عنوان: **بررسی سرعت و کارآیی AutoMapper**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۲/۱۲ ۲۲:۱۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, Entity framework, AutoMapper

AutoMapper تنها کتابخانه‌ی نگاشت اشیاء مخصوص دات نت نیست. در این مطلب قصد داریم سرعت AutoMapper را با حالت نگاشت دستی، نگاشت توسط EmitMapper و نگاشت به کمک ValueInjecter ، مقایسه کنیم.

مدل مورد استفاده

در اینجا قصد داریم، شیء User را یک میلیون بار توسط روش‌های مختلف، به خودش نگاشت کنیم و سرعت انجام این‌کار را در حالت‌های مختلف اندازه گیری نمائیم:

```
public class User
{
    public int Id { get; set; }
    public string UserName { get; set; }
    public string Password { get; set; }
    public DateTime LastLogin { get; set; }
}
```

روش بررسی سرعت انجام هر روش

برای کاهش کدهای تکراری، می‌توان قسمت تکرار شونده را به صورت یک Action، در بین سایر کدهایی که هر بار نیاز است به یک شکل فراخوانی شوند، قرار داد:

```
public static void RunActionMeasurePerformance(Action action)
{
    GC.Collect();
    var initMemUsage = Process.GetCurrentProcess().WorkingSet64;
    var stopwatch = new Stopwatch();
    stopwatch.Start();
    action();
    stopwatch.Stop();
    var currentMemUsage = Process.GetCurrentProcess().WorkingSet64;
    var memUsage = currentMemUsage - initMemUsage;
    if (memUsage < 0) memUsage = 0;
    Console.WriteLine("Elapsed time: {0}, Memory Usage: {1:N2} KB", stopwatch.Elapsed, memUsage /
1024);
}
```

انجام آزمایش

در مثال زیر، ابتدا یک میلیون شیء User ایجاد می‌شوند و سپس هربار توسط روش‌های مختلفی به شیء User دیگری نگاشت می‌شوند:

```
static void Main(string[] args)
{
    var length = 1000000;
    var users = new List<User>(length);
    for (var i = 0; i < length; i++)
    {

        var user = new User
        {
            Id = i,
            UserName = "User" + i,
            Password = "1" + i + "2" + i,
            LastLogin = DateTime.Now
```

```
        };
        users.Add(user);
    }

    Console.WriteLine("Custom mapping");
    RunActionMeasurePerformance(() =>
    {
        var userList =
            users.Select(
                o =>
                    new User
                    {
                        Id = o.Id,
                        UserName = o.UserName,
                        Password = o.Password,
                        LastLogin = o.LastLogin
                    }).ToList();
    });

    Console.WriteLine("EmitMapper mapping");
    RunActionMeasurePerformance(() =>
    {
        var map = EmitMapper.ObjectMapperManager.DefaultInstance.GetMapper<User, User>();
        var emitUsers = users.Select(o => map.Map(o)).ToList();
    });

    Console.WriteLine("ValueInjecter mapping");
    RunActionMeasurePerformance(() =>
    {
        var valueUsers = users.Select(o => (User)new User().InjectFrom(o)).ToList();
    });

    Console.WriteLine("AutoMapper mapping, DynamicMap using List");
    RunActionMeasurePerformance(() =>
    {
        var userMap = Mapper.DynamicMap<List<User>>(users).ToList();
    });

    Console.WriteLine("AutoMapper mapping, Map using List");
    RunActionMeasurePerformance(() =>
    {
        var userMap = Mapper.Map<List<User>>(users).ToList();
    });

    Console.WriteLine("AutoMapper mapping, Map using IEnumerable");
    RunActionMeasurePerformance(() =>
    {
        var userMap = Mapper.Map<IEnumerable<User>>(users).ToList();
    });


    Console.ReadKey();
}
```

