



Universal-Apps

activeVB Workshop 2014

30. August 2014



Auszug aus dem Buch:

Windows Phone 8.1 Kochbuch

Erscheinungsdatum: Q4/2014

Autoren: Matthias Fischer, Gordon Breuer

Gordon Breuer

Senior IT Consultant
Software Engineer / Architect

msg systems ag

Silverlight
Fantasy & RPG
Windows 8
Musik
Marketing
SQL
Software-Architektur
Mediengestalter
Schulungen
.NET-Development
C#
User Experience
Social Media
Schreiben
Videobearbeitung
User Interfaces
WPF
Windows Phone
Augmented Reality

Gordon Breuer

Senior IT Consultant
Software Engineer / Architect
msg systems ag



Windows Phone 8.1



Windows 8.1



mail@gordon-breuer.de

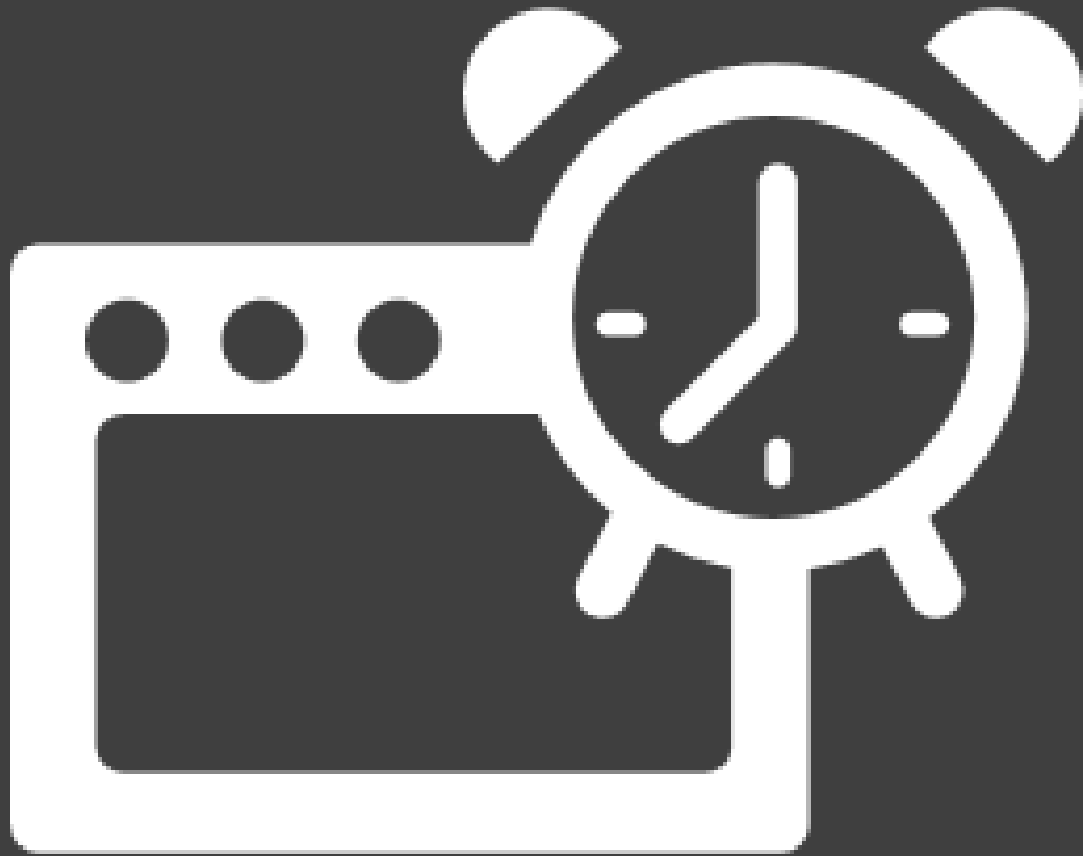
+49 172 5717667

gordon.breuer

@anheledir

anheledir

gordonbreuer



Ankündigungen 2014

Vorträge und Veröffentlichungen



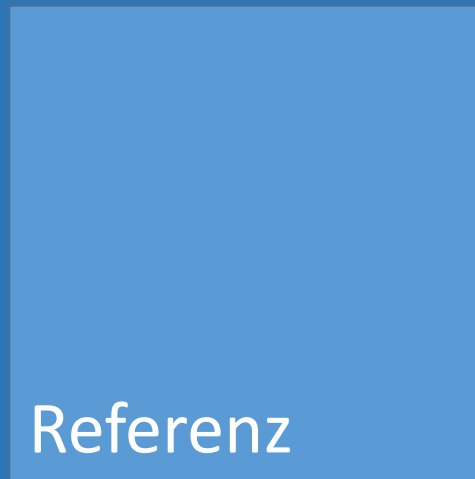
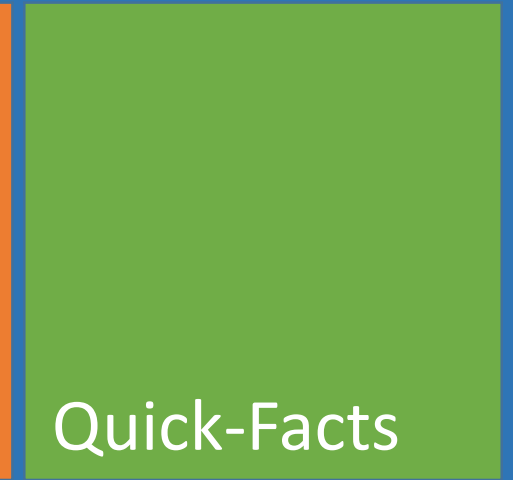
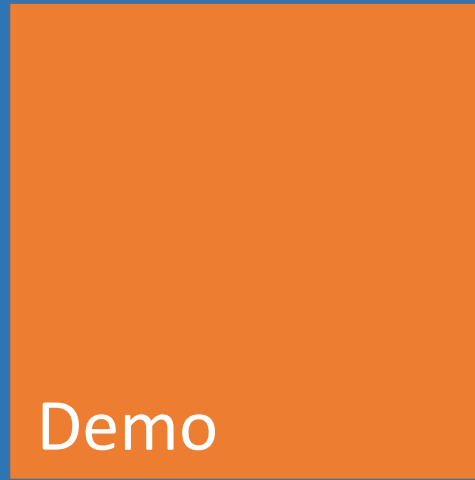
Vortrag Going Blue: Neues in Windows Phone 8.1



