Collections, collection interfaces and yield return

Agenda

Collection and IEnumerable

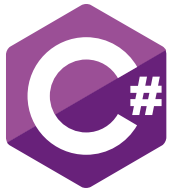
yield return and IEnumerable

Implement IEnumerable

ICollection interface

Collection performance

Collections and IEnumerable interface



.NET collections

Collection other than arrays

Not fixed size

Generic/Non generic

If possible, always use generic collections

Generic collections

Type safe

Holds same data type elements

Most used collections

- Lists
- Dictionaries
- Queues
- Stacks

Namespaces

System.Collections : legacy namespace that contains legacy non generic collections

System.Collections.Generics : generic collections namespace

System.Collections.Immutable : immutable collection namespace

System.Collections.Concurent : concurent collection namespace

Collection interfaces

```
IEnumerable<T>
ICollection<T>/IReadOnlyCollection<T>
IList<T> /IReadOnlyList<T>
IDictionary<T>/IReadOnlyDictionary<T>
```

IEnumeable
IEnumerable<T>

Base interface for generic collections

Allows to enumerate the elements of a collection

With a `foreach` statement

IEnumerable<T> members

```
GetEnumerator()
```

foreach statement

- Enumerates an `IEnumerable` object elements
- No control over the order of getting the elements. `for` statement for more control
- Elements can’t be modified
- You can use `break/jump` statements

foreach

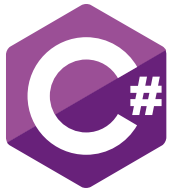
```
foreach (var planet in planets)
{
    Console.WriteLine(planet);
}
```

Demo

Declare/Instantiate IEnumerable objects

Use foreach statement

yield return and IEnumerable



yield return

Helps to return an element in a method or get accessor

The `yield return` indicates that the method where it appears is an iterator

yield return

```
public static IEnumerable<string> GetRockyPlanets()
{
    yield return "Mercury";
    yield return "Venus";
    yield return "Earth";
    yield return "Mars";
}
```

yield return

```
foreach (var planet in GetRockyPlanets())
{
    Console.WriteLine(planet);
}
```

yield return

The return type must be `IEnumerable`, `IEnumerable<T>`, `IEnumerator`, or `IEnumerator<T>`

The sequence returned by an iterator is consumed in a `foreach` statement

The `foreach` statement hides the complexity of enumerators

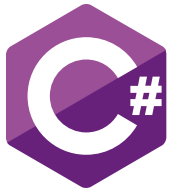
Demo

yield return statement

Demo

Implement IEnumerable interface

ICollection interface




IEnumerable<T>
ICollection<T>
IReadOnlyCollection<T>
IList<T>
IReadOnlyList<T>
IDictionary<T>
IReadOnlyDictionary<T>

ICollection
ICollection<T>

Base interface for all collection after `IEnumerable<T>`

Exposes additional features on collections

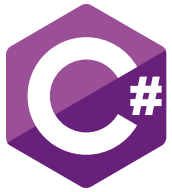
ICollection<T> members

```
Count
IsReadOnly
Add(T)
Clear()
Contains(T)
Remove(T)
…
```

Demo

Declare/Instantiate ICollection objects

Collection performance

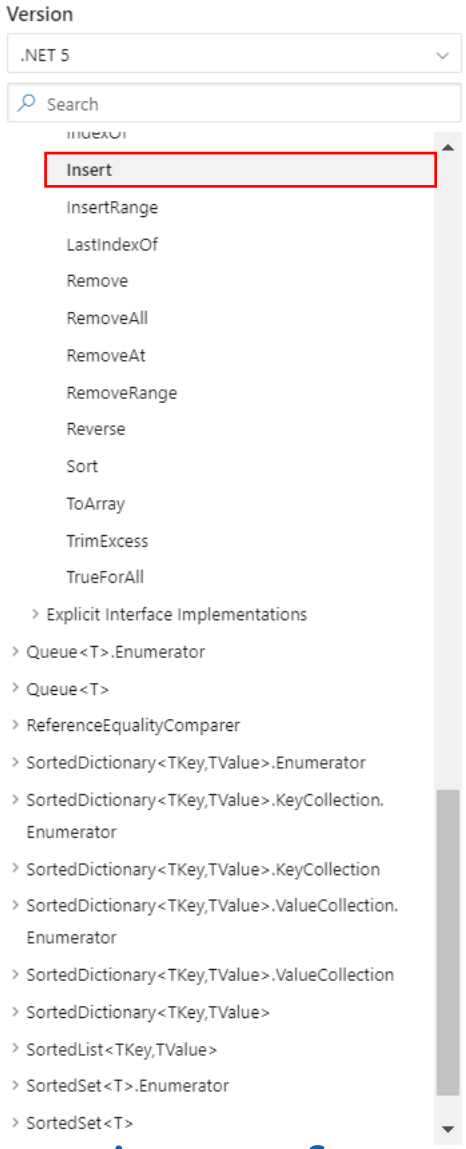


Collection performance

The behavior of a function to complete a problem of size n can be measured

Also called the factor of growth or the order

List Insert method performance



Remarks

List<T> accepts `null` as a valid value for reference types and allows duplicate elements.

If Count already equals Capacity, the capacity of the List<T> is increased by automatically reallocating the internal array, and the existing elements are copied to the new array before the new element is added.

If `index` is equal to Count, `item` is added to the end of List<T>.

This method is an O(*n*) operation, where *n* is Count.

Applies to

Product	Versions
.NET	5.0, 6.0 Preview 3
.NET Core	1.0, 1.1, 2.0, 2.1, 2.2, 3.0, 3.1
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8
.NET Standard	1.0, 1.1, 1.2, 1.3, 1.4, 1.6, 2.0, 2.1
UWP	10.0
Xamarin.Android	7.1
Xamarin.iOS	10.8
Xamarin.Mac	3.0

See also

- InsertRange(Int32, IEnumerable<T>)
- Add(T)
- Remove(T)

https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1?view=net-5.0

Big O Notation

O(1) constant

O(log(n)) logarithmic

O(n) linear

O(n²) quadratic

Summary

Collections are generic dynamic size data structure types

The IEnumerable and ICollection interfaces are base interfaces for all .NET collections

The yield return statement returns IEnumerable elements in a method (during the execution of a foreach loop)

The performance of an operation on a collection can be measured with the factor of growth