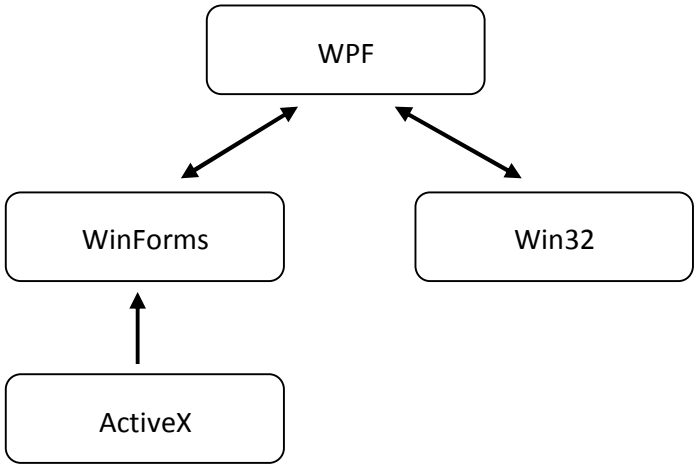
Workshop:

WPF und Integration von WPF-Elementen in Windows Forms

WinForms und WPF (Interoperabilität)

Mögliche Kombinationen



Technologische Unterschiede

Darstellung
WinForms ist Window-Handle-basiert, jedes Control ist aus Sicht von Windows ein eigenes Fenster mit eigenem Handle, welches für das Zeichnen seines Bereichs auf dem Bildschirm verantwortlich ist. Das Zeichnen auf Bereiche, die zu einem anderen Handle gehören, ist nicht möglich. Zur grafischen Darstellung wird GDI+ verwendet.

Die WPF dagegen zeichnet ihre Inhalte selbst. Pro Window gibt es nur einen Window-Handle (ausgenommen einige Steuerelemente wie Kontextmenüs). Dadurch können Controls übereinander gezeichnet werden, was z.B. Transparenzeffekte ermöglicht. Zur grafischen Darstellung wird DirectX verwendet.

Ereignisse
In WinForms wird ein normales Ereignis relativ einfach ausgelöst: Das Control unter dem Cursor, das aktiviert und sichtbar ist, erhält das Ereignis. Alle anderen Controls bekommen davon nichts mit.

In der WPF funktioniert dieses Prinzip nicht mehr, da Controls beliebig und sehr komplex ineinander verschachtelt werden können. Ereignisse wurden daher um eine Routing-Fähigkeit erweitert. Klickt der User auf einen Bildschirmbereich wird das Klick-Ereignis durch den Elemtenten-Baum zunächst als Vorschau nach unten durchgereicht (Tunneling) und das als normales Ereignis wieder nach oben zurückgereicht (Bubbeling). Auf diese Weise wird das Ereignis bei jedem Element in der Hierarchie ausgelöst. Es kann bei jedem Element geprüft werden, es können Aktionen ausgelöst werden und die Ereigniskette kann unterbrochen werden, weil z.B. bestimmte Bedingungen nicht erfüllt sind.

Aus diesen Unterschieden ergeben sich für die Interoperabilität gewisse Einschränkungen:

- Das gegenseitige Einbinden der Komponenten ist nicht uneingeschränkt möglich. Bindet man Windows Forms-Komponenten in eine WPF-Anwendung, liegen diese immer vor allen anderen Komponenten im Element Tree (ausgenommen sind WPF-Komponenten mit eigenem Fensterhandle, z.B. Menüs)

- Jeder Pixel innerhalb eines WPF-Windows/Win Forms kann nur eine Technologie beinhalten. Daher sind z.B. Transparenzeffekte nicht möglich

- Befinden sich mehrere Technologien übereinander, erhält nur die oberste die Tastatur- oder Mausereignisse. Sie müssen ggf. manuell an die anderen Technologien weitergeleitet werden.

- Fokuswechsel werden nur von der Technologie bemerkt, die gerade den Fokus besitzt.

- Die Inhalte einer gehosteten Technologie können nicht über Transformationen, z.B. Skalierung, manipuliert werden.

Workshop:
WPF und Integration von WPF-Elementen in Windows Forms

Windows Forms mit WPF-Elementen

Das Hosten von WPF-Elementen in WinForms erfolgt über den ElementHost, ein Control, das genau ein WPF-Element aufnehmen kann. Dabei kann es sich auch um einen WPF-Container handeln, der weitere WPF-Elemente enthalten kann. Innerhalb des gehosteten WPF-Elements bestehen keine Grenzen. Der ElementHost ist von System.Windows.Forms.Control abgeleitet und läßt sich wie jedes andere Control verwenden.

