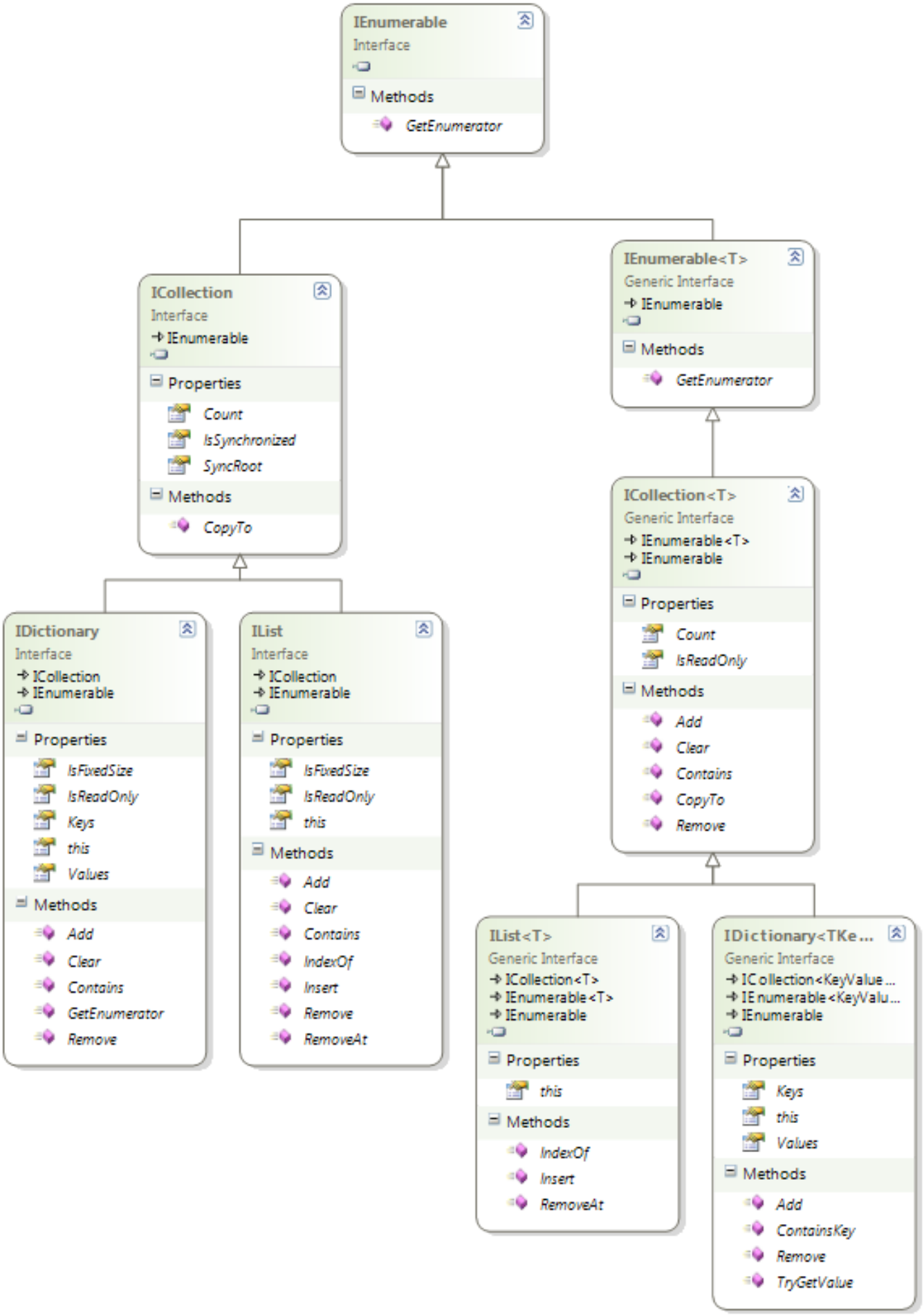
عنوان: **اشیاء Enumerable و Enumerator و استفاده از قابلیت‌های yield (قسمت اول)**
نویسنده: ابراهیم بیاگوی
تاریخ: ۱۳۹۱/۰۵/۱۷ ۱۸:۲۵
آدرس: www.dotnettips.info
برچسب‌ها: C#, LINQ, Enumerator, Enumerable

در این مقاله می‌خواهیم نحوهٔ ساخت اشیایی با خصوصیات Enumerable را بررسی کنیم. بررسی ویژگی این اشیاء دارای اهمیت است حداقل به این دلیل که پایهٔ یکی از قابلیت مهم زبانی سی‌شارپ یعنی LINQ هستند. برای یافتن پیش‌زمینه‌ای در این موضوع خواندن این مقاله‌های بسیار خوب (۱ و ۲) نیز توصیه می‌شود.

