عنوان: **سرعت واکشی اطلاعات در List و Dictionary**
نویسنده: شاهین کیاست
تاریخ: ۱۳۹۲/۰۳/۱۷ ۱۳:۵
آدرس: www.dotnettips.info
برچسب‌ها: C#, LINQ, .NET, Performance

دسترسی به داده‌ها پیش شرط انجام همه‌ی منطق‌های اکثر نرم افزارهای تجاری می‌باشد. داده‌های ممکن در حافظه ، پایگاه داده ، فایل‌های فیزیکی و هر منبع دیگری قرار گرفته باشند.
هنگامی که حجم داده‌ها کم باشد شاید روش دسترسی و الگوریتم مورد استفاده اهمیتی نداشته باشد اما با افزایش حجم داده‌ها روش‌های بهینه‌تر تاثیر مستقیم در کارایی برنامه دارند.
در این مثال سعی بر این است که در یک سناریوی خاص تفاوت بین Dictionary و List را بررسی کنیم :
فرض کنید 2 کلاس Student و Grade موجود است که وظیفه‌ی نگهداری اطلاعات دانش آموز و نمره را بر عهده دارند.

```
public class Grade
    {
        public Guid StudentId { get; set; }
        public string Value { get; set; }

        public static IEnumerable<Grade> GetData()
        {
            for (int i = 0; i < 10000; i++)
            {
                yield return new Grade
                                {
                                    StudentId = GuidHelper.ListOfIds[i], Value = "Value " + i
                                };
            }
        }
    }

    public class Student
    {
        public Guid Id { get; set; }
        public string Name { get; set; }
        public string Grade { get; set; }

        public static IEnumerable<Student> GetStudents()
        {
            for (int i = 0; i < 10000; i++)
            {
                yield return new Student
                                {
                                    Id = GuidHelper.ListOfIds[i],
                                    Name = "Name " + i
                                };
            }
        }
    }
```

از کلاس GuidHelper برای تولید و نگهداری شناسه‌های یکتا برای دانش آموز کمک گرفته شده است :

```
public class GuidHelper
    {
        public static List<Guid> ListOfIds=new List<Guid>();

        static GuidHelper()
        {
            for (int i = 0; i < 10000; i++)
            {
                ListOfIds.Add(Guid.NewGuid());
            }
        }
    }
```

سپس لیستی از دانش آموزان و نمرات را درون حافظه ایجاد کرده و با یک حلقه نمره‌ی هر دانش آموز به Property مورد نظر مقدار داده می‌شود.

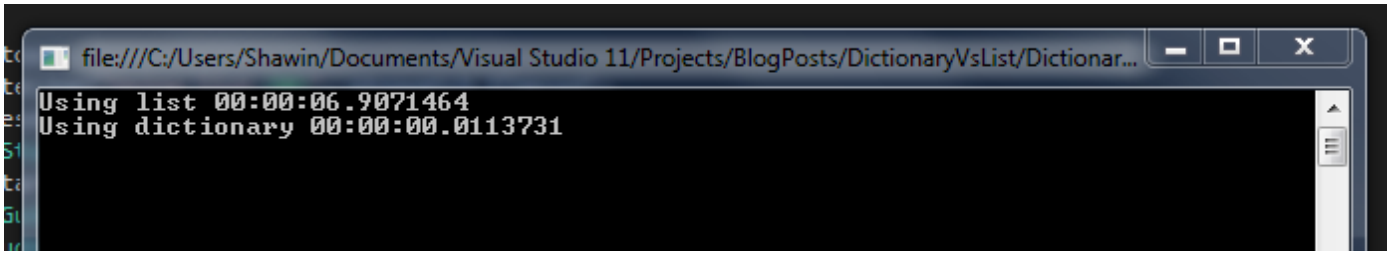
ابتدا از LINQ روی لیست برای پیدا کردن نمره‌ی مورد نظر استفاده کرده و در روش دوم برای پیدا کردن نمره‌ی هر دانش آموز از Dictionary استفاده شده :

```
internal class Program
    {
        private static void Main(string[] args)
        {
            var stopwatch = new Stopwatch();
            List<Grade> grades = Grade.GetData().ToList();
            List<Student> students = Student.GetStudents().ToList();

            stopwatch.Start();
            foreach (Student student in students)
            {
                student.Grade = grades.Single(x => x.StudentId == student.Id).Value;
            }
            stopwatch.Stop();
            Console.WriteLine("Using list {0}", stopwatch.Elapsed);
            stopwatch.Reset();
            students = Student.GetStudents().ToList();
            stopwatch.Start();
            Dictionary<Guid, string> dictionary = Grade.GetData().ToDictionary(x => x.StudentId, x =>
x.Value);

            foreach (Student student in students)
            {
                student.Grade = dictionary[student.Id];
            }
            stopwatch.Stop();
            Console.WriteLine("Using dictionary {0}", stopwatch.Elapsed);
            Console.ReadKey();
        }
    }
```

