X:Bind

Compiled databinding in UWP

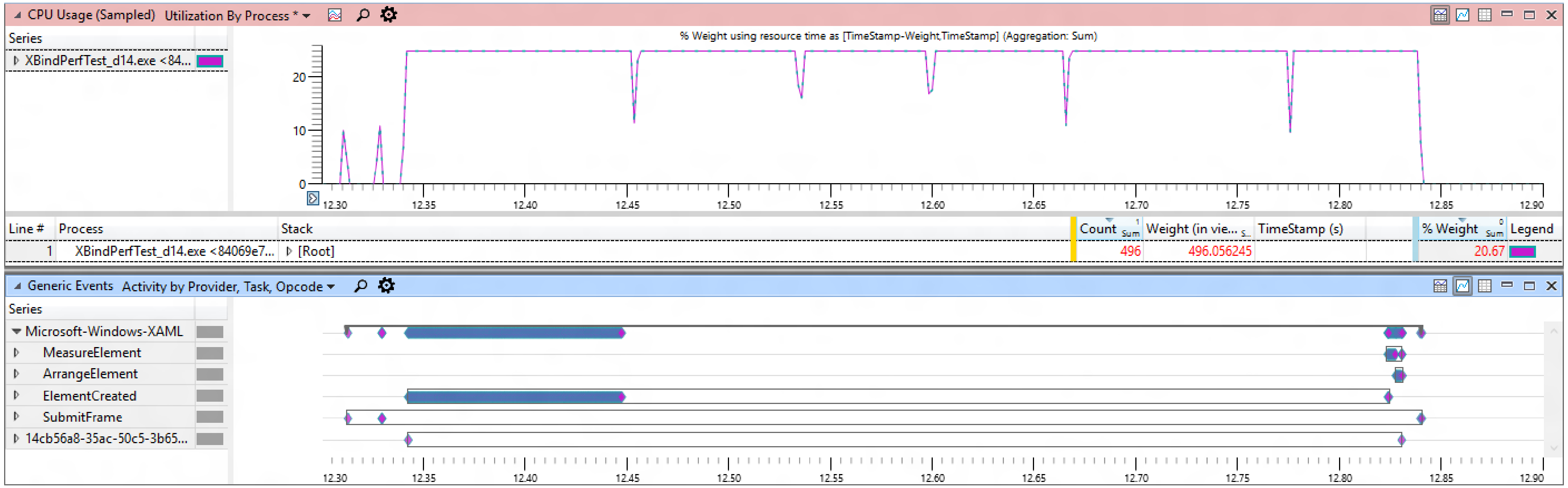
Introducing compiled binding

How do we keep the power of data binding but make it faster?

- New mechanism for data binding in Xaml Apps
- Heavy lifting is done at project build time rather than at runtime
  - Declarative bindings are converted into generated code behind
  - Eliminates need for slow runtime “reflection” operations
  - Code can be inspected and debugged
- x:Bind bindings are validated at build time

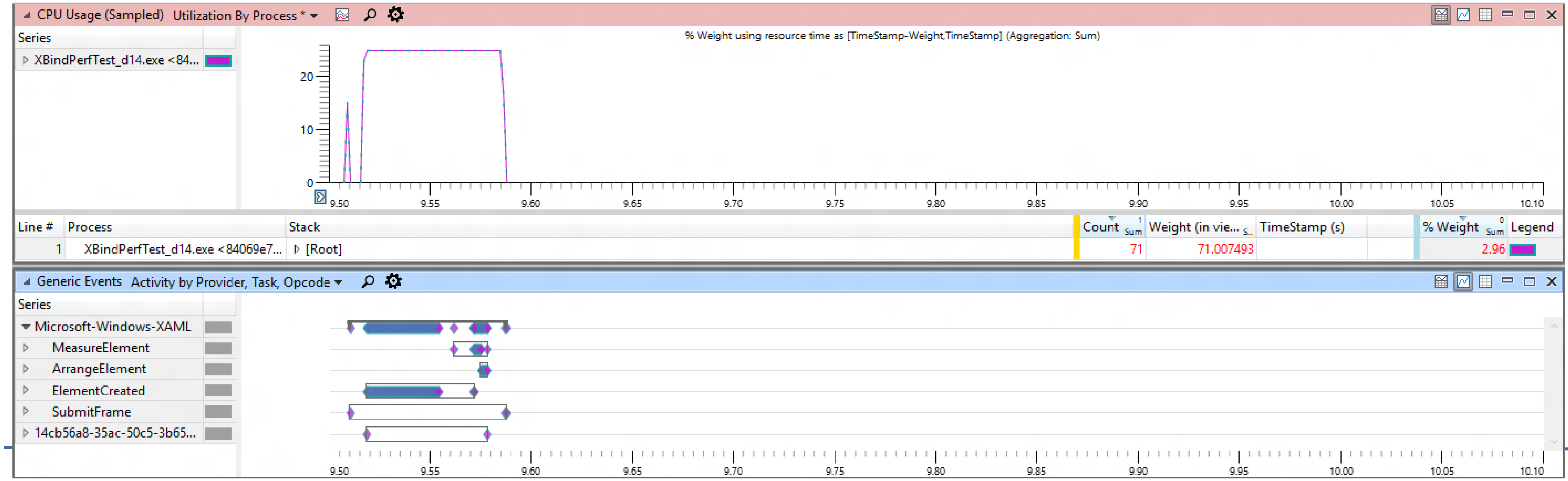
What is the problem with classic data binding?

