Matthias Fischer | www.it-visions.de | dotnetautor.de

Say „Hello“ to Windows Phone 8

Zeit:
27. September 2013, 09.00 to 17.00

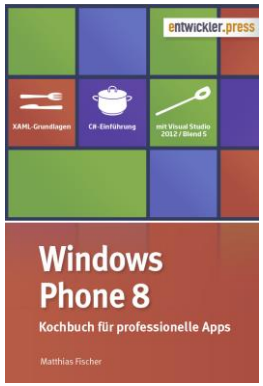
Sie sind Entwickler und haben bereits Erfahrungen mit der Programmierung von Anwendungen mit Visual Studio und C#, Sie stehen vor oder mitten in einem Projekt mit Windows Phone oder Sie interessieren sich für die Welt von Apps und Co., dann sind Sie hier genau richtig. In diesem praxisorientierten Workshop bekommen Sie einen Einblick in die Welt der mobilen Anwendungsentwicklung sowie einen Überblick über die aktuellen Entwicklungswerkzeuge. Entwickeln Sie gemeinsam mit Matthias Fischer eine mobile App für Windows Phone 8.

Referentenvorstellung

BASTA!

- Consultant und Trainer
  - .NET-Entwicklung seit 2001
  - ASP.NET, WCF, MVC4, SQL Server 2012
  - WPF, MVVM, Windows Phone 8, Windows 8
- Autor (Auswahl)
  - Carl Hanser Verlag, Addison-Wesley, Wrox,
  - Windows.Developer
- Projekte / Apps
  - OSMLogger, BatchUploader, MobileTech Conference usw.

NOKIA Developer Certified Trainer



Matthias Fischer

NOKIA Developer Champion

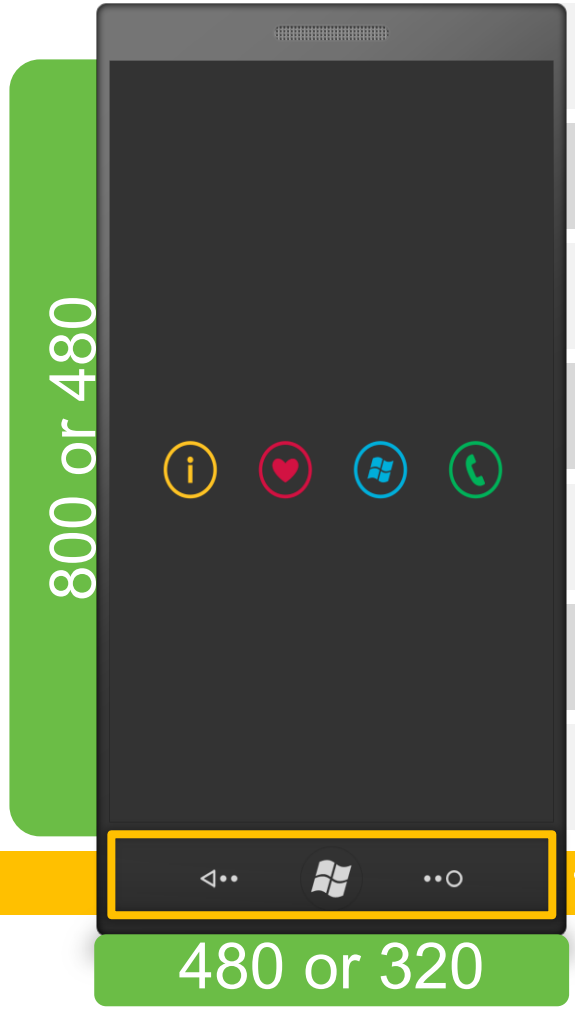
- Material zu diesem Workshop : http://bit.ly/mfslides
- Mehr Informationen: www.dotnetautor.de
- WP 8 Seminare : dotnetautor.de/training/wp8
- Kontakt : matthias@dotnetautor.de

Premium Developer Program

- developer.nokia.com/windowsphone
- developer.nokia.com/entwicklergeraet
- entwickler@nokia.com

Windows Phone Hardware

[optional]
unavailable



Capacitive touch
4 or more contact points

Sensors
A-GPS, Accelerometer, [Compass], Light, Proximity, [Gyro]

Camera
[Back 5 mega pixels or more], [Font 1,2 mega pixels or more]

