DispatcherTimer (Counter v Admin -repricing) a System. Threading.Timer

# DispatcherTimer

Neni disposable

|  |
| --- |
| Petr Holubec:  Ani nahodou:  Timers are not guaranteed to execute exactly when the time interval occurs, but they are guaranteed to not execute before the time interval occurs. This is because DispatcherTimer operations are placed on the Dispatcher queue like other operations. When the DispatcherTimer operation executes is dependent on the other jobs in the queue and their priorities.  Tzn. nemas sanci zajistit, ze to bude trvat 10 sekund, kdyz to zavolas 10x po jedne vterine. Chapu, ze tady je to vsem burt, ale je to spatne. Navic nikde nevidim hlidani, ze se refresh nespusti znovu, i kdyz predchozi nedobehl.  Jak to ma byt spravne: Bezi timer (pokud mozno System.Threading.Timer), ktery kazdych treba 250ms spusti checkovani, zda neni potreba vyvolat refresh. Pamatuje se posledni cas dobehnuti refreshe + jestli refresh bezi. Aktualizuje se informace o case, za jak dlouho pobezi dalsi refresh. Ano, muze se to dostat i do zapornych hodnot, kdyz refresh nedobehl. Tehdy bud roztocis nejake kolecko, anebo nechas zapornou hodnotu. Kazdopadne uzivatel uvidi, ze se neco deje. Kdyz uplynulo vice jak 10s od dobehnuti posledniho refreshe, vyvola se refresh, na jehoz konci se nastavi cas refreshe. |

**DispatcherTimer:**

namespace MIR.Media.Admin.Screens.MessageRepricing

{

public class CounterControlViewModel: Screen

{

private int m\_counter;

public CounterControlViewModel()

{

Counter = 10;

new DispatcherTimer(TimeSpan.FromSeconds(1), DispatcherPriority.Normal,

delegate

{

var newValue = Counter == 0 ? 10 : Counter - 1;

Counter = newValue;

}, Dispatcher.CurrentDispatcher

);

}

public int Counter

{

get => m\_counter;

set

{

m\_counter = value;

NotifyOfPropertyChange();

}

}

}

}

<UserControl

x:Class="MIR.Media.Admin.Screens.MessageRepricing.CounterControlView"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:MIR.Media.Admin.Screens.MessageRepricing"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

x:Name="CustomCounterControl"

d:DataContext="{d:DesignInstance local:CounterControlViewModel}"

mc:Ignorable="d">

<TextBlock

VerticalAlignment="Bottom"

FontSize="15"

FontWeight="DemiBold"

Text="{Binding Counter}" />

</UserControl>

# System.Threading.Timer

Se musi disposovat ! Integer v kodu neni ThreadSafe. Viz Admin.CounterControlViewModel.

using System;

using System.Threading;

using Caliburn.Micro;

using Mediaresearch.Framework.DataAccess.Core.Auditable;

using MIR.Media.Admin.Resources;

using MIR.Media.Admin.Screens.MessageRepricing.Wrappers;

namespace MIR.Media.Admin.Screens.MessageRepricing

{

public class CounterControlViewModel : Screen, IDisposable

{

private const int RefreshInterval = 10;

private readonly IEventAggregator m\_eventAggregator;

private readonly Timer m\_refreshTimer;

private readonly ITimeProvider m\_timeProvider;

private bool m\_disposed;

private volatile bool m\_isRefreshRunning;

private DateTimeOffset m\_lastRefresh;

private string m\_message;

public CounterControlViewModel(IEventAggregator eventAggregator, ITimeProvider timeProvider)

{

m\_timeProvider = timeProvider;

m\_lastRefresh = m\_timeProvider.Now;

m\_eventAggregator = eventAggregator;

m\_eventAggregator.Subscribe(this);

m\_refreshTimer = new Timer(\_ => RefreshData(), null, TimeSpan.FromSeconds(1), TimeSpan.FromSeconds(1));

}

public string Message

{

get => m\_message;

set

{

m\_message = value;

NotifyOfPropertyChange();

}

}

private void RefreshData()

{

try

{

var now = m\_timeProvider.Now;

if (m\_isRefreshRunning)

{

Message = Localisation.RefreshIsRunning;

return;

}

if (m\_lastRefresh.AddSeconds(RefreshInterval) < now)

{

m\_isRefreshRunning = true;

m\_eventAggregator.Publish(new RefreshStatusWrapper());

m\_lastRefresh = m\_timeProvider.Now;

}

Message = string.Format(Localisation.AutomaticRefreshIn, Convert.ToInt32(RefreshInterval - (now - m\_lastRefresh).TotalSeconds));

}

finally

{

m\_isRefreshRunning = false;

}

}

public void Dispose()

{

Dispose(true);

GC.SuppressFinalize(this);

}

~CounterControlViewModel()

{

Dispose(false);

}

private void Dispose(bool disposing)

{

if (!m\_disposed)

{

if (disposing)

{

}

m\_refreshTimer?.Dispose();

}

m\_disposed = true;

}

}

}