IEnumerable and IEnumerator and Iquerable & Yield

IEnumerable Internally uses IEnumerator

& Ienumerator have Current State of Execution while IEnumerable does not have

List<int> lst = new List<int>() { 1990,1991,1992,1993,1994,2001,2002,2003 };

public void readUsingIEnumerable()

{

foreach(var i in lst.AsEnumerable())

{

Console.WriteLine(i);

}

}

static void Main(string[] args)

{

Program p = new Program();

p.readUsingIEnumerable();

}

Now suppose after reading till 2000 we want to pass lst to other method and process after 2000 then then Enumerable object need be traverse agin but in case of IEnumerator having State so it will start traverse from 2000

public void readTill2000(IEnumerator<int> o)

{

while (o.MoveNext())

{

if(o.Current<=2000)

{

Console.WriteLine(o.Current.ToString());

}

else

{

readAfter2000(o);

}

}

}

public void readAfter2000(IEnumerator<int> o)

{

while (o.MoveNext())

{

Console.WriteLine(o.Current.ToString());

}

}

IEnumerable Fetch full table from data base then apply filter but in case of Iquerable it apply filter in Database

Yield :

It give statefull iteration , we get response as soon as it available , suppose we hit a DB and it traverseing some data there when particular sequence found we want to get that instead of getting full object and return over that .

IEnumerable<int> IteratorBlock()

{

Console.WriteLine("Begin");

yield return 1;

Console.WriteLine("After 1");

yield return 2;

Console.WriteLine("After 2");

yield return 42;

Console.WriteLine("End");

}

foreach (var i in IteratorBlock())

Console.WriteLine(i);

public static IEnumerable<int> CreateCollectionWithList()

{

var list = new List<int>();

list.Add(10);

list.Add(0);

list.Add(1);

list.Add(2);

list.Add(20);

return list;

}

public static IEnumerable<int> CreateCollectionWithYield()

{

yield return 10;

for (int i = 0; i < 3; i++)

{

yield return i;

}

yield return 20;

}