Enumerableها

اشیاء Enumerable یا به‌عبارت دیگر اشیائی که اینترفیس IEnumerable را پیاده‌سازی می‌کنند، دامنهٔ گسترده‌ای از Collectionهای CLI را شامل می‌شوند. همانطور که در نمودار زیر نیز می‌توانید مشاهده کنید IEnumerable (از نوع غیر Generic آن) در بالای سلسله مراتب اینترفیس‌های Collectionهای CLI قرار دارد:


IEnumerable
Interface
Methods
GetEnumerator
ICollection
Interface
IEnumerable
Properties
Count
IsSynchronized
SyncRoot
Methods
CopyTo
IEnumerable<T>
Generic Interface
IEnumerable
Methods
GetEnumerator
ICollection<T>
Generic Interface
IEnumerable<T>
IEnumerable
Properties
Count
IsReadOnly
Methods
Add
Clear
Contains
CopyTo
Remove
IDictionary
Interface
ICollection
IEnumerable
Properties
IsFixedSize
IsReadOnly
Keys
this
Values
Methods
Add
Clear
Contains
GetEnumerator
Remove
IList
Interface
ICollection
IEnumerable
Properties
IsFixedSize
IsReadOnly
this
Methods
Add
Clear
Contains
IndexOf
Insert
Remove
RemoveAt
IList<T>
Generic Interface
ICollection<T>
IEnumerable<T>
IEnumerable
Properties
this
Methods
IndexOf
Insert
RemoveAt
IDictionary<TKe...
Generic Interface
ICollection<KeyValue...
IEnumerable<KeyValu...
IEnumerable
Properties
Keys
this
Values
Methods
Add
ContainsKey
Remove
TryGetValue

درخت اینترفیس‌های Collectionها در CLI منبع

IEnumerableها همچنین دارای اهمیت دیگری نیز هستند؛ قابلیت‌های LINQ که از دات‌نت ۳.۵ به دات‌نت اضافه شدند به‌عنوان Extensionهای این اینترفیس تعریف شده‌اند و پیاده‌سازی Linq to Objects را می‌توانید در کلاس استاتیک System.Linq.Enumerable در System.Core مشاهده کنید. (می‌توانید برای دیدن آن را با ILDasm یا Reflector باز کنید یا پیاده‌سازی آزاد آن در پروژهٔ Mono را اینجا مشاهده کنید که برای شناخت بیشتر LINQ واقعاً مفید است.)

همچنین این Enumerableها هستند که foreach را امکان‌پذیر می‌کنند. به عبارتی دیگر هر شئ‌ای که قرار باشد در foreach (var x in object) قرار بگیرد و بدین طریق اشیاء درونی‌اش را برای پیمایش یا عملی خاص قرار دهد باید Enumerable باشد.

همانطور که قبلاً هم اشاره شد IEnumerable از نوع غیر Generic در بالای نمودار Collectionها قرار دارد و حتی IEnumerable از نوع Generic نیز باید آن را پشتیبانی کند. این موضوع به احتمال به این دلیل در طراحی لحاظ شد که مهاجرت به .NET 2.0 که قابلیت‌های Generic را افزوده بود ساده‌تر کند. IEnumerable همچنین قابلیت covariance که از قابلیت‌های جدید C# 4.0 هست را دارا است (در اصل IEnumerable دارای Generic از نوع out است).

Enumerableها همانطور که از اسم اینترفیس IEnumerable انتظار می‌رود اشیایی هستند که می‌توانند یک شئ Enumerator که IEnumerator را پیاده‌سازی کرده‌است را از خود ارائه دهند. پس طبیعی است برای فهم و درک دلیل وجودی Enumerable باید Enumerator را بررسی کنیم.

Enumeratorها

Enumerator شئ است که در یک پیمایش یا به‌عبارت دیگر گذر از روی تک‌تک عضوها ایجاد می‌شود که با حفظ موقعیت فعلی و پیمایش امکان ادامهٔ پیمایش را برای ما فراهم می‌آورد. اگر بخواهید آن را در حقیقت بازسازی کنید شئ Enumerator به‌مانند کاغذ یا جسمی است که بین صفحات یک کتاب قرار می‌دهید که مکانی که در آن قرار دارید را گم نکنید؛ در این مثال، Enumerable همان کتاب است که قابلیت این را دارد که برای پیمایش به وسیلهٔ قرار دادن یک جسم در وسط آن را دارد.

