```
using Common;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Collections.Specialized;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows;

namespace ControllerLib.Business.Convert
{
    public class AsyncObservableCollection<T> : Collection<T>, INotifyCollectionChanged
    {
        public event NotifyCollectionChangedEventHandler CollectionChanged;

        public object LockRefresh = new object();

        public void AddRange(IEnumerable<T> collection)
        {
            try
            {
                lock (LockRefresh)
                {

                    Application.Current.Dispatcher.BeginInvoke(new Action(() =>
                    {
                        try
                        {
                            lock (LockRefresh)
                            {
                                base.ClearItems();
                                int index = 0;
                                foreach (T item in collection)
                                {
                                    //base.Add(item);
                                    index = base.Count;
                                    base.InsertItem(index, item);
                                }
                                NotifyCollectionChangedEventArgs
notifyCollectionChangedEventArgs = new
NotifyCollectionChangedEventArgs(NotifyCollectionChangedAction.Reset);
                                CollectionChanged?.Invoke(this,
notifyCollectionChangedEventArgs);
                            }
                        }
                        catch (Exception ex)
                        {
                            Log.GetIns().WriteErrorLog(ex, Log.LogType.Error);
                        }
```

```
                }));
            }
        }
        catch (Exception ex)
        {
            Log.GetIns().WriteErrorLog(ex, Log.LogType.Error);
        }
    }

    public void AddRangeNoRefresh(IEnumerable<T> collection)
    {
        try
        {
            lock (LockRefresh)
            {
                base.ClearItems();
                int index = 0;
                foreach (T item in collection)
                {
                    index = base.Count;
                    base.InsertItem(index, item);
                }
            }
        }
        catch (Exception ex)
        {
            Log.GetIns().WriteErrorLog(ex, Log.LogType.Error);
        }
    }
    protected override void InsertItem(int index, T item)
    {
        try
        {
            lock (LockRefresh)
            {
                base.InsertItem(index, item);

                Application.Current.Dispatcher.InvokeAsync(() =>
                {
                    try
                    {
                        lock (LockRefresh)
                        {
                            CollectionChanged?.Invoke(this, new
NotifyCollectionChangedEventArgs(NotifyCollectionChangedAction.Add, item, index));
                        }
                    }
                    catch (Exception ex)
                    {

                    }
                });
```

```
                }
            }
            catch (Exception ex)
            {
                Log.GetIns().WriteErrorLog(ex, Log.LogType.Error);
            }
        }

        protected override void ClearItems()
        {
            try
            {
                lock (LockRefresh)
                {
                    base.ClearItems();

                    Application.Current.Dispatcher.InvokeAsync(() =>
                    {
                        try
                        {
                            lock (LockRefresh)
                            {
                                CollectionChanged?.Invoke(this, new
NotifyCollectionChangedEventArgs(NotifyCollectionChangedAction.Reset));
                            }
                        }
                        catch (Exception ex)
                        {

                        }
                    });

                }
            }
            catch (Exception ex)
            {
                Log.GetIns().WriteErrorLog(ex, Log.LogType.Error);
            }
        }

        protected override void RemoveItem(int index)
        {
            try
            {
                lock (LockRefresh)
                {
                    var item = this[index];

                    base.RemoveItem(index);

                    Application.Current.Dispatcher.InvokeAsync(() =>
                    {
```

```
                        try
                        {
                            lock (LockRefresh)
                            {
                                CollectionChanged?.Invoke(this, new
NotifyCollectionChangedEventArgs(NotifyCollectionChangedAction.Remove, item, index));
                            }
                        }
                        catch (Exception ex)
                        {

                        }
                    });
                }
            }
            catch (Exception ex)
            {
                Log.GetIns().WriteErrorLog(ex, Log.LogType.Error);
            }
        }

        protected override void SetItem(int index, T item)
        {
            try
            {
                lock (LockRefresh)
                {
                    var oldItem = this[index];

                    base.SetItem(index, item);

                    Application.Current.Dispatcher.InvokeAsync(() =>
                    {
                        try
                        {
                            lock (LockRefresh)
                            {
                                CollectionChanged?.Invoke(this, new
NotifyCollectionChangedEventArgs(NotifyCollectionChangedAction.Replace, item, oldItem,
index));
                            }
                        }
                        catch (Exception ex)
                        {

                        }
                    });
                }
            }
            catch (Exception ex)
            {
                Log.GetIns().WriteErrorLog(ex, Log.LogType.Error);
```

```
            }
        }


        public void ResetRefresh()
        {
            try
            {
                Application.Current.Dispatcher.BeginInvoke(new Action(() =>
                {
                    try
                    {
                        lock (LockRefresh)
                        {
                            NotifyCollectionChangedEventArgs
notifyCollectionChangedEventArgs = new
NotifyCollectionChangedEventArgs(NotifyCollectionChangedAction.Reset);
                            CollectionChanged?.Invoke(this,
notifyCollectionChangedEventArgs);
                        }
                    }
                    catch (Exception ex)
                    {
                        Log.GetIns().WriteErrorLog(ex, Log.LogType.Error);
                    }
                }));
            }
            catch (Exception ex)
            {
                Log.GetIns().WriteErrorLog(ex, Log.LogType.Error);
            }
        }

    }
}
```