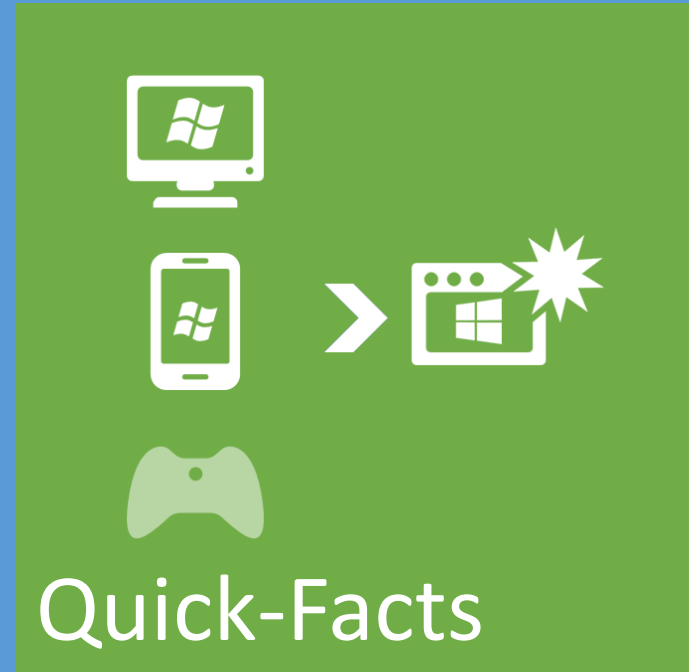
Buch: Windows Phone 8.1 Kochbuch

Erscheinungsdatum: Q4/2014

Autoren: Matthias Fischer,
Gordon Breuer



Universal-Apps



Universal-Apps



Rückblick

- **Windows Phone 7.x**

- Silverlight für Windows Phone
- XNA-Framework

- **Windows Phone 8.0**

- Silverlight für Windows Phone
- XNA-Framework
- WinPRT
- Native Schnittstellen

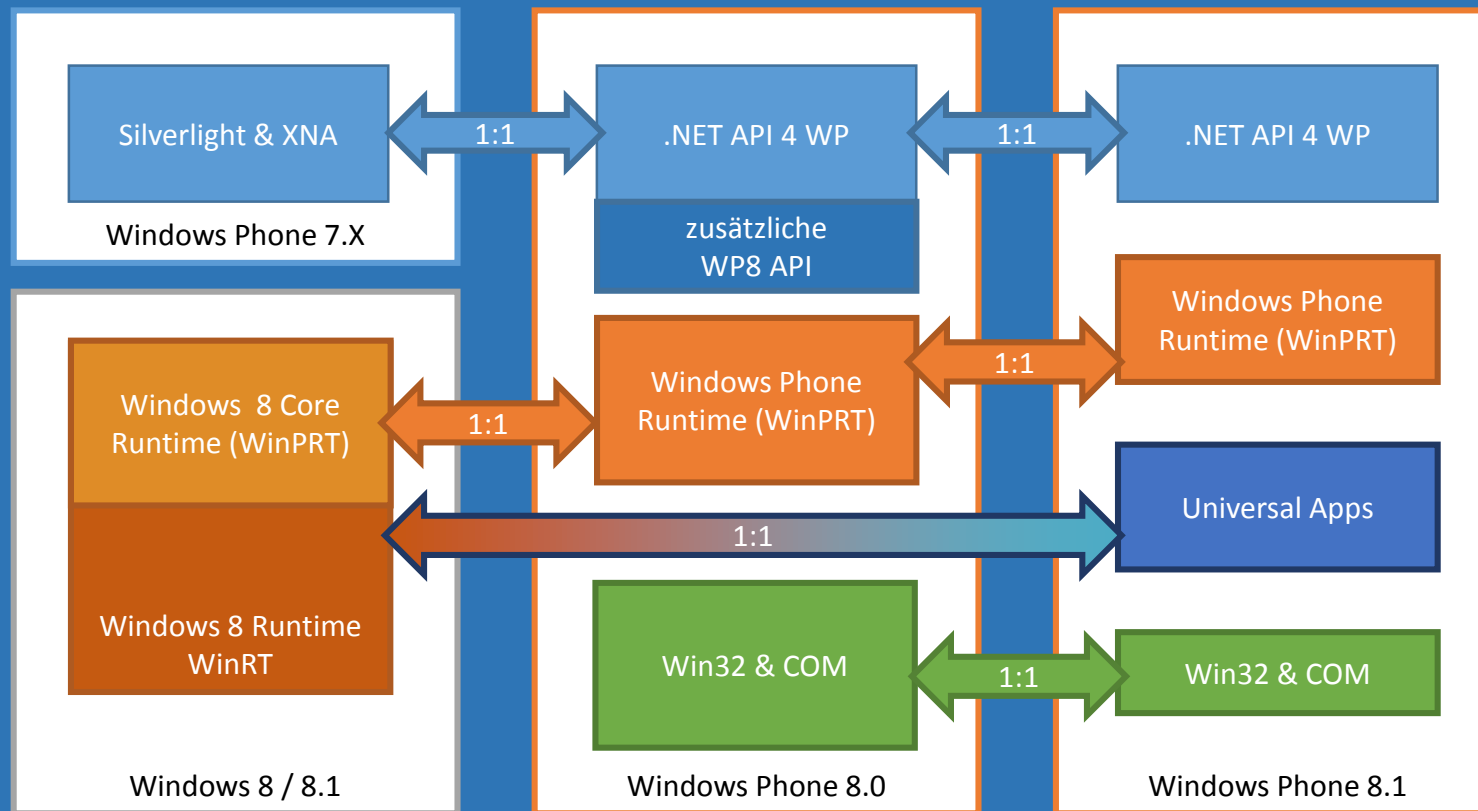
- **Wahl der API vor allem abhängig von der Art der Kompatibilität...**

- ...mit Windows Phone 7.x = Silverlight-App
- ...mit Windows 8 = WinPRT-App



Universal-Apps | Rückblick

- Der neue Projekt-Typ der Universal-Apps basiert auf der WinRT-API
- Neben dem Code-Sharing ermöglichen diese nun auch das gemeinsame Verwenden visueller Elemente