حال برای اینکه دید بهتری از رابطهٔ بین Enumerable و Enumerator از نظر برنامه‌نویسی به این موضوع پیدا کنیم یک کد نمونهٔ عملی را بررسی می‌کنیم.

در اینجا نمونهٔ ساده و خوانایی از استفاده از یک List برای پیشمایش تمامی اعداد قرار دارد:

```
List<int> list = new List<int>();
list.Add(1);
list.Add(2);
list.Add(3);
foreach (int i in list)
{
    Console.WriteLine(i);
}
```

همانطور که قبلاً اشاره foreach نیاز به یک Enumerable دارد و List هم با پیاده‌سازی IList که گسترشی از IEnumerable هست نیز یک نوع Enumerable هست. اگر این کد را Compile کنیم و IL آن را بررسی کنیم متوجه می‌شویم که CLI در اصل چنین کدی را برای اجرا می‌بینید:

```
List<int> list = new List<int>();
list.Add(1);
list.Add(2);
list.Add(3);
IEnumerator<int> listIterator = list.GetEnumerator();
while (listIterator.MoveNext())
{
    Console.WriteLine(listIterator.Current);
```

```
}
listIterator.Dispose();
```

(می‌توان از using استفاده نمود که Dispose را خود انجام دهد که اینجا برای سادگی استفاده نشده‌است.)

همانطور که می‌بینیم یک Enumerator برای Enumerable ما (یعنی List) ایجاد شد و پس از آن با پرسش این موضوع که آیا این پیمایش امکان ادامه دارد، کل اعضا پیموده‌شده و عمل مورد نظر ما بر آن‌ها انجام شده‌است.

خب، تا اینجای کار با خصوصیات و اهمیت Enumeratorها و Enumerableها آشنا شدیم، حال نوبت به آن می‌رسد که بررسی کنیم آن‌ها را چگونه می‌سازند و بعد از آن با کاربردهای فراتری از آن‌ها نسبت به پیمایش یک List آشنا شویم.

ساخت Enumeratorها و Enumerableها

همانطور که اشاره شد ایجاد اشیاء Enumerable به اشیاء Enumerator مربوط است، پس ما در یک قطعه کد که پیمایش از روی یک آرایه را فراهم می‌آورد ایجاد هر دوی آن‌ها و رابطهٔ بینشان را بررسی می‌کنیم.

```
public class ArrayEnumerable<T> : IEnumerable<T>
    {
        private T[] _array;
        public ArrayEnumerable(T[] array)
        {
            _array = array;
        }


        public IEnumerator<T> GetEnumerator()
        {
            return new ArrayEnumerator<T>(_array);
        }

        System.Collections.IEnumerator System.Collections.IEnumerable.GetEnumerator()
        {
            return GetEnumerator();
        }
    }

    public class ArrayEnumerator<T> : IEnumerator<T>
    {
        private T[] _array;
        public ArrayEnumerator(T[] array)
        {
            _array = array;
        }

        public int index = -1;

        public T Current { get { return _array[index]; } }

        object System.Collections.IEnumerator.Current { get { return this.Current; } }

        public bool MoveNext()
        {
            index++;
            return index < _array.Length;
        }

        public void Reset()
        {
            index = 0;
        }

        public void Dispose() { }
    }
```

ادامه

نظرات خوانندگان

نویسنده: مرتضی
تاریخ: ۱۳۹۱/۰۵/۱۷ ۱۹:۲۰

درخت اینترفیس‌های Collectionها در سی‌شارپ منبع: http://www.mbaldinger.com/post/NET-Collection-Interface-Hierarchy.aspx
بجای سی‌شارپ به دانت‌نت تغییرش بدید
درخت اینترفیس‌های Collectionها در دانت‌نت

نویسنده: ابراهیم بیاگوی
تاریخ: ۱۳۹۱/۰۵/۱۷ ۱۹:۲۹

من شخصاً اطمینان ندارم که همهٔ زبان‌های CLI از همین Collectionها استفاده کنند و البته این نمودار با Syntax سی‌شارپ بود به همین دلیل سی‌شارپ نوشته بودم با این حال آن را به Collectionهای CLI تبدیل کردم.