        - **Communication**
        - **Web Applications**
        - **Industrial**
        - **Household Equipments**
        - **Auto**
        - **Career and Certifications**
    - **Science**
        - **Biology**
        - **Chemistry**
        - **Health**
        - **Disease**
        - **Diet & Fitness**
        - **Drugs**
        - **Mathematics & Statistics**
        - **Nature**
        - **Animals**
        - **Birds**
        - **Physics**
        - **Psychology**
    - **Objects**
        - **Food**
        - **Processed Foods**
        - **Vegetables & Fruits**
    - **Language**
        - **Words**
        - **Grammar**

Difference Between IEnumerable and IEnumerator

• Categorized under **Technology** | **Difference Between IEnumerable and IEnumerator**

Before we jump into the difference between IEnumerable and IEnumerator, let us first understand the terms Enumerator and Enumerable and when and why to use them.

Enumerable is a module used in the Array class that gives you Enumerator. The class itself doesn't each method. In fact, it uses a method called 'GetEnumerator' that gives you an Enumerator to retrieve its item.

An Enumerator is an object that returns each item in a collection in a specific order, exactly as they are requested. The compiler generates code that uses an Enumerator, when you write a foreach loop in C#.

IEnumerator is an interface implemented by an enumerator and the enumerable class implements the IEnumerable interface.

What is Ienumerable?

IEnumerable is a basic interface used to obtain an object that knows how to enumerate (or iterate) over the elements in the collection.

It's an implementation of the iterator pattern that provides a mechanism to traverse an object in the collection regardless of its internal structure. The IEnumerable interface represents an object that can be iterated over in a collection.

It uses only one method "GetEnumerator" that returns an enumerator class instance that implement IEnumerator interface.

```
public MostRecentFile[] GetFileList()
{
    if (!Loaded)
    {
        LoadMruList();
    }
    object[] array = files.ToArray();
    MostRecentFile[] mrfArray = new MostRecentFile[array.Length];
    array.CopyTo(mrfArray, 0);
    return mrfArray;
}
public bool Contains(string fileName)
{
    if (!Loaded)
    {
        LoadMruList();
    }
    string lcFileName = fileName.ToLower();

    foreach (MostRecentFile mrf in files)
    {
        string lcMrf = mrf.FileName.ToLower();

        if (0 == String.Compare(lcMrf, lcFileName))
        {
            return true;
        }
    }
```

What is Ienumerator?

The IEnumerator interface, one the other hand, declares two methods – Reset ( ) and MoveNext ( ) – and one property, Current. The MoveNext ( ) returns a Boolean vale that indicates the end of the list and helps position the first element in the list after calling the Reset ( ) method – it set the enumerator to its default position so that to re-iterate the list from the beginning.

The Current property can only be called through an instance of the IEnumerator interface and it returns the current element in the list.

Difference between IEnumerable and IEnumerator

Basics of IEnumerable and Ienumerator

Both IEnumerable and IEnumerator are interfaces that implement the iterator software design pattern in the .Net Framework together. The .Net Framework makes accessing individual elements in the custom collection while implementing IEnumerable and IEnumerator interfaces. The IEnumerable interface declares only one method called GetEnumerator which returns another type of interface called the IEnumerator interface for that particular collection. IEnumerator, on the other hand, is the base interface for all non-generic enumerators which are used to read the data in the collection. However, enumerators cannot be used to modify the underlying collection.

Methods

IEnumerable interface defines only one method GetEnumerator ( ) which is an instance method used on several different collection types. It gets an IEnumerator iterator that can be used to iterate over all the values from the collection. When you write a foreach loop in C# the code it generates calls the GetEnumerator method to create the Enumerator used by the loop.

IEnumerator, on the other hand, uses two methods MoveNext ( ) and Reset ( ) and a property Current. The MoveNext ( ) method takes the enumerator to the next element of the collection, whereas Reset ( ) method sets the enumerator to its default position which is before the first element in the collection.

Implementation

IEnumerable is a generic interface that provides an abstraction for looping over elements and by implementing the IEnumerable interface, a generic class essentially enables iteration via the IEnumerator interface. In doing so, these classes end up providing a common interface to retrieve an instance of an IEnumerator <T> object that supports all the basic set of navigation methods.

IEnumerator is the base interface for enumerators and the use of IEnumerable interface requires that the class must implement the IEnumerator. Both interfaces need to be implemented, if you want to provide support for foreach. The abstract class ‘AbstractEnumerator’ implements the IEnumerator interface.

Functionality of IEnumerable and IEnumerator

The IEnumerable interface, along with supporting IEnumerator interface, allows you to iterate over the elements in the stack using the ‘foreach’ statement. It is the basic interface used for collection type objects. The iteration begins with the top element in the stack and ends with the oldest element in the stack. Simply put, it represents an object that can be enumerated.

Together with the IEnumerator interface, they enable the ArrayList elements to be iterated in a standardized, sequential fashion, starting with the first element and going forward. The foreach statement uses the methods and properties of the IEnumerator interface to iterate all elements in a collection.

IEnumerable vs. IEnumerator: Comparison Chart

Summary of

IEnumerable verses IEnumerator

In a nutshell, both IEnumerable and IEnumerator are interfaces used to enumerate or iterate a class that has a collection nature meaning they facilitate iterative access in a custom collection. IEnumerable is the generic interface available for collection type objects and by implementing the IEnumerable interface, a generic class essentially enables iteration via the IEnumerator interface. It uses only one method GetEnumerator that returns an enumerator class instance that implement IEnumerator interface. IEnumerator interface is meant to be used as accessors and it cannot be used to modify the underlying collection. It provides two abstract methods Reset ( ) and MoveNext ( ), and a property Current to call a particular element in a collection.

Author **Recent Posts**

Sagar Khillar

Sagar Khillar is a prolific content/article/blog writer working as a Senior Content Developer/Writer in a reputed client services firm based in India. He has that urge to research on versatile topics and develop high-quality content to make it the best read. Thanks to his passion for writing, he has over 7 years of professional experience in writing and editing services across a wide variety of print and electronic platforms.

Outside his professional life, Sagar loves to connect with people from different cultures and origin. You can say he is curious by nature. He believes everyone is a learning experience and it brings a certain excitement, kind of a curiosity to keep going. It may feel silly at first, but it loosens you up after a while and makes it easier for you to start conversations with total strangers – that's what he said."