Multimedia
Common detailed specs, Codec acceleration, FM Radio

Memory
256MB RAM or more, 8GB Flash or more

GPU
DirectX 9 acceleration

CPU
ARMv7 Cortex/Scorpion or better

6 buttons | Back, Start, Search, Volume, Power, Photo

Windows Phone 8 Hardware

*) Installtion Packages Only

Screen Resolution
3 different screen resolutions

Multi-Core
Support for multicore (up to 64)

SD-CARD
SD-Card for media, music and program installation

Internet Explorer 10
Anti mobile phishing

Native C++, Direct X, XAudio2
For a better gaming performance

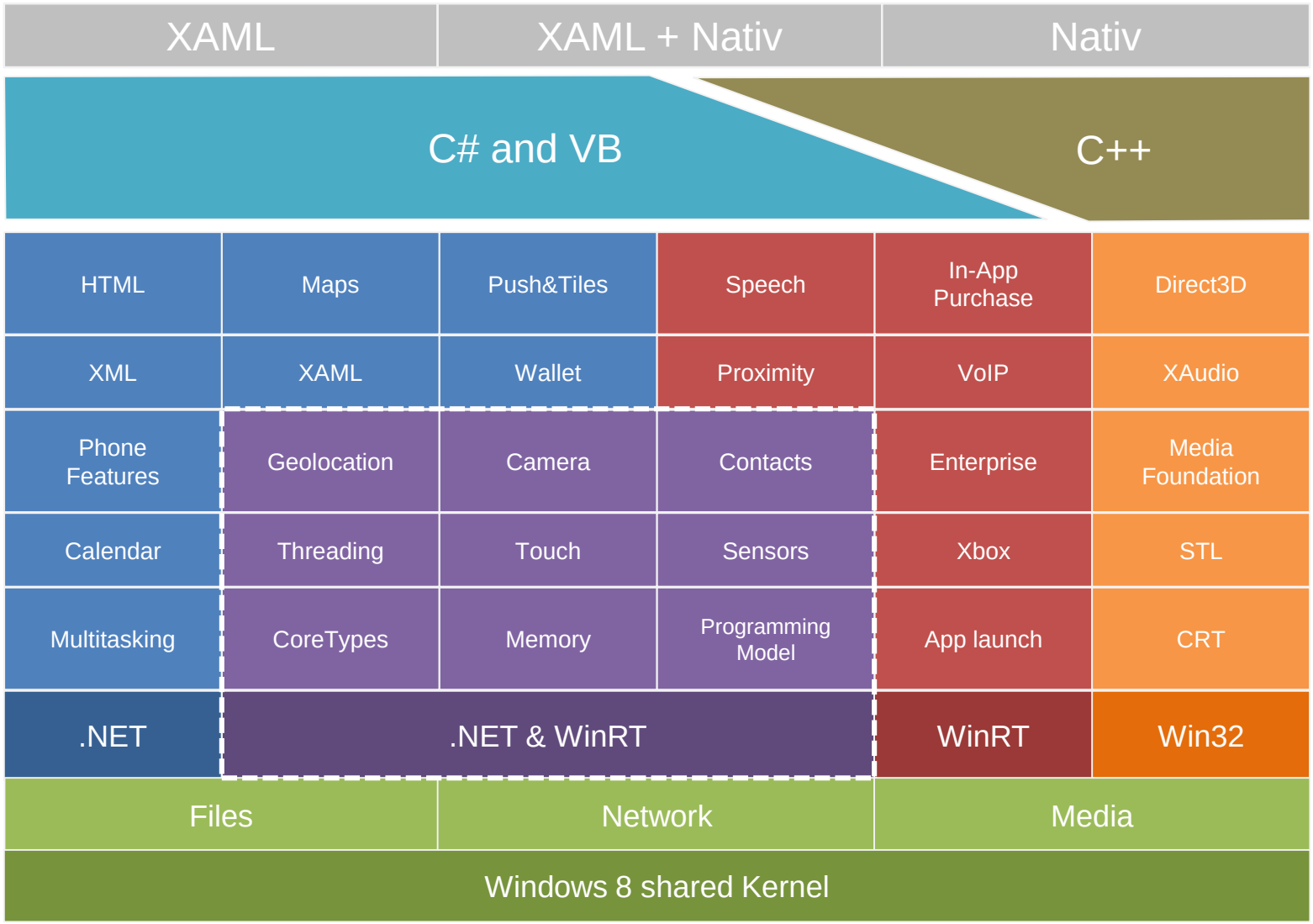
NFC
Wallet Experience & p2p communication

Nokia HERE Maps
New offline navigation API

