

C# 中有Enumerable和 IEnumerable以及 IEnumerable<TSource>

Enumerable：意为可枚举的

这三个常常让人搞混，简单来说

Enumerable：

位于System.Core程序集，System.Linq下，是一个静态类，里面包含了很多linq的扩展方法

```
程序集 System.Core, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089

using ...

namespace System.Linq
{
    ...public static class Enumerable
    {
        ...public static TSource Aggregate<TSource>(this IEnumerable<TSource> source, Func<TSource, TSource, TSource> func, TSource seed)
        ...public static TAccumulate Aggregate<TSource, TAccumulate>(this IEnumerable<TSource> source, TAccumulate seed, Func<TAccumulate, TSource, TAccumulate> func)
        ...public static TResult Aggregate<TSource, TAccumulate, TResult>(this IEnumerable<TSource> source, TAccumulate seed, Func<TAccumulate, TSource, TAccumulate> func, Func<TAccumulate, TResult> resultSelector)
        ...public static bool All<TSource>(this IEnumerable<TSource> source, Func<TSource, bool> predicate)
        ...public static bool Any<TSource>(this IEnumerable<TSource> source)
        ...public static bool Any<TSource>(this IEnumerable<TSource> source, Func<TSource, bool> predicate)
        ...public static IEnumerable<TSource> AsEnumerable<TSource>(this IEnumerable<TSource> source);
    }
}
```

IEnumerable：

位于mscorlib程序集，System.Collections下，是一个接口，只有一个方法GetEnumerator()

```
程序集 mscorlib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089

using System.Runtime.InteropServices;

namespace System.Collections
{
    ...public interface IEnumerable
    {
        ...IEnumerator GetEnumerator();
    }
}
```

<https://blog.csdn.net/wcc27857285>

IEnumerable<TSource>：

位于mscorlib程序集，System.Collections.Generic下，是一个带泛型的接口，只有一个方法GetEnumerator()，而且也继承了IEnumerable接口

```
程序集 mscorlib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=
```

```
using System.Runtime.CompilerServices;
```

```
namespace System.Collections.Generic
{
    ...public interface IEnumerable<out T> : IEnumerable
    {
        ...IEnumerator<T> GetEnumerator();
    }
}
```

<https://blog.csdn.net/wcc27857285>

如何转化:

很明显Enumerable是无法和IEnumerable还有IEnumerable<TSource>相互转换的

但是IEnumerable还有IEnumerable<TSource>是可以的, 由于IEnumerable<TSource>继承IEnumerable, 所以前者转后者没有必要也毫无意义

所以我们主要讨论的是后者转化成前者

我们常用的集合如List, 是已经继承了IEnumerable<TSource>, 所以可以使用所以Linq的扩展方法, 如select,where等

但是比如DataTable.Rows,这只继承了IEnumerable, 若想对其进行linq操作, 首先必须得将其转化成IEnumerable<TSource>

如何转化, 代码:

```
DataTable tableData = new DataTable();
var list=tableData.Rows.Cast<DataRow>().Where(row => row["名称"].ToString().Equals("zhangsan"));
if(list.Count()==1)
{
}
```

可以通过Cast<>来转, 因为Cast<>方法是IEnumerable的扩展方法, 而datatable.Rows是间接继承了IEnumerable, 所以可以调用Cast

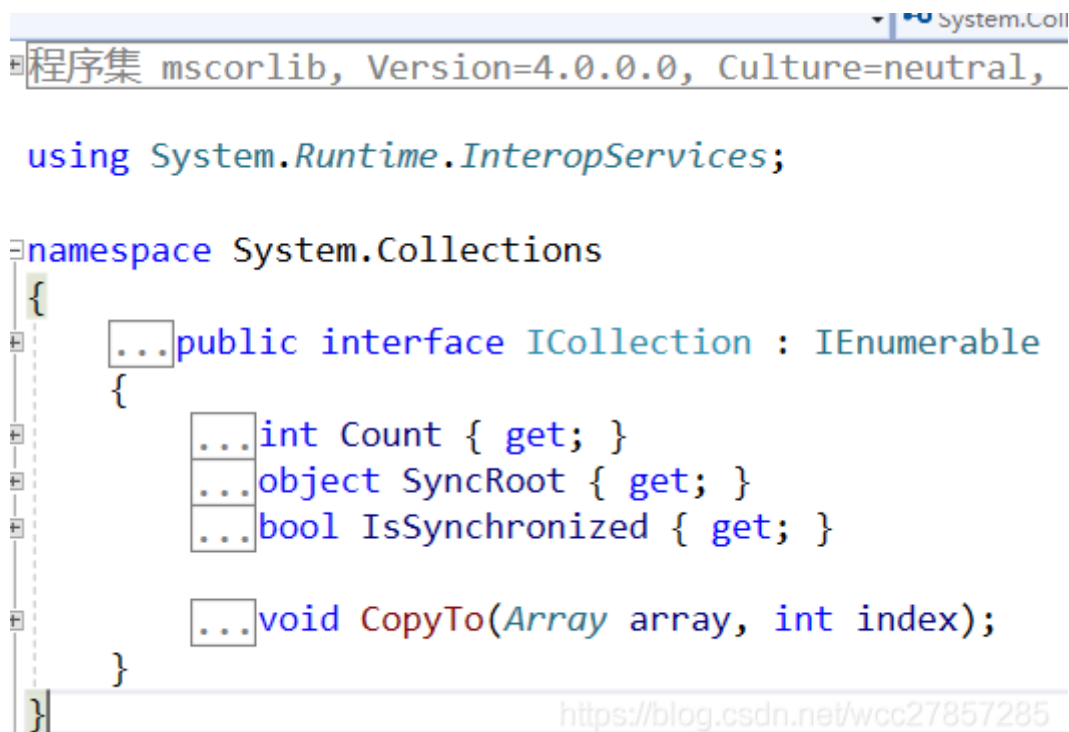
```
...public static float? Average(this IEnumerable<float?> source);
...public static IEnumerable<TResult> Cast<TResult>(this IEnumerable source);
...public static IEnumerable<TSource> Concat<TSource>(this IEnumerable<TSource> first,
```

总结:

在.NET中使用linq, 必须想办法将集合转化成IEnumerable<TSource>

幸运的是大多数集合都继承者IEnumerable和IEnumerable<TSource>

即便是ICollection，没错，ICollection也是继承IEnumerable，那么也可以通过Cast转化，如图：



```
程序集 mscorlib, Version=4.0.0.0, Culture=neutral,
using System.Runtime.InteropServices;

namespace System.Collections
{
    ...public interface ICollection : IEnumerable
    {
        ...int Count { get; }
        ...object SyncRoot { get; }
        ...bool IsSynchronized { get; }

        ...void CopyTo(Array array, int index);
    }
}
```

<https://blog.csdn.net/wcc27857285>

也就是说，.NET中绝大多数的集合都可以使用Linq