عنوان: **دسترسی به Collectionها در یک ترد دیگر در WPF**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۰۷ ۱:۴۵
آدرس: www.dotnettips.info
گروه‌ها: WPF, Threading

اگر در WPF سعی کنیم آیتمی را به مجموعه اعضای یک Collection مانند یک List یا ObservableCollection از طریق تردی دیگر اضافه کنیم، با خطای ذیل متوقف خواهیم شد:

```
This type of CollectionView does not support changes to its SourceCollection
from a thread different from the Dispatcher thread
```

راه حلی که برای آن تا دات نت 4 در اکثر سایت‌ها توصیه می‌شد به نحو ذیل است:

Adding to an ObservableCollection from a background thread

مشکل!

اگر همین برنامه را که برای دات نت 4 کامپایل شده‌است، بر روی سیستمی که دات نت 4.5 بر روی آن نصب است اجرا کنیم، برنامه با خطای ذیل متوقف می‌شود:

```
System.InvalidOperationException: This exception was thrown because the generator for control
'System.Windows.Controls.ListView Items.Count:62' with name '(unnamed)' has received sequence of
CollectionChanged events that do not agree with the current state of the Items collection.
The following differences were detected:  Accumulated count 61 is different from actual count 62.
```

مشکل از کجاست؟

در دات نت 4 و نیم، دیگر نیازی به استفاده از کلاس MTObservableCollection یاد شده نیست و به صورت توکار امکان کار با Collectionها از طریق تردی دیگر میسر است. فقط برای فعال سازی آن باید نوشت:

```
private static object _lock = new object();
//...
BindingOperations.EnableCollectionSynchronization(persons, _lock);
```

پس از اینکه برای نمونه، مجموعه‌ی فرضی persons وهله سازی شد، تنها کافی است متد جدید EnableCollectionSynchronization بر روی آن فراخوانی شود.

برای برنامه‌ی دات نت 4 ایی که قرار است در سیستم‌های مختلف اجرا شود چطور؟

در اینجا باید از Reflection کمک گرفت. اگر متد EnableCollectionSynchronization بر روی کلاس BindingOperations یافت شد، یعنی برنامه‌ی دات نت 4، در محیط جدید در حال اجرا است:

```
public static void EnableCollectionSynchronization(IEnumerable collection, object lockObject)
{
      MethodInfo method = typeof(BindingOperations).GetMethod("EnableCollectionSynchronization",
            new Type[] { typeof(IEnumerable), typeof(object) });
      if (method != null)
      {
         method.Invoke(null, new object[] { collection, lockObject });
      }
}
```

در این حالت فقط کافی است این متد جدید یافت شده را بر روی Collection مدنظر فراخوانی کنیم.
همچنین اگر بخواهیم کلاس MTObservableCollection معرفی شده را جهت سازگاری با دات نت 4 و نیم به روز کنیم، به کلاس ذیل خواهیم رسید. این کلاس با دات نت 4 و 4.5 سازگار است و جهت کار با ObservableCollectionها از طریق تردهای مختلف

تهیه شده‌است:

```
using System;
using System.Collections;
using System.Collections.ObjectModel;
using System.Collections.Specialized;
using System.Linq;
using System.Windows.Data;
using System.Windows.Threading;

namespace WpfAsyncCollection
{
    public class AsyncObservableCollection<T> : ObservableCollection<T>
    {
        public override event NotifyCollectionChangedEventHandler CollectionChanged;
        private static object _syncLock = new object();

        public AsyncObservableCollection()
        {
            enableCollectionSynchronization(this, _syncLock);
        }

        protected override void OnCollectionChanged(NotifyCollectionChangedEventArgs e)
        {
            using (BlockReentrancy())
            {
                var eh = CollectionChanged;
                if (eh == null) return;

                var dispatcher = (from NotifyCollectionChangedEventHandler nh in eh.GetInvocationList()
                                  let dpo = nh.Target as DispatcherObject
                                  where dpo != null
                                  select dpo.Dispatcher).FirstOrDefault();

                if (dispatcher != null && dispatcher.CheckAccess() == false)
                {
                    dispatcher.Invoke(DispatcherPriority.DataBind, (Action)(() =>
OnCollectionChanged(e)));
                }
                else
                {
                    foreach (NotifyCollectionChangedEventHandler nh in eh.GetInvocationList())
                        nh.Invoke(this, e);
                }
            }
        }

        private static void enableCollectionSynchronization(IEnumerable collection, object lockObject)
        {
            var method = typeof(BindingOperations).GetMethod("EnableCollectionSynchronization",
                                    new Type[] { typeof(IEnumerable), typeof(object) });
            if (method != null)
            {
                // It's .NET 4.5
                method.Invoke(null, new object[] { collection, lockObject });
            }
        }
    }
}
```

در این کلاس، در سازنده‌ی آن متد عمومی enableCollectionSynchronization فراخوانی می‌شود. اگر برنامه در محیط دات نت 4 فراخوانی شود، تاثیری نخواهد داشت چون method در حال بررسی نال است. در غیراینصورت، برنامه در حالت سازگار با دات نت 4.5 اجرا خواهد شد.

نظرات خوانندگان

نویسنده: جلال
تاریخ: ۱۳۹۴/۰۱/۲۵ ۱۲:۴

چرا _syncLock به صورت static تعریف شده؟
به نظرم static بودن نه تنها غیرضروریه، بلکه حتی در مواردی (هر چند نادر) که چند نمونه‌ی جداگانه از Persons در بخش‌های مختلف برنامه همزمان در حال کار باشند، باید همه شون منتظر آزاد شدن شیء syncLock برای انجام عملیات خودشون بشن که در حقیقت نیازی نیست. Why does the lock object have to be static?

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۵ ۱۳:۰

lock تعریف شده در اینجا، صرفا جهت کار با UI Dispatcher وجود دارد و هر برنامه یک UI Dispatcher بیشتر برای به روز رسانی UI ندارد (Application.Current.Dispatcher) و EnableCollectionSynchronization دات نت 4.5، شبیه به کار متد OnCollectionChanged نوشته شده جهت دات نت 4 را انجام می‌دهد. در متد OnCollectionChanged هم یک lock وجود دارد تا هربار از هر تردی، فقط یک مورد جهت ارسال به dispatcher برنامه، قابلیت ورود پیدا کند. هدف اصلی این است که اطلاعات دریافتی از تردهای مختلف، در ترد UI نمایش داده شوند یا به روز شوند و ترد UI هربار فقط یک آیتم را قبول می‌کند، آن هم نه از طریق تردهای دیگر. به همین جهت، این تردها باید صبر کنند تا عملیات قبلی UI خاتمه یابد.