Classic Binding


CPU Usage (Sampled) Utilization By Process *
% Weight using resource time as [TimeStamp-Weight,TimeStamp] (Aggregation: Sum)
Series
XBindPerfTest_d14.exe <84...
Generic Events Activity by Provider, Task, Opcode
Microsoft-Windows-XAML
MeasureElement
ArrangeElement
ElementCreated
SubmitFrame
14cb56a8-35ac-50c5-3b65...

Line #	Process	Stack	Count Sum	Weight (in vie... Sum)	TimeStamp (s)	% Weight Sum	Legend
1	XBindPerfTest_d14.exe <84069e7...	[Root]	496	496.056245		20.67	🟪

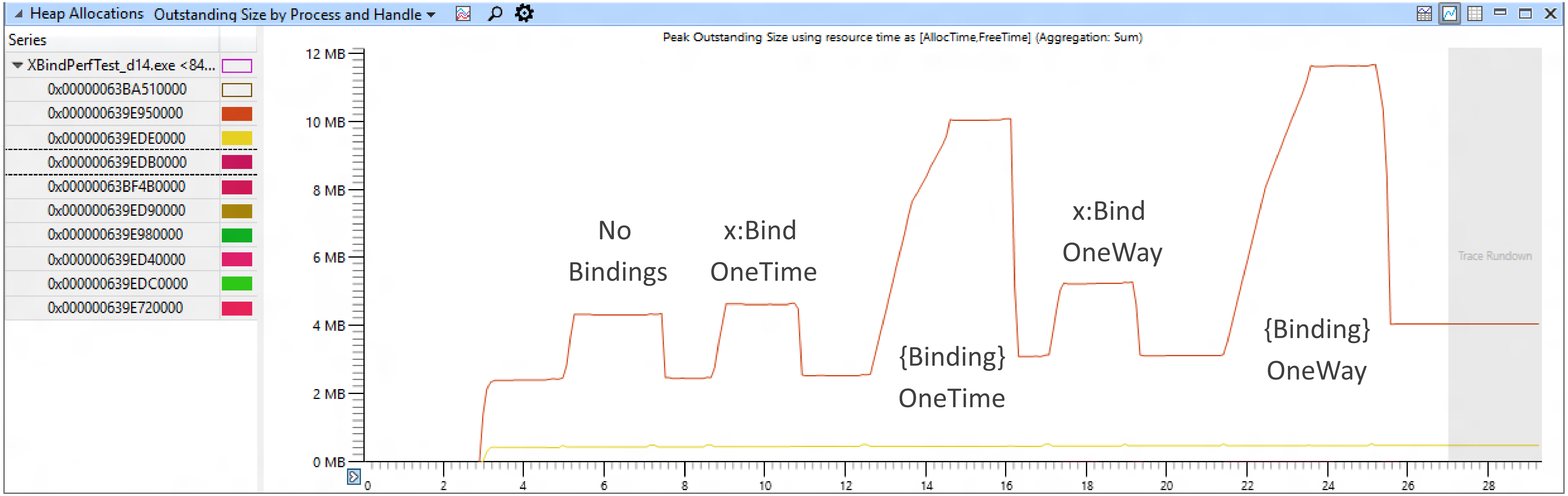
Compiled Binding


CPU Usage (Sampled) Utilization By Process *
% Weight using resource time as [TimeStamp-Weight,TimeStamp] (Aggregation: Sum)
Series
XBindPerfTest_d14.exe <84...
Generic Events Activity by Provider, Task, Opcode
Microsoft-Windows-XAML
MeasureElement
ArrangeElement
ElementCreated
SubmitFrame
14cb56a8-35ac-50c5-3b65...