نتیجه‌ی مقایسه در سیستم من اینگونه می‌باشد :



همانگونه که مشاهده می‌شود در این سناریو خواندن نمره از روی Dictionary بر اساس 'کلید' بسیار سریع‌تر از انجام یک پرس و جوی LINQ روی لیست است.

زمانی که از LINQ on list

```
student.Grade = grades.Single(x => x.StudentId == student.Id).Value;
```

برای پیدا کردن مقدار مورد نظر یک به یک روی اعضا لیست حرکت می‌کند تا به مقدار مورد نظر برسد در نتیجه پیچیدگی زمانی آن O n هست. پس هر چه میزان داده‌ها بیشتر باشد این روش کندتر می‌شود.

زمانی که از Dictonary

```
student.Grade = dictionary[student.Id];
```

برای پیدا کردن مقدار استفاده می‌شود با اولین تلاش مقدار مورد نظر یافت می‌شود پس پیچیدگی زمانی آن 1 O می‌باشد.

در نتیجه اگر نیاز به پیدا کردن اطلاعات بر اساس یک مقدار یکتا یا کلید باشد تبدیل اطلاعات به Dictionary و خواندن از آن بسیار به صرفه‌تر است.

تفاوت این 2 روش وقتی مشخص می‌شود که میزان داده‌ها زیاد باشد.

در همین رابطه (1 ، 2)

DictionaryVsList.zip

نظرات خوانندگان

نویسنده: حسین مرادی نیا
تاریخ: ۱۳۹۲/۰۳/۱۷ ۲۱:۳۵

یه نگاهی هم به این بندازید. جالبه: http://stackoverflow.com/questions/1009107/what-net-collection-provides-the-fastest-search

نویسنده: مهدی فرزاد
تاریخ: ۱۳۹۲/۰۳/۱۸ ۰:۲

با تشکر از دوست خوبم ، یک سئوال مطرح میشه شما این نتیجه رو از روی داده‌های موجود در حافظه انجام دادید ، اگر این داده‌ها در دیتا بیس باشه و با استفاده از یک ORM مثل EF به داده‌ها دسترسی داشته باشیم برای استفاده از Dictionary ابتدا تمام داده‌ها یک بار واکشی شده و در نتیجه جستجو میشه؟ آیا این مطلب درسته؟ اگر آره پس نتیجه به نفع Linq تغییر میکنه

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۳/۱۸ ۰:۲۴

نه. ToList یا ToDictionary اصطلاحا یک نوع Projection هستند و پس از دریافت اطلاعات مطابق کوئری لینک شما اعمال خواهند شد (شکل دادن به اطلاعات دریافت شده از بانک اطلاعاتی؛ فرضا 100 رکورد دریافت شده، حالا شما خواستید از این رکوردها برای استفاده، List درست کنید یا دیکشنری یا حالت‌های دیگر).