عنوان: **ایجاد «خواص الحاقی»**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۵/۱۷ ۱۱:۳۵
آدرس: www.dotnettips.info
گروه‌ها: #C

حتما با متدهای الحاقی یا Extension methods آشنایی دارید؛ می‌توان به یک شیء، که حتی منبع آن در دسترس ما نیست، متدی را اضافه کرد. سؤال: در مورد خواص چطور؟ آیا می‌شود به وهله‌ای از یک شیء موجود از پیش طراحی شده، یک خاصیت جدید را اضافه کرد؟
احتمالا شاید عنوان کنید که با اشیاء dynamic می‌توان چنین کاری را انجام داد. اما سؤال در مورد اشیاء غیر dynamic است.
یا نمونه‌ی دیگر آن Attached Properties در برنامه‌های مبتنی بر Xaml هستند. می‌توان به یک شیء از پیش موجود Xaml، خاصیتی را افزود که البته پیاده سازی آن منحصر است به همان نوع برنامه‌ها.

راه حل پیشنهادی

یک Dictionary را ایجاد کنیم تا ارجاعی از اشیاء، به عنوان کلید، در آن ذخیره شده و سپس key/valueهایی به عنوان value هر شیء، در آن ذخیره شوند. این key/valueها همان خواص و مقادیر آن‌ها خواهند بود. هر چند این راه حل به خوبی کار می‌کند اما ... مشکل نشتی حافظه دارد.
شیء Dictionary یک ارجاع قوی را از اشیاء، درون خودش نگه داری می‌کند و تا زمانیکه در حافظه باقی است، سیستم GC مجوز رهاسازی منابع آن‌ها را نخواهد یافت؛ چون عموما این نوع Dictionaryها باید استاتیک تعریف شوند تا طول عمر آن‌ها با طول عمر برنامه یکی گردد. بنابراین اساسا اشیایی که به این نحو قرار است پردازش شوند، هیچگاه dispose نخواهند شد. راه حلی برای این مساله در دات نت 4 به صورت توکار به دات نت فریم ورک اضافه شده‌است؛ به نام ساختار داده‌ای ConditionalWeakTable.

معرفی ConditionalWeakTable

ConditionalWeakTable جزو ساختارهای داده‌ای کمتر شناخته شده‌ی دات نت است. این ساختار داده، اشاره‌گرهایی را به ارجاعات اشیاء، درون خود ذخیره می‌کند. بنابراین چون ارجاعاتی قوی را به اشیاء ایجاد نمی‌کند، مانع عملکرد GC نیز نشده و برنامه در دراز مدت دچار مشکل نشتی حافظه نخواهد شد. هدف اصلی آن ایجاد ارتباطی بین CLR و DLR است. توسط آن می‌توان به اشیاء دلخواه، خواصی را افزود. به علاوه طراحی آن به نحوی است که thread safe است و مباحث قفل گذاری بر روی اطلاعات، به صورت توکار در آن پیاده سازی شده‌است. کار DLR فراهم آوردن امکان پیاده سازی زبان‌های پویایی مانند Ruby و Python برفراز CLR است. در این نوع زبان‌ها می‌توان به وهله‌هایی از اشیاء موجود، خاصیت‌های جدیدی را متصل کرد.
به صورت خلاصه کار ConditionalWeakTable ایجاد نگاشتی است بین وهله‌هایی از اشیاء CLR (اشیایی غیرپویا) و خواصی که به آن‌ها می‌توان به صورت پویا انتساب داد. در کار GC اخلال ایجاد نمی‌کند و همچنین می‌توان به صورت همزمان از طریق تردهای مختلف، بدون مشکل با آن کار کرد.

پیاده سازی خواص الحاقی به کمک ConditionalWeakTable

در اینجا نحوه‌ی استفاده از ConditionalWeakTable را جهت اتصال خواصی جدید به وهله‌های موجود اشیاء مشاهده می‌کنید:

```
using System.Collections.Generic;
using System.Runtime.CompilerServices;

namespace ConditionalWeakTableSamples
{
    public static class AttachedProperties
    {
        public static ConditionalWeakTable<object,
            Dictionary<string, object>> ObjectCache = new ConditionalWeakTable<object,
                Dictionary<string, object>>();

        public static void SetValue<T>(this T obj, string name, object value) where T : class
        {
            var properties = ObjectCache.GetOrCreateValue(obj);
```

```
            if (properties.ContainsKey(name))
                properties[name] = value;
            else
                properties.Add(name, value);
        }

        public static T GetValue<T>(this object obj, string name)
        {
            Dictionary<string, object> properties;
            if (ObjectCache.TryGetValue(obj, out properties) && properties.ContainsKey(name))
                return (T)properties[name];
            return default(T);
        }

        public static object GetValue(this object obj, string name)
        {
            return obj.GetValue<object>(name);
        }
    }
}
```

ObjectCache تعریف شده از نوع استاتیک است؛ بنابراین در طول عمر برنامه زنده نگه داشته خواهد شد، اما اشیایی که به آن منتسب می‌شوند، خیر. هرچند به ظاهر در متد GetOrCreateValue، یک وهله از شیءایی موجود را دریافت می‌کند، اما در پشت صحنه صرفا IntPtr یا اشاره‌گری به این شیء را ذخیره سازی خواهد کرد. به این ترتیب در کار GC اخلالی صورت نخواهد گرفت و شیء مورد نظر، تا پایان کار برنامه به اجبار زنده نگه داشته نخواهد شد.

