Around and About .NET World

Marco Minerva's Blog

- Home
- My Apps
- Contacts

Type text to search here...

Home > .NET, C#, Universal Windows Platform > Pay attention to .NET Native when using Reflection (MVVM scenario)

Pay attention to .NET Native when using Reflection (MVVM scenario)

22/02/2016

Among the other things, .NET Native tries to reduce the app size by analyzing the code and striping out unused .NET objects. This approach works very well, but we need to pay attention when we you work with Reflection, because in this case we must explicitly tell to .NET Native which elements are available for it.

A particular scenario occurs if we're following the MVVM pattern and we have a classic app with a GridView or a ListView and we want to invoke a command when the user clicks an item, passing the item itself to the command. With the Universal Windows Platform, this task can be accomplished using an **EventTriggerBehavoir** form XamlBehavoirs along with an **InvokeCommandAction** and an **InputConverter**:

```
1   <GridView IsItemClickEnabled="True" ItemsSource="{Binding Pe
2       <GridView.ItemTemplate>
3           <DataTemplate>
4               ...
5           </DataTemplate>
6       </GridView.ItemTemplate>
7
8       <Interactivity:Interaction.Behaviors>
9           <Interactions:EventTriggerBehavior EventName="ItemCl
10              <Interactions:InvokeCommandAction
11                  Command="{Binding ItemSelectedCommand}"
12                  InputConverter="{StaticResource EventArgsCo
13                  InputConverterParameter="ClickedItem" />
14          </Interactions:EventTriggerBehavior>
15      </Interactivity:Interaction.Behaviors>
16  </GridView>
```

At lines 9-14 we use an EventTriggerBehavior to catch the ItemClick event of the GridView and execute in response the **ItemSelectedCommand** action of the ViewModel. Thanks to InputConverter and **InputConverterParamter** (lines 12-13), this action automatically receives the clicked item as argument:

```
public sealed class EventArgsConverter : IValueConverter
{
    public object Convert(object value, Type targetType, obj
        string language)
    {
        if (value != null)
        {
            var propertyPath = parameter as string;
            if (!string.IsNullOrWhiteSpace(propertyPath))
            {
                var propertyPathParts = propertyPath.Split('
                object propertyValue = value;
                foreach (var propertyPathPart in propertyPat
                {
                    var propInfo = propertyValue.GetType().G
                        GetDeclaredProperty(propertyPathPart

                    propertyValue = propInfo.GetValue(proper
                }

                return propertyValue;
            }
        }

        return value;
    }

    public object ConvertBack(object value, Type targetType,
29            string language)
30        {
31            throw new NotImplementedException();
32        }
33    }
```

This Converter extracts the property name of the event argument that is passed as parameter (line 8, in this case **ClickedItem**) and then uses Reflection (lines 15-18) to get the actual value that will be received by the command.

This code works without problem in Debug mode, but when we switch to Release, enabling the .NET Native toolchain (that is required for app submission), if we click an item on the grid we'll get a runtime error like the following one:

> Unhandled exception at 0x77026D7E (combase.dll)

This occurs because the .NET Native toolchain has removed the ItemClickEventArgs class from the compiled code, as it hasn't found any reference to it in the code, because it is accessed only at runtime through Reflection.

So, as said at the beginning we need to tell to .NET Native to keep this type during compilation. Let's open the file **Default.rd.xml** inside the project *Properties* folder: it contains the Runtime Directives for .NET Native. All that we need is to add a declaration for the ItemClickEventArgs type:

```
1   <Directives xmlns="http://schemas.microsoft.com/netfx/2013/
2       <Application>
3           <!--
4             An Assembly element with Name="*Application*" appl
5             assemblies in the application package. The asteris
6             wildcards.
7           -->
8   
9           <Assembly Name="*Application*" Dynamic="Required All
10  
11          <!-- Add your application specific runtime directive
12          <Type Name="Windows.UI.Xaml.Controls.ItemClickEventA
13                Dynamic="Required Public" />
14  
15      </Application>
16  </Directives>
```

At lines 12-13 we have specified that we want to keep the ItemClickEventArgs type in compilation, even if it isn't statically referenced in the code. Now we can compile the app in Release mode again: this time everything will work as expected.