Line #	Process	Stack	Count Sum	Weight (in vie... Sum)	TimeStamp (s)	% Weight Sum	Legend
1	XBindPerfTest_d14.exe <84069e7...	[Root]	71	71.007493		2.96	🟪

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Memory Comparison



Series	
XBindPerfTest_d14.exe <84...	▭
0x00000063BA510000	▭
0x000000639E950000	🟧
0x000000639EDE0000	🟨
0x000000639EDB0000	🟥
0x00000063BF4B0000	🟥
0x000000639ED90000	🟫
0x000000639E980000	🟩
0x000000639ED40000	🟥
0x000000639EDC0000	🟩
0x000000639E720000	🟥

1600 borders with their background databound

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

x:Bind

- Compiled binding
  - Bindings are committed at compile-time
- Strongly-typed binding
  - Duck binding is not supported
- Default mode is OneTime
  - OneWay and TwoWay are still available
- Standard binding approaches
  - INotifyPropertyChanged, IObservableVector, INotifyCollectionChanged

The data context of x:Bind is the code-behind class!!!

Syntax

```
<TextBox Text="{Binding
        Converter
        ConverterLanguage
        ConverterParameter
        ElementName
        FallbackValue
        Mode
        Path
        RelativeSource
        Source
        TargetNullValue
        UpdateSourceTrigger}
```

```
<TextBox Text="{x:Bind
        Converter
        ConverterLanguage
        ConverterParameter
        ~~ElementName~~
        FallbackValue
        Mode
        Path
        ~~RelativeSource~~
        ~~Source~~
        TargetNullValue
        ~~UpdateSourceTrigger~~}
```

Data Templates

```
<ListView ItemsSource="{x:Bind ViewModel.Employees}">

    <ListView.ItemTemplate>

        <DataTemplate x:DataType="model:Employee">

            <Grid>

                <TextBlock Text="{x:Bind Name}"/>

            </Grid>

        </DataTemplate>

    </ListView.ItemTemplate>

</ListView>
```

Syntax differences

```
 <ListView ItemsSource="{Binding Items}" Header="Classic" Grid.Column="0">
  <ListView.ItemTemplate>
    <DataTemplate>
      <TextBlock Text="{Binding Title}" />
    </DataTemplate>
  </ListView.ItemTemplate>
</ListView>
```

```
<ListView ItemsSource="{x:Bind ViewModel.Items}" xmlns:m="using:Blank3.Models"
            Header="Compiled" Grid.Column="1">
    <ListView.ItemTemplate>
        <DataTemplate x:DataType="m:TodoItem">
            <TextBlock Text="{x:Bind Title}" />
        </DataTemplate>
    </ListView.ItemTemplate>
</ListView>
```

Improve performance by simplifying your templates

Resource dictionaries

```xml
<ResourceDictionary
    x:Class="MyNamespace.MyTemplates"
    xmlns:model="using:xBindSampleModel">

    <DataTemplate
              x:Key="MyTemplate"
              x:DataType="model:Employee">
        <TextBlock Text="{x:Bind Name}" />
    </DataTemplate>

</ResourceDictionary>
```

```csharp
namespace MyNamespace
{
    public class MyTemplates
    {
        public MyTemplates()
        {
            InitializeComponent();
        }
    }
}
```

Referencing a dictionary

```
</UserControl.Resources>
    <ResourceDictionary>
        <ResourceDictionary.MergedDictionaries>
            <local:MyTemplates/>
            <ResourceDictionary Source="filename" />
         </ResourceDictionary.MergedDictionaries>
    </ResourceDictionary>
</UserControl.Resources>
```

Use Bindings.Update()
for async data (incl. OneTime)

Binding for Events

```
<Button Click="PokeEmployee">Poke Employee</Button>

<Button Click="{x:Bind Employee.Poke}'>Poke Employee</Button>
```

Signature

Have no parameters - `void Poke()`

Match event parameters - `void Poke(object sender, RoutedEventArgs e)`

Match event base types - `void Poke(object sender, object e)`

Overloading is not supported

Because all events are eligible:

This may replace ICommand & EventToCommand

Note: this does not include parameter or CanExecute

Bindings.StopTracking() pauses compiled bindings

How do I?

RelativeSource = Self & ElementName

Reference elements by name in Text="{x:Bind MyElement.Text}"

RelativeSource = TemplatedParent

Cannot use x:Bind in control templates; TemplateBinding is already optimized

Source / DataContext

Add a ViewModel to your code-behind

Page.ViewModel

```
public sealed partial class MainPage : Page
{
    public MainPage()
    {
        InitializeComponent();
        this.DataContextChanged += (s, e) =>
        {
            ViewModel = DataContext as ViewModels.MainPageViewModel;
        };
    }

    // strongly-typed view models enable x:bind
    public ViewModels.MainPageViewModel ViewModel { get; set; }
}
```

{x:Bind} is not for every situation

When to use classic binding

- Duck Typing
  - Text=“{Binding Age}” works for both PersonModel & WineModel
- Dictionary graphs
  - Use {Binding} with JSON or other untyped objects
- Code-behind binding
  - Can add/remove {x:Bind} @ runtime
- Use in a style
  - {x:Bind} can’t be used in a style for setters
  - {x:Bind} can be used in a DataTemplate that is defined in the style

x:Bind can meet your binding needs most of the time