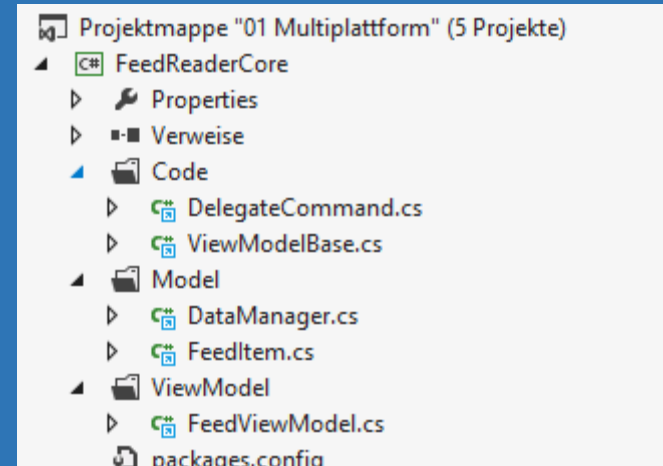




Alternativen

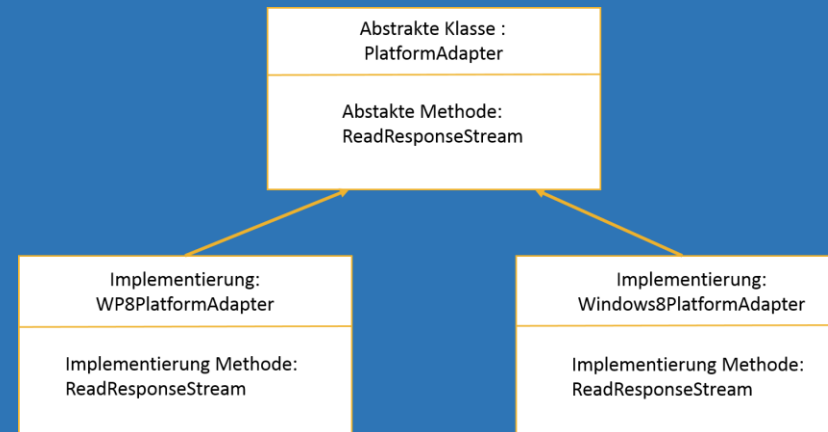
Universal-Apps | Alternativen

- Copy, Cut & Paste
- Dateien verlinken
- Präprozessor Anweisungen



```
#if !SILVERLIGHT
using FeedreaderCorePortableWinRT.Code.Utilities;
#endif
```

- Portable Class Library
- Plattform Adapter





Universal-Apps

Universal-Apps

- Universal-Apps basieren auf der Win(P)RT-API
- Lauffähig unter Windows 8.1 und Windows Phone 8.1
 - Geplant ist auch die Unterstützung durch die XBOX ONE
- Großteil der Ressourcen lassen sich zwischen den Projekten für die jeweiligen Plattformen gemeinsam nutzen
- Erforderlich war hierfür die Angleichung der WinRT- und der WinPRT-APIs, sowie die Anpassung von Steuerelementen um auch XAML-Dateien und Views gemeinsam verwenden zu können



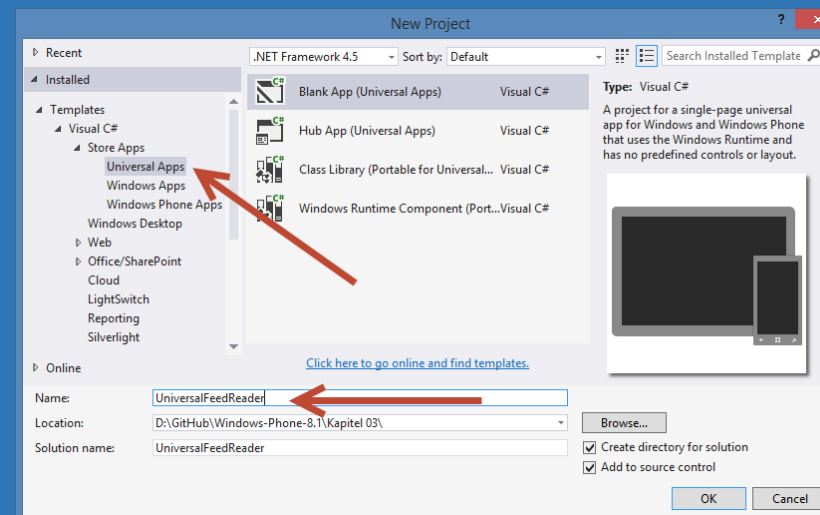
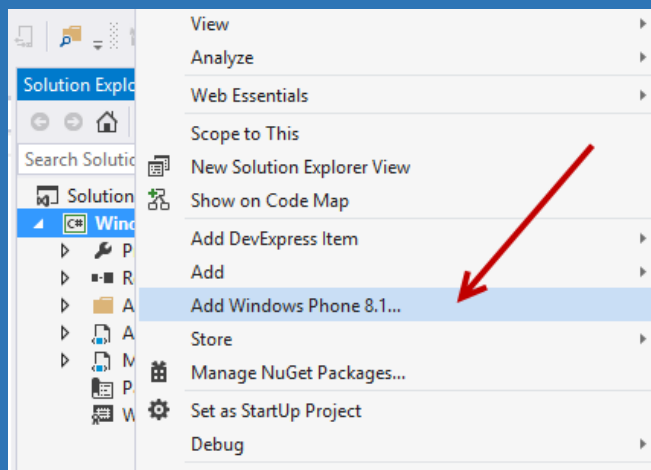
	Klassen	Strukturen	Schnittstellen
Windows 8.1	566	119	59
Windows Phone 8.1	624	131	57