We can refer to MSDN documentation to get more information about the Runtime Directives Configuration File.

Share this:

Email Print Twitter Facebook 11 LinkedIn

Like

Be the first to like this.

Related

Universal apps, User Controls and MVVM
In "C#"

Adaptive Triggers, RelativePanel and DataTemplate in the Universal Windows Platform
In "Universal Windows Platform"

NavigationService in MVVM Light V5
In "C#"

Categories: .NET, C#, Universal Windows Platform
Clear history in Xamarin Android with MVVM Light NavigationServiceUsing MVVM Light ViewModelBase class with Template 10
RSS feed

Twitter

Microsoft MVP Windows Development

MVP
Microsoft®
Most Valuable Professional

News from Twitter

- RT @MadsTorgersen: @danfma @matthewwarren Put `<NullableContextOptions>enable</NullableContextOptions>` in your csproj file. That enables i… 7 hours ago
- RT @matthewwarren: Nice example of 'dog-fooding', the .NET Core CLR is starting to use the new C# 8.0 'Nullable Reference Types' feature ht… 7 hours ago
- RT @ClintRutkas: just read a great article by @pnp0a03 on multi-instance support in UWP. Solid read! ddlgjp.blogspot.com/2019/03/uwp-ap… 7 hours ago
- RT @DotNetAppDev: Visual Studio 2019 Launch Event agenda and speakers now published devblogs.microsoft.com/visualstudio/v… https://t.co/H2fMN4P0t0 21 hours ago
- RT @DotNetAppDev: Using the Azure SignalR Service Bindings in Azure Functions to Create Real-time Serverless

Applications https://t.co/KVPE…
21 hours ago
- @betsyweber Here I am :) 21 hours ago
- Un ringraziamento a tutti quelli che sono passati al mio talk sui #CognitiveServices durante #CloudConf2019. È semp… twitter.com/i/web/status/1… 23 hours ago
- RT @MarcoDalPino: Con @marcominerva si parla di #Cognitive #Azure a #cloudconf2019 @_CloudConf_ https://t.co/AeBEEMImuQ 23 hours ago
- Tutto pronto per il talk #CloudConf2019 /w @DotNetToscana #MVPBuzz https://t.co/DY7Fd9ZuXM 1 day ago
- .@MarcoDalPino all'opera durante la #CloudConf2019 /w @DotNetToscana #mvpbuzz https://t.co/sqfOiVggwC 1 day ago

Categories

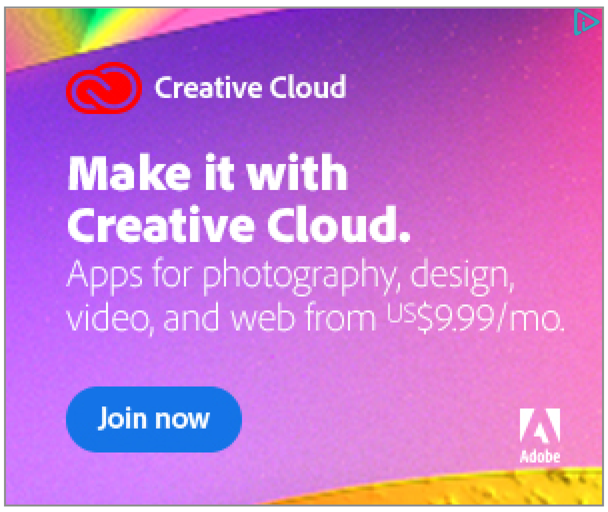
- .NET
- .NET Core
- ADO.NET
- ASP.NET MVC
- ASP.NET WebAPI
- Azure
- C#
- CodePlex
- Community
- Entity Framework
- Gadgeteer
- General
- Mobile Services
- MVVM
- Team Foundation Server
- Universal Windows Platform
- Visual Studio
- Windows Phone
- WinRT
- WPF
- Xamarin

Blogroll

- Emanuele B.'s Blog
- Integral Design
- Marco Dal Pino's Blog

Meta

- Register
- Log in
- Entries RSS
- Comments RSS
- WordPress.com


Creative Cloud
Make it with Creative Cloud.
Apps for photography, design, video, and web from US$9.99/mo.
Join now
Adobe

Top
Blog at WordPress.com.