کاربرد اول

اگر با ASP.NET کار کرده باشید حتما با IPrincipal آشنایی دارید. خواصی مانند Identity یک کاربر در آن ذخیره می‌شوند.
سؤال: چگونه می‌توان یک خاصیت جدید به نام مثلا Disclaimer را به وهله‌ای از این شیء افزود:

```
public static class ISecurityPrincipalExtension
    {
        public static bool Disclaimer(this IPrincipal principal)
        {
            return principal.GetValue<bool>("Disclaimer");
        }

        public static void SetDisclaimer(this IPrincipal principal, bool value)
        {
            principal.SetValue("Disclaimer", value);
        }
    }
```

در اینجا مثالی را از کاربرد کلاس AttachedProperties فوق مشاهده می‌کنید. توسط متد SetDisclaimer یک خاصیت جدید به نام Disclaimer به وهله‌ای از شیءایی از نوع IPrincipal قابل اتصال است. سپس توسط متد Disclaimer قابل دستیابی خواهد بود.

اگر صرفا قرار است یک خاصیت به شیءایی متصل شود، روش ذیل نیز قابل استفاده می‌باشد (بجای استفاده از دیکشنری از یک کلاس جهت تعریف خاصیت اضافی جدید استفاده شده‌است):

```
using System.Runtime.CompilerServices;

namespace ConditionalWeakTableSamples
{
    public static class PropertyExtensions
    {
        private class ExtraPropertyHolder
        {
            public bool IsDirty { get; set; }
        }

        private static readonly ConditionalWeakTable<object, ExtraPropertyHolder> _isDirtyTable
                = new ConditionalWeakTable<object, ExtraPropertyHolder>();
```

```
        public static bool IsDirty(this object @this)
        {
            return _isDirtyTable.GetOrCreateValue(@this).IsDirty;
        }

        public static void SetIsDirty(this object @this, bool isDirty)
        {
            _isDirtyTable.GetOrCreateValue(@this).IsDirty = isDirty;
        }
    }
}
```

کاربرد دوم

ایجاد Id منحصربفرد برای اشیاء برنامه.
فرض کنید در حال نوشتن یک Entity framework profiler هستید. طراحی فعلی سیستم Interception آن به نحو زیر است:

```
public void Closed(DbConnection connection, DbConnectionInterceptionContext interceptionContext)
{
}
```

سؤال: اینجا رویداد بسته شدن یک اتصال را دریافت می‌کنیم؛ اما ... دقیقا کدام اتصال؟ رویداد Opened را هم داریم اما چگونه این اشیاء را به هم مرتبط کنیم؟ شیء DbConnection دارای Id نیست. متد GetHashCode هم الزامی ندارد که اصلا پیاده سازی شده باشد یا حتی یک Id منحصربفرد را تولید کند. این متد با تغییر مقادیر خواص یک شیء می‌تواند مقادیر متفاوتی را ارائه دهد. در اینجا می‌خواهیم به ازای ارجاعی از یک شیء، یک Id منحصربفرد داشته باشیم تا بتوانیم تشخیص دهیم که این اتصال بسته شده، دقیقا کدام اتصال باز شده‌است؟
راه حل: خوب ... یک خاصیت Id را به اشیاء موجود متصل کنید!

```
using System;
using System.Runtime.CompilerServices;

namespace ConditionalWeakTableSamples
{
    public static class UniqueIdExtensions
    {
        static readonly ConditionalWeakTable<object, string> _idTable =
                                    new ConditionalWeakTable<object, string>();

        public static string GetUniqueId(this object obj)
        {
            return _idTable.GetValue(obj, o => Guid.NewGuid().ToString());
        }

        public static string GetUniqueId(this object obj, string key)
        {
            return _idTable.GetValue(obj, o => key);
        }
    }
}
```

در اینجا مثالی دیگر از پیاده سازی و استفاده از ConditionalWeakTable را ملاحظه می‌کنید. اگر در کش آن ارجاعی به شیء مورد نظر وجود داشته باشد، مقدار Guid آن بازگشت داده می‌شود؛ اگر خیر، یک Guid به ارجاعی از شیء، انتساب داده شده و سپس بازگشت داده می‌شود. به عبارتی به صورت پویا یک خاصیت UniqueId به وهله‌هایی از اشیاء اضافه می‌شوند. به این ترتیب به سادگی می‌توان آن‌ها را ردیابی کرد و تشخیص داد که اگر این Guid پیشتر جایی به اتصال باز شده‌ای منتسب شده‌است، در چه زمانی و در کجا بسته شده است یا اصلا ... خیر. جایی بسته نشده‌است.

برای مطالعه بیشتر

The Conditional Weak Table: Enabling Dynamic Object Properties

How to create mixin using C# 4.0

Disclaimer Page using MVC

Extension Properties Revised

Easy Modeling

Providing unique ID on managed object using ConditionalWeakTable