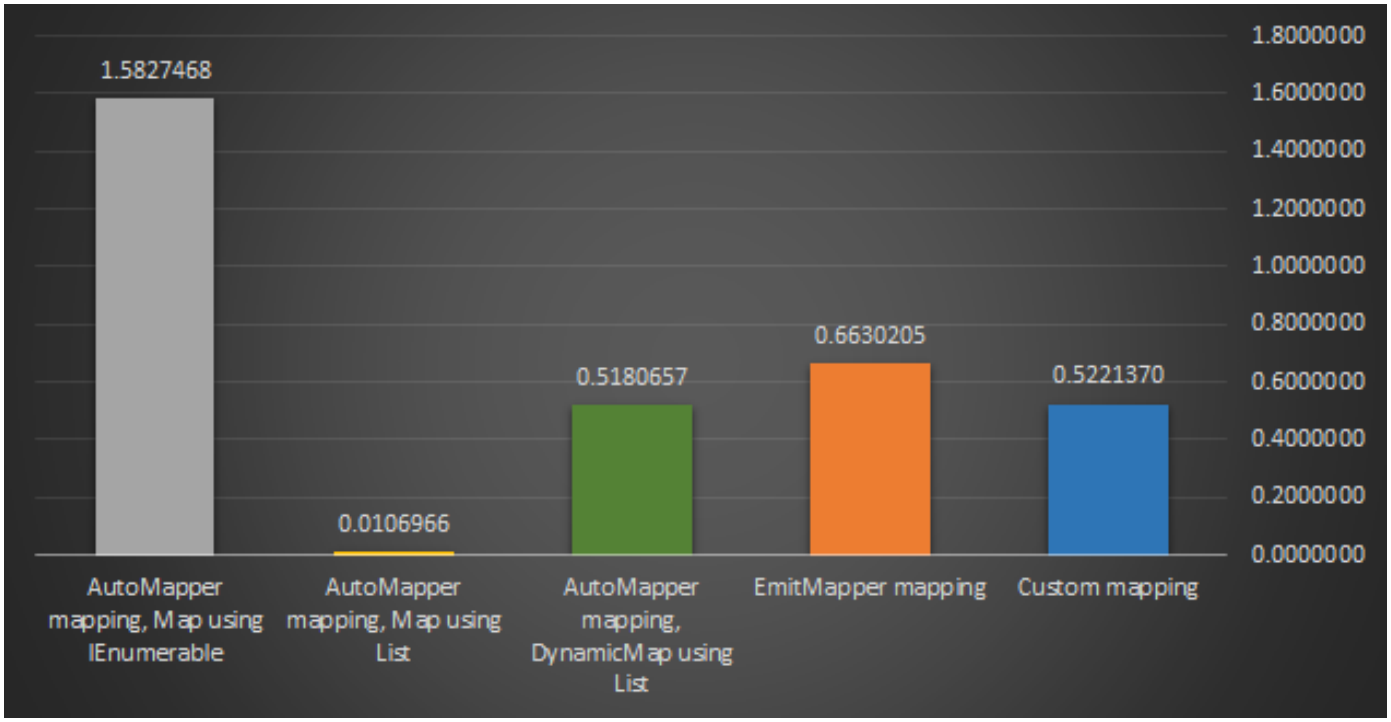
خروجی آزمایش

در ادامه یک نمونه‌ی خروجی نهایی را مشاهده می‌کنید:

```
Custom mapping
Elapsed time: 00:00:00.4869463, Memory Usage: 58,848.00 KB

EmitMapper mapping
Elapsed time: 00:00:00.6068193, Memory Usage: 62,784.00 KB

ValueInjecter mapping
Elapsed time: 00:00:15.6935578, Memory Usage: 21,140.00 KB

AutoMapper mapping, DynamicMap using List
Elapsed time: 00:00:00.6028971, Memory Usage: 7,164.00 KB

AutoMapper mapping, Map using List
Elapsed time: 00:00:00.0106244, Memory Usage: 680.00 KB

AutoMapper mapping, Map using IEnumerable
Elapsed time: 00:00:01.5954456, Memory Usage: 40,248.00 KB
```

ValueInjecter از همه کندتر است.
EmitMapper از AutoMapper سریعتر است (البته فقط در بعضی از حالت‌ها).
سرعت AutoMapper زمانیکه نوع آرگومان ورودی به آن به IEnumerable تنظیم شود، نسبت به حالت استفاده از List معمولی، به مقدار قابل توجهی کندتر است. زمانیکه از List استفاده شده، سرعت آن از سرعت حالت نگاشت دستی (مورد اول) هم بیشتر است.
متد DynamicMap اندکی کندتر است از متد Map.

در این بین اگر ValueInjecter را از لیست حذف کنیم، به نمودار ذیل خواهیم رسید (اعداد آن برحسب ثانیه هستند):



البته حین انتخاب یک کتابخانه، باید به آخرین تاریخ به روز شدن آن نیز دقت داشت و همچنین میزان استقبال جامعه‌ی برنامه نویس‌ها و از این لحاظ، AutoMapper نسبت به سایر کتابخانه‌های مشابه در صدر قرار می‌گیرد.

کدهای کامل این قسمت را از اینجا می‌توانید دریافت کنید:

AM_Sample06.zip