Im Windows-Projekt müssen zusätzlich folgende Assemblies eingebunden werden:
- PresentationCore.dll
- PresentationFramework.dll
- WindowsBase.dll
- WindowsFormsIntegration.dll

Innerhalb eines Formulars werden mind. folgende Verweise benötigt:
System.Windows.Forms.Integration - zur Verwendung der Klasse ElementHost
System.Windows.Controls - zur Verwendung der WPF-Controls
Hier empfiehlt es sich, mit einem Alias zu arbeiten:

```
Imports WPF = System.Windows.Controls
```

Die Deklaration eines WPF-Buttons muss dann über diesen Alias erfolgen, um ihn von einem WinForms-Button zu unterscheiden:

```
Dim wpfButton As WPF.Button = New WPF.Button
```

WPF-Dialoge aus Windows Forms öffnen

Um ein WPF-Window aus einer WinForms-Anwendung heraus öffnen zu können, wird zunächst ein Verweis auf das WPF-Projekt benötigt. Anschließend dann ein Objekt auf das WPF-Window instanziiert werden.

Öffnen eines WPF-Window als nicht modaler Dialog:

```
Dim win As WPFZeichenbrett.Window1 = New WPFZeichenbrett.Window1
ElementHost.EnableModelessKeyboardInterop(win)
win.Show()
```

Auszug aus der MSDN:
Rufen Sie die EnableModelessKeyboardInterop-Methode auf, um alle Tastaturmeldungen weiterzuleiten, wenn ein WPF-Fenster im nicht modalen Modus geöffnet wird.Die EnableModelessKeyboardInterop-Methode installiert einen Meldungsfilter in der Windows Forms-basierten Anwendung.Der Filter leitet alle Tastaturmeldungen weiter, wenn das nicht modale Fenster aktiv ist.

Öffnen eines WPF-Window als modaler Dialog:

```
Dim win As WPFZeichenbrett.Window1 = New WPFZeichenbrett.Window1
If win.ShowDialog = True Then
    MessageBox.Show("Sie haben OK geklickt.")
End If
```

Hier ist zu beachten, dass im Unterschied zu WinForms der Rückgabewert von ShowDialog ein Boolean-Wert (True/False) ist und kein DialogResult-Enum (Yes, No, OK, Cancel etc.)!

WPF mit WinForms-Elementen

Alle hier dargestellten Varianten funktionieren in gleicher Weise auch umgekehrt. Ein WinForms-Objekt kann in WPF in einem WindowsFormsHost gehostet werden. WinForms-Dialoge lassen sich ebenfalls aus einer WPF-Anwendung heraus starten. So lassen sich bestehende WinForms-Objekte zunächst in WPF einbinden, um sie ggf. später umzustellen.

Tips zum Weiterlesen und Ausprobieren

Bücher:

Anwendung = Code + Markup (C#), Code als Download
Autor: Charles Petzold, 2006 bei Microsoft Press, ISBN 978-3-86645-407-1

Vorteile: Viel WPF auch per Quellcode erläutert
Nachteile: bereits 2006 erschienen, einige Themen fehlen ganz (z.B. Interoperabilität)

Windows Presentation Foundation, Das umfassende Handbuch (C#), inkl. DVD
Autor: Thomas Claudius Huber, 2008 bei Galileo Computing, ISBN 978-3-8362-1108-6

Vorteile: sehr umfangreich, tiefgründige Erläuterungen über das Warum und Wie, kleines komplexes Beispiel
Nachteile: teilweise zu viel Erklärung, zu wenig Beispiel

Pro WPF with VB 2008 (englisch), Code als Download
Autor: Metthew MacDonald, 2008 bei Apress, ISBN 978-1-59059-962-4

Vorteile: VB, sehr umfachreich, alle Themen enthalten einschl. Deployment
Nachteile: englisch

Windows Presentation Foundation (C#), inkl. CD
Autor: Dirk Frischalowski, 2007 bei Addison-Wesley, ISBN 978-3-8273-2522-8

Vorteile: sehr "multimedia-lastig", separates Kapitel zu Expression Blend
Nachteile: sehr wenig zu Datenbindung und Datenverwaltung

Windows Presentation Foundation (C#), Code als Download
Autor: Jörg Wegener, 2009 Hanser Verlag, ISBN 978-3-446-41041-1

Vorteile: durch klaren und logischen Aufbau schnelles Einarbeiten möglich, ausführliche Einführung
Nachteile: reines Einführungsniveau

Beispiele und Demos:

www.codeplex.com

msdn.microsoft.com/de-de/default.aspx
- MSDN-Library
- Windows SDK
- Webcasts
- CodeClips

www.dotnetframework.de/dotnet/worldwidewings.aspx
Kostenlose .NET 3.5-Beispielanwendung World Wide Wings von Dr. Holger Schwichtenberg