones con sincronización integrada, vea las clases en el System.Collections.Concurrent espacio de nombres. Para una alternativa inherentemente: seguridad de subprocesos, vea la ImmutableList<T> clase.

Ver también

IList
ImmutableList<T>
Espacio de nombres System.Collections.Generic
Realizar operaciones de cadenas que no tienen en cuenta las referencias culturales en colecciones
Iteradores (C# y Visual Basic)

Volver al principio