عنوان: **وهله سازی یک کلاس موجود توسط Reflection.Emit**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۵/۱۵ ۲۱:۳۵
آدرس: www.dotnettips.info
برچسب‌ها: C#, CIL, CLR, IL, MSIL, Reflection

در قسمت‌های قبل، نحوه ایجاد یک Type کاملا جدید را که در برنامه وجود خارجی ندارد، توسط Reflection.Emit بررسی کردیم. اکنون حالتی را در نظر بگیرید که کلاس مدنظر پیشتر در کدهای برنامه تعریف شده است، اما می‌خواهیم در یک DynamicMethod آن‌را وهله سازی کرده و حاصل را استفاده نمائیم.
کدهای کامل مثالی را در این زمینه در ادامه ملاحظه می‌کنید:

```
using System;
using System.Reflection.Emit;

namespace FastReflectionTests
{
    public class Order
    {
        public string Name { set; get; }
        public Order()
        {
            Name = "Order01";
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            var myMethod = new DynamicMethod(name: "myMethod",
                                             returnType: typeof(Order),
                                             parameterTypes: Type.EmptyTypes,
                                             m: typeof(Program).Module);
            var il = myMethod.GetILGenerator();
            il.Emit(OpCodes.Newobj, typeof(Order).GetConstructor(Type.EmptyTypes));
            il.Emit(OpCodes.Ret);

            var getOrderMethod = (Func<Order>)myMethod.CreateDelegate(typeof(Func<Order>));

            Console.WriteLine(getOrderMethod().Name);
        }
    }
}
```

کار با ایجاد یک DynamicMethod شروع می‌شود. خروجی آن از نوع کلاس Order تعریف شده، پارامتری را نیز قبول نمی‌کند و برای تعریف آن از Type.EmptyTypes استفاده شده است.
سپس با دسترسی به ILGenerator سعی خواهیم کرد تا وهله جدیدی را از کلاس Order ایجاد کنیم. برای این منظور باید از OpCode جدیدی به نام Newobj استفاده کنیم که مخفف new object است. این OpCode برای عملکرد خود، نیاز به دریافت اشاره‌گری به سازنده کلاسی دارد که قرار است آن‌را وهله سازی کند. در اینجا با Ret، کار متد را خاتمه داده و در ادامه برای استفاده از آن تنها کافی است یک delegate را ایجاد نمائیم.

بنابراین به مجموعه متدهای سریع خود، متد ذیل را نیز می‌توان افزود:

```
public static Func<T> CreatFastObjectInstantiater<T>()
        {
            var t = typeof(T);
            var ctor = t.GetConstructor(Type.EmptyTypes);

            if (ctor == null)
                return null;

            var dynamicCtor = new DynamicMethod("_", t, Type.EmptyTypes, t, true);
            var il = dynamicCtor.GetILGenerator();
            il.Emit(OpCodes.Newobj, ctor);
            il.Emit(OpCodes.Ret);

            return (Func<T>)dynamicCtor.CreateDelegate(typeof(Func<T>));
        }
```

این نوع متدها که delegate بر می‌گردانند، باید یکبار در ابتدای برنامه ایجاد شده و نتیجه آن‌ها کش شوند. پس از آن به وهله سازی بسیار سریع دسترسی خواهیم داشت.

اگر علاقمند بودید که سرعت این روش را با روش متداول Activator.CreateInstance مقایسه کنید، مطلب زیر بسیار مفید است:

Creating objects - Perf implications

یک کاربرد مهم این مساله در نوشتن ORM مانندهایی است که قرار است لیستی جنریک را خیلی سریع تولید کنند؛ از این جهت که در حلقه DataReader آن‌ها مدام نیاز است یک وهله جدید از شیء مدنظر ایجاد و مقدار دهی شود:

Mapping Datareader to Objects Using Reflection.Emit