Module 11

"Integrating Windows Forms and WPF"

Agenda

- Interoperability of WPF
- Using Windows Forms in WPF
- Using WPF in Windows Forms

Interoperability of WPF

- Windows Forms
  - Windows Forms -> WPF: **`WindowsFormsHost`**
  - WPF -> Windows Forms: **`ElementHost`**

- Native Win32
  - Win32 -> WPF: `HwndHost`
  - WPF -> Win32: `HwndSource`
    `HWND` via `WindowsInteropHelper`

Agenda

- Interoperability of WPF
- **Using Windows Forms in WPF**
- Using WPF in Windows Forms

WindowsFormsHost

- Manually add reference to
  - `WindowsFormsIntegration.dll`
  - `System.Windows.Forms.dll`
- `WindowsFormsHost`
  - **`Child`** property contains a single Windows Forms control
    - Add "**`x:Name`**" to refer to contained Windows Forms control in code
- Can be constructed
  - Declaratively in XAML
  - Programmatically in code
- Integrating Windows Forms controls into WPF
  - Properties can be set directly
  - Event handlers can be added seamlessly (but are CLR events!
  - Consider enabling Windows Form visual styles explicitly

PropertyMap

- Windows Forms and WPF have different property architectures
  - Properties can be translated via a **`PropertyMap`** object
  - Default mapping is supplied out-of-the-box
    - Can be modified and extended

- **`WindowsFormsHost.PropertyMap`**
  - **`Add()`**
    - Property name
    - **`PropertyTranslator`**
  - **`Remove()`**
    - Property name
  - **`...`**

- Example:
  - Translating **`UIElement.Clip`** to **`Form.Region`**

Windows Forms Dialogs

- Windows Forms built-in dialogs
  - `OpenFileDialog`
  - `SaveFileDialog`
  - `ColorDialog`
  - …
- Manually add reference to `System.Windows.Forms.dll`
- Same core principle for custom Windows Forms dialogs
- But…

WindowInteropHelper

- In namespace **System.Windows.Interop**

- To supply an WPF-owner to a Windows Forms dialog
  - Create **OwnerWindow** class implementing **System.Windows.Forms.IWin32Window**
  - Use **WindowInteropHelper** to retrieving interop properties of WPF parent
  - Create instance of **OwnerWindow** and store **WindowInteropHelper.Handle**
  - Call **System.Windows.Forms.ShowDialog()** with **OwnerWindow** instance

Agenda

- Interoperability of WPF
- Using Windows Forms in WPF
- **Using WPF in Windows Forms**

ElementHost

- `ElementHost` is already added to the Visual Studio 2012 toolbox
  - Drag it into the designer
    - This automatically adds necessary WPF assemblies
      - `PresentationFramework.dll`
      - `PresentationCore.dll`
      - `...`

- `ElementHost`
  - `Child` property contains a single WPF element
    - Cannot be set in the Forms designer – Only in code!

- `PropertyMap` can once again be applied to translate properties

- Different representations of
  - `Color`, `Size`, `Point`, `Rect`, …
- Convert colors with `Color.FromArgb()`

Summary

- Interoperability of WPF
- Using Windows Forms in WPF
- Using WPF in Windows Forms

Question

- You are developing a WPF application. The application contains a Grid named layoutRoot. You need to add a custom Windows Forms control called MyControl to layoutRoot. Which approach should you take?

a)
```
ElementHost host = new ElementHost();
host.Child = new MyControl();
layoutRoot.Children.Add( host );
```

b)
```
WindowsFormsHost host = new WindowsFormsHost();
host.Child = new MyControl();
layoutRoot.Children.Add( host );
```

c)
```
ElementHost host = new ElementHost();
host.Content = new MyControl();
layoutRoot.Children.Add( host );
```

d)
```
WindowsFormsHost host = new WindowsFormsHost();
host.Content = new MyControl();
layoutRoot.Children.Add( host );
```


WINCUBATE
Jesper Gulmann Henriksen
PhD, MCT, MCSD, MCPD
Phone : +45 22 12 36 31
Email : jgh@wincubate.net
WWW : http://www.wincubate.net
Hasselvangen 243
8355 Solbjerg
Denmark