3/11/2015

David Subires Parra

Programación de servicios software

Memoria práctica 02

Comparación colecciones genéricas

- Compare las clases de colecciones genéricas en .NET existentes, y en base a sus implementaciones, analice los respectivos rendimientos en las operaciones (inserción, eliminación y búsqueda de elementos).

A continuación se detallan los tipos de colección existentes:

* [List<T>](https://msdn.microsoft.com/es-es/library/6sh2ey19(v=vs.110).aspx) es la clase genérica que se corresponde con [ArrayList](https://msdn.microsoft.com/es-es/library/system.collections.arraylist(v=vs.110).aspx).
* [Dictionary<TKey, TValue>](https://msdn.microsoft.com/es-es/library/xfhwa508(v=vs.110).aspx) y [ConcurrentDictionary<TKey, TValue>](https://msdn.microsoft.com/es-es/library/dd287191(v=vs.110).aspx) son las clases genéricas que se corresponden con [Hashtable](https://msdn.microsoft.com/es-es/library/system.collections.hashtable(v=vs.110).aspx).
* [Collection<T>](https://msdn.microsoft.com/es-es/library/ms132397(v=vs.110).aspx) es la clase genérica que se corresponde con [CollectionBase](https://msdn.microsoft.com/es-es/library/system.collections.collectionbase(v=vs.110).aspx). [Collection<T>](https://msdn.microsoft.com/es-es/library/ms132397(v=vs.110).aspx) puede utilizarse como una clase base, pero, a diferencia de [CollectionBase](https://msdn.microsoft.com/es-es/library/system.collections.collectionbase(v=vs.110).aspx), no es abstracta. Esto la hace mucho más fácil de usar.
* [ReadOnlyCollection<T>](https://msdn.microsoft.com/es-es/library/ms132474(v=vs.110).aspx) es la clase genérica que se corresponde con [ReadOnlyCollectionBase](https://msdn.microsoft.com/es-es/library/system.collections.readonlycollectionbase(v=vs.110).aspx). [ReadOnlyCollection<T>](https://msdn.microsoft.com/es-es/library/ms132474(v=vs.110).aspx) no es abstracta y tiene un constructor que hace más fácil exponer una [List<T>](https://msdn.microsoft.com/es-es/library/6sh2ey19(v=vs.110).aspx) existente como una colección de solo lectura.
* Las clases genéricas [Queue<T>](https://msdn.microsoft.com/es-es/library/7977ey2c(v=vs.110).aspx), [ConcurrentQueue<T>](https://msdn.microsoft.com/es-es/library/dd267265(v=vs.110).aspx), [Stack<T>](https://msdn.microsoft.com/es-es/library/3278tedw(v=vs.110).aspx), [ConcurrentStack<T>](https://msdn.microsoft.com/es-es/library/dd267331(v=vs.110).aspx) y [SortedList<TKey, TValue>](https://msdn.microsoft.com/es-es/library/ms132319(v=vs.110).aspx) se corresponden con las respectivas clases no genéricas de igual nombre.

También hay varios tipos de colección genéricos que no tienen un tipo homólogo no genérico:

* [LinkedList<T>](https://msdn.microsoft.com/es-es/library/he2s3bh7(v=vs.110).aspx) es una lista vinculada de uso general que proporciona operaciones de eliminación e inserción O(1).
* [SortedDictionary<TKey, TValue>](https://msdn.microsoft.com/es-es/library/f7fta44c(v=vs.110).aspx) es un diccionario ordenado con operaciones de eliminación e inserción O(log *n*), lo que lo convierte en una alternativa útil a[SortedList<TKey, TValue>](https://msdn.microsoft.com/es-es/library/ms132319(v=vs.110).aspx).
* [KeyedCollection<TKey, TItem>](https://msdn.microsoft.com/es-es/library/ms132438(v=vs.110).aspx) es un híbrido entre una lista y un diccionario, lo que proporciona una manera de almacenar objetos que contienen sus propias claves.
* [BlockingCollection<T>](https://msdn.microsoft.com/es-es/library/dd267312(v=vs.110).aspx) implementa una clase de colección con funcionalidad de límite y bloqueo.
* [ConcurrentBag<T>](https://msdn.microsoft.com/es-es/library/dd381779(v=vs.110).aspx) proporciona una rápida inserción y eliminación de elementos no ordenados.

- Explique el concepto de Varianza y Covarianza en parámetros de interfaces genéricas y proponga algún nuevo ejemplo.

Covarianza: nos permite usar un tipo más derivado que el especificado originalmente.

Puede asignar una instancia de IEnumerable<Derived> (IEnumerable(Of Derived) en Visual Basic) a una variable de tipo IEnumerable<Base>.

Un parámetro de tipo covariante se marca con la palabra clave **out** (palabra clave **Out** en Visual Basic, **+** para el [Ensamblador de MSIL](https://msdn.microsoft.com/es-es/library/496e4ekx(v=vs.110).aspx)). Puede usar un parámetro de tipo covariante como el valor devuelto de un método que pertenece a una interfaz o como el tipo de valor devuelto de un delegado. No puede usar un parámetro de tipo covariante como una restricción de tipo genérico para los métodos de interfaz.