Universal-Apps

Universal-Apps

- Es gib drei Möglichkeiten eine Universal App zu erstellen
 1. Konvertieren einer auf der WinRT basierenden Windows 8.1-App
 2. Konvertieren einer auf der WinPRT basierenden Windows Phone 8.1-App
 3. Erstellen einer neuen Projektvorlage (leer oder mittels Assistent)
- Bei der Konvertierung spielt es keine Rolle, ob das Projekt mit C#, JavaScript oder C++ programmiert wurde
- Es können keine Projekte basierend auf Silverlight konvertiert werden!



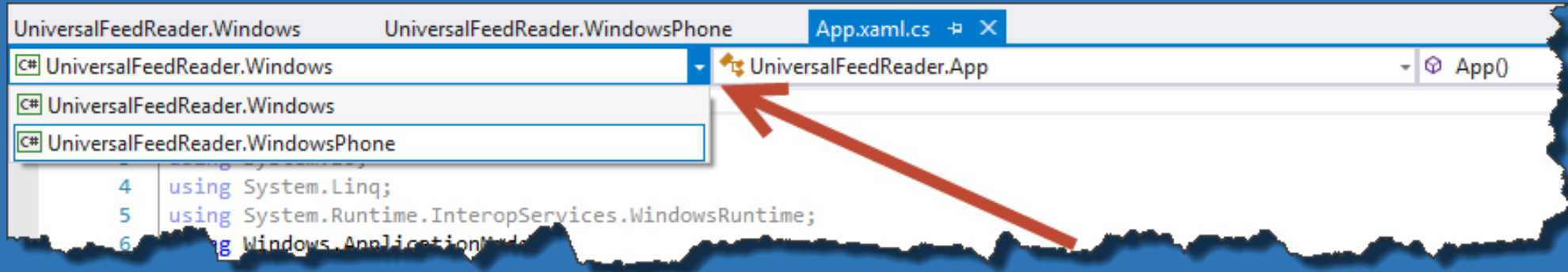
Universal-Apps

- Aus der Sicht von Visual Studio 2013 sind Universal Apps lediglich eine Weiterentwicklung der Quellcode-Verlinkung
- Beim Anlegen einer Universal App erstellt Visual Studio 2013 drei Projekte:
 - „Shared“ für gemeinsame Programmteile und Ressourcen
 - „Windows“ für Programmteile und Ressourcen spezifisch für die Windows-App
 - „WindowsPhone“ für Programmteile und Ressourcen spezifisch für die WinPhone-App
- Alle Ressourcen und Quellen des geteilten Projektes werden für beide Plattformen verwendet
- Geringe Abweichungen im Quellcode lassen sich weiterhin mit Präprozessor-Anweisungen kennzeichnen:
 - `WINDOWS_PHONE_APP`
 - `WINDOWS_APP`



Universal-Apps

- Weiterhin bietet Visual Studio 2013 die Möglichkeit, die Ansicht des Quellcodes speziell für eine der beiden Plattformen umzuschalten für eine leichtere und schnellere Bearbeitung



```
UniversalFeedReader.Windows  UniversalFeedReader.WindowsPhone  App.xaml.cs  + X
C# UniversalFeedReader.Windows  UniversalFeedReader.App  App0
C# UniversalFeedReader.Windows
C# UniversalFeedReader.WindowsPhone
4 using System.Linq;
5 using System.Runtime.InteropServices.WindowsRuntime;
6 using Windows.ApplicationModel;
```

- Universal-Apps verwenden Code-Sharing, und kein Binary-Sharing wie bei der Portable Class Library
- Für jede Plattform wird ein eigenes Projekt gebaut und im Store veröffentlicht

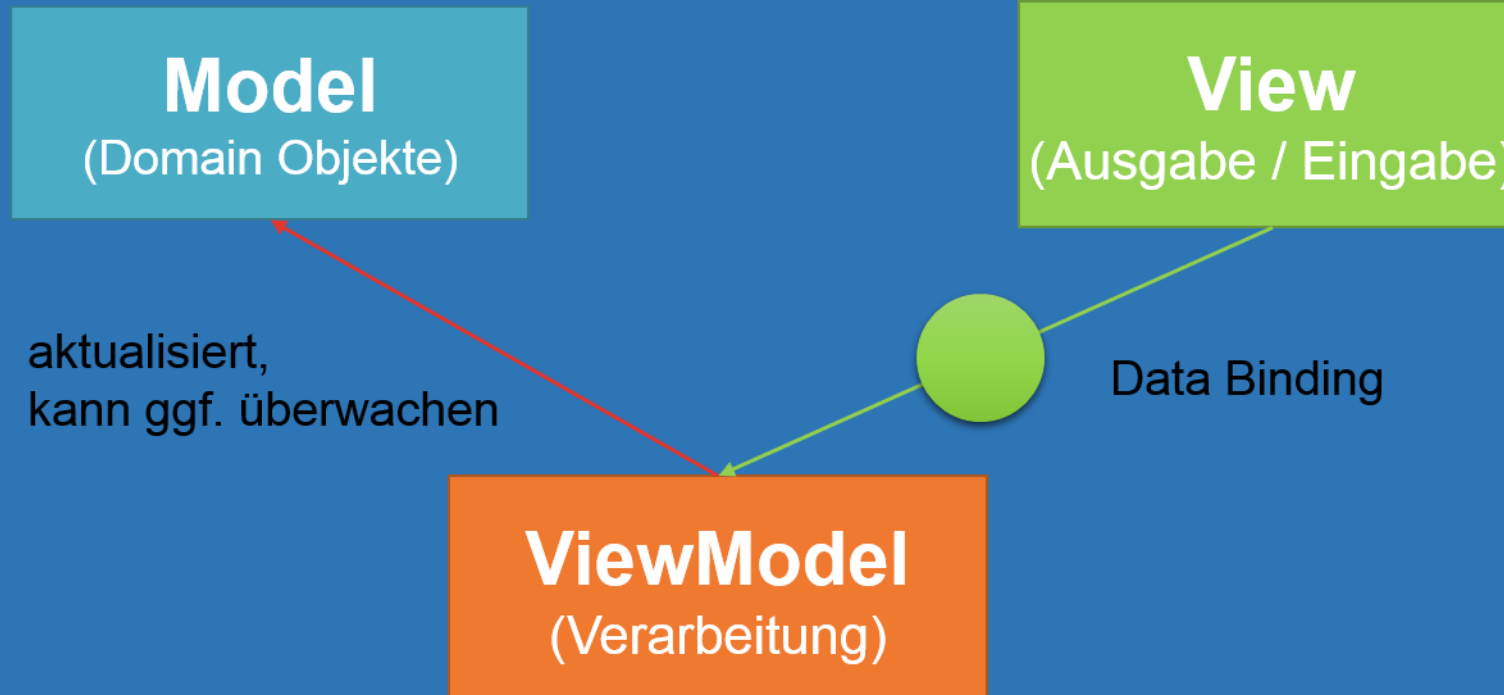
Universal-Apps

- Der App-Name, der beim Anlegen einer neuen Windows (Phone) App im Store eingetragen wird, bleibt für ein Jahr in beiden Stores reserviert
- Nach dem Hochladen der Apps für beide Plattformen können diese im Store miteinander verknüpft werden
- App muss nur auf einer der beiden Plattformen gekauft werden
 - In-App Produkte können ebenfalls für beide Apps geteilt werden
 - Kann je Produkt individuell ausgewählt werden
- APIs für das Roaming von Daten (bsp. Einstellungen) synchronisieren plattformübergreifend





MVVM-Pattern



```
View.DataContext = ViewModel;
```

- Für die Implementierung des MVVM-Pattern in Universal-Apps bieten sich entsprechende Hilfsklassen im Shared-Projekt an
- **Basisklasse ViewModel:**



```
public class ViewModelBase : INotifyPropertyChanged {  
    public event PropertyChangedEventHandler PropertyChanged;  
  
    protected virtual void OnPropertyChanged([CallerMemberName] string propertyName = null) {  
        PropertyChangedEventHandler handler = PropertyChanged;  
        if (handler != null) handler(this, new PropertyChangedEventArgs(propertyName));  
    }  
}
```



• Basisklasse DelegateCommand:

```
public class DelegateCommand : ICommand {
    readonly Func<object, bool> _canExecute;
    readonly Action<object> _executeAction;

    public DelegateCommand(Action<object>
executeAction)
        : this(executeAction, null) {
    }

    public DelegateCommand(Action<object>
executeAction,
        Func<object, bool> canExecute) {
        if (executeAction == null) {
            throw new
                ArgumentNullException("executeAction");
        }
        this._executeAction = executeAction;
        this._canExecute = canExecute;
    }
}
```

```
public bool CanExecute(object parameter) {
    bool result = true;
    if (_canExecute != null) {
        result = _canExecute(parameter);
    }
    return result;
}

public event EventHandler CanExecuteChanged;
public void RaiseCanExecuteChanged() {
    if (CanExecuteChanged != null) {
        CanExecuteChanged(this,
            new EventArgs());
    }
}

public void Execute(object parameter) {
    _executeAction(parameter);
}
}
```

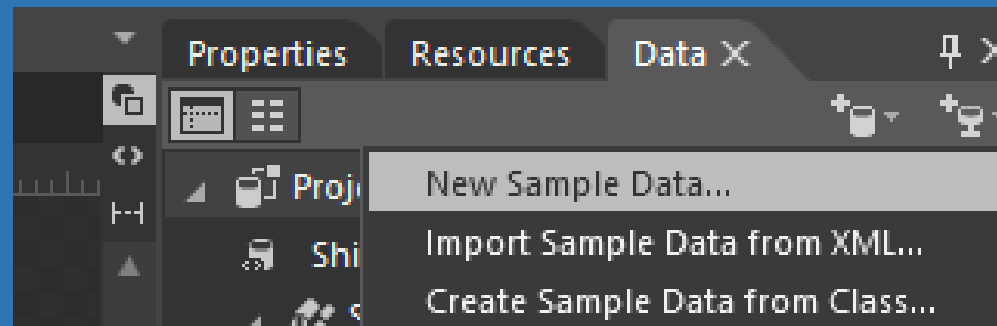




DesignMode.
DesignModeEnabled

Universal-Apps | IsInDesignMode

- Darstellung von Musterdaten im Designer von Visual Studio und/oder Blend
- Vereinfacht die Erstellung und Bearbeitung der Views
- Unterscheidung im Konstruktor (c-tor) des ViewModel
 - Im DesignMode werden vorbereitete Musterdaten für das Model geladen
 - Zur Laufzeit die Bindung an die tatsächliche Datenquelle
- Blend unterstützt die Generierung von plausiblen Beispieldaten über einen Assistenten





• Auszug aus dem c-tor des ViewModels

```
public class FeedReaderViewModel : ViewModelBase {  
    public FeedReaderViewModel() {  
        if (DesignMode.DesignModeEnabled) {  
            FeedItems = new ObservableCollection<FeedItem> {  
                new FeedItem {  
                    ArticleURL =  
"http://demo.server.de/some/path/someId.html",  
                    DatePublished = DateTime.Now.ToString("f"),  
                    Title = "Design Entry 01",  
                    Description = "Das ist ein Design Dateneintrag für die  
Visualisierung im UI Designer oder Blend."  
                },  
                new FeedItem {  
                    ArticleURL =  
"http://demo.server.de/some/path/someId.html",  
                    DatePublished = DateTime.Now.ToString("f"),  
                    Title = "Design Entry 02",
```

```
                    Description = "Das ist ein noch ein Design Dateneintrag,  
mit einer anderem Beschreibung."  
                },  
            };  
        } else {
```

```
            RefreshCommand = new DelegateCommand(async arg => {  
                var res = await DataManager.Instance.  
UpdateFeed("http://dotnetautor.de/GetRssFeed");  
                FeedItems = new ObservableCollection<FeedItem>(res);  
            });  
        }  
    }  
}
```



A diagram on a yellow background illustrating cross-platform development. On the left, three icons are stacked vertically: a desktop monitor with the Windows logo, a smartphone with the Windows logo, and a game controller. A white arrow points from these icons to a single icon on the right representing a desktop window with the Windows logo and a starburst effect, symbolizing the output of a single code example running on multiple platforms.

Code-Beispiel

Universal-Apps



RSS-Feedreader



Datenmodelle

FeedItem
<i>string</i> : Title
<i>string</i> : Description
<i>string</i> : DatePublished
<i>string</i> : ArticleURL



FeedItem

DataManager (singleton)
<i>Task<IEnumerable<FeedItem>></i> : UpdateFeed (string)



DataManager

MVVM-Hilfsklassen



ViewModelBase



DelegateCommand

ViewModel

FeedReaderViewModel : ViewModelBase

ObservableCollection<FeedItem>: FeedItems
DelegateCommand: RefreshCommand



FeedReaderView
Model



Ansichten

- Gemeinsam genutztes Template für die Darstellung der Einträge im RSS-Feed



Design Entry 01

Das ist ein DesignDateneintrag für die
Visualisierung im UI Designer oder Blend.

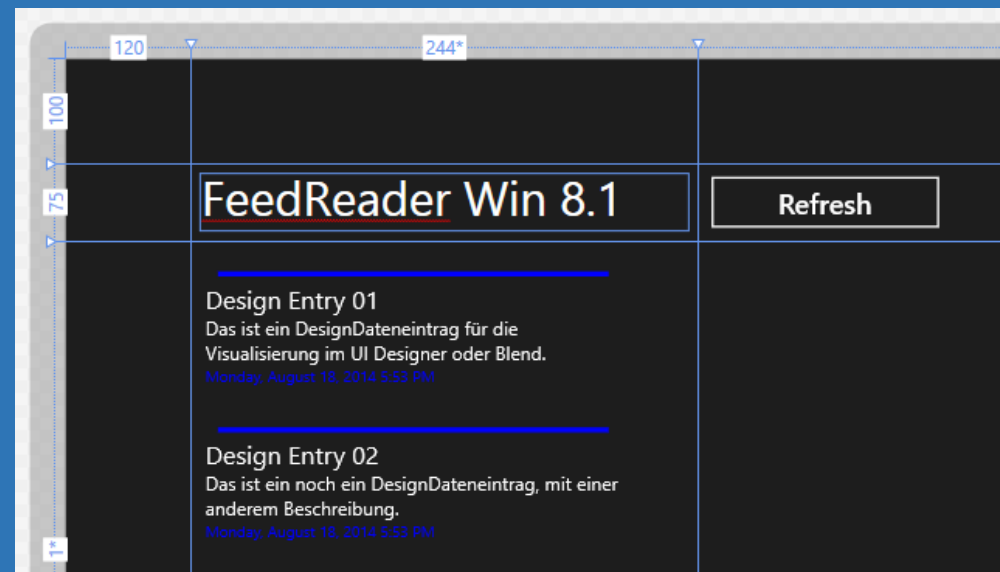
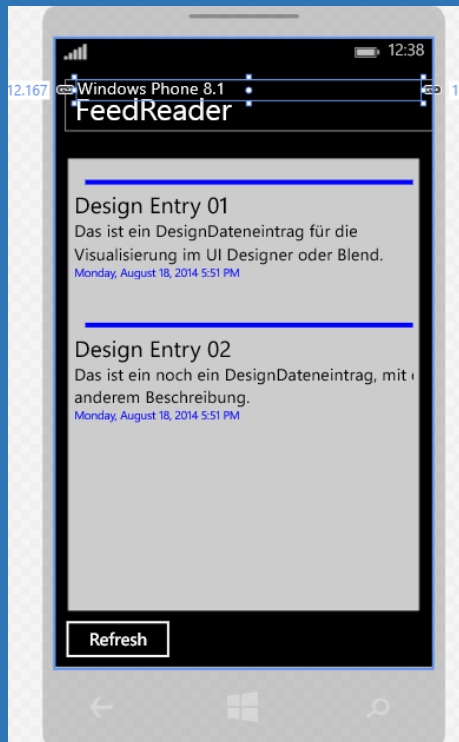
Monday, August 18, 2014 5:51 PM



FeedItemTemplat
e

Ansichten

- Startseite mit Übersicht aller Einträge soll plattformabhängig gestaltet werden
 - Windows Phone: Alle Einträge untereinander
 - Windows: Alle Einträge untereinander und nebeneinander (zweispaltig)



MainPage
Windows Phone



MainPage
Windows



Chatroom

Offene Fragen- und Diskussionsrunde



Vielen Dank!



Gordon Breuer



Vielen Dank!



Auszug aus dem Buch:

Windows Phone 8.1 Kochbuch

Erscheinungsdatum: Q4/2014

Autoren: Matthias Fischer, Gordon Breuer