Un parámetro de tipo contravariante se marca con la palabra clave **in** (palabra clave **In** en Visual Basic, **-** para el [Ensamblador de MSIL](https://msdn.microsoft.com/es-es/library/496e4ekx(v=vs.110).aspx)). Puede usar un parámetro de tipo contravariante como el tipo de un parámetro de un método que pertenece a una interfaz o como el tipo de un parámetro de un delegado. Puede usar un parámetro de tipo contravariante como una restricción de tipo genérico para un método de interfaz.

[Interfaces genéricas con parámetros de tipo covariante](javascript:void(0)): a partir de .NET Framework 4, varias interfaces genéricas tienen parámetros de tipo covariante; por ejemplo: [IEnumerable<T>](https://msdn.microsoft.com/es-es/library/9eekhta0(v=vs.110).aspx), [IEnumerator<T>](https://msdn.microsoft.com/es-es/library/78dfe2yb(v=vs.110).aspx), [IQueryable<T>](https://msdn.microsoft.com/es-es/library/bb351562(v=vs.110).aspx) y[IGrouping<TKey, TElement>](https://msdn.microsoft.com/es-es/library/bb344977(v=vs.110).aspx). Todos los parámetros de tipo de estas interfaces son covariantes, por lo que los parámetros de tipo se usan únicamente para los tipos de valor devuelto de los miembros.

- Describa métodos para ordenar una Lista genérica (utilizando .NET).

Ordena los elementos en la [List<T>](https://msdn.microsoft.com/es-es/library/6sh2ey19(v=vs.110).aspx) o en una parte de ella. Este método utiliza el comparador predeterminado [Comparer<T>.Default](https://msdn.microsoft.com/es-es/library/azhsac5f(v=vs.110).aspx) para el tipo *T* con el fin de determinar el orden de los elementos de la lista. La propiedad[Comparer<T>.Default](https://msdn.microsoft.com/es-es/library/azhsac5f(v=vs.110).aspx) comprueba si el tipo *T* implementa la interfaz genérica [IComparable<T>](https://msdn.microsoft.com/es-es/library/4d7sx9hd(v=vs.110).aspx) y, si está disponible, utiliza esa implementación. En caso contrario,[Comparer<T>.Default](https://msdn.microsoft.com/es-es/library/azhsac5f(v=vs.110).aspx) comprueba si el tipo *T* implementa la interfaz [IComparable](https://msdn.microsoft.com/es-es/library/system.icomparable(v=vs.110).aspx). Si el tipo *T* no implementa ninguna de las dos interfaces, [Comparer<T>.Default](https://msdn.microsoft.com/es-es/library/azhsac5f(v=vs.110).aspx) produce una excepción [InvalidOperationException](https://msdn.microsoft.com/es-es/library/system.invalidoperationexception(v=vs.110).aspx).

Este método utiliza el método de [Array.Sort](https://msdn.microsoft.com/es-es/library/system.array.sort(v=vs.110).aspx) , que aplica la ordenación introspectiva como sigue:

* Si el tamaño de la partición es menor que 16 elementos, utiliza un algoritmo de ordenación de inserción.
* Si el número de particiones supera 2 \* LogN, donde es el intervalo *f* de la matriz de entrada, utiliza un algoritmo de Heapsort.
* Si no, utiliza un algoritmo de Quicksort.

Esta implementación realiza una ordenación inestable; es decir, si hay dos elementos iguales, es posible que no se mantenga su orden. En contraste, una ordenación estable conserva el orden de los elementos que son iguales.

Este método suele ser una operación O(*n* log *n*), donde *n* es [Count](https://msdn.microsoft.com/es-es/library/27b47ht3(v=vs.110).aspx); aunque en el peor de los casos es una operación O(*n*^ 2).

|  |  |
| --- | --- |
| [Sort()](https://msdn.microsoft.com/es-es/library/b0zbh7b6(v=vs.110).aspx) | Ordena los elementos de todo el objeto [List<T>](https://msdn.microsoft.com/es-es/library/6sh2ey19(v=vs.110).aspx) utilizando el comparador predeterminado. |
| [Sort(Comparison<T>)](https://msdn.microsoft.com/es-es/library/w56d4y5z(v=vs.110).aspx) | Ordena los elementos de todo el objeto [List<T>](https://msdn.microsoft.com/es-es/library/6sh2ey19(v=vs.110).aspx) utilizando el delegado [System.Comparison<T>](https://msdn.microsoft.com/es-es/library/tfakywbh(v=vs.110).aspx) especificado. |
| [Sort(IComparer<T>)](https://msdn.microsoft.com/es-es/library/234b841s(v=vs.110).aspx) | Ordena los elementos en la [List<T>](https://msdn.microsoft.com/es-es/library/6sh2ey19(v=vs.110).aspx) completa utilizando el comparador especificado. |
| [Sort(Int32, Int32, IComparer<T>)](https://msdn.microsoft.com/es-es/library/8ce6t5ad(v=vs.110).aspx) | Ordena los elementos en un intervalo de elementos de la matriz [List<T>](https://msdn.microsoft.com/es-es/library/6sh2ey19(v=vs.110).aspx) utilizando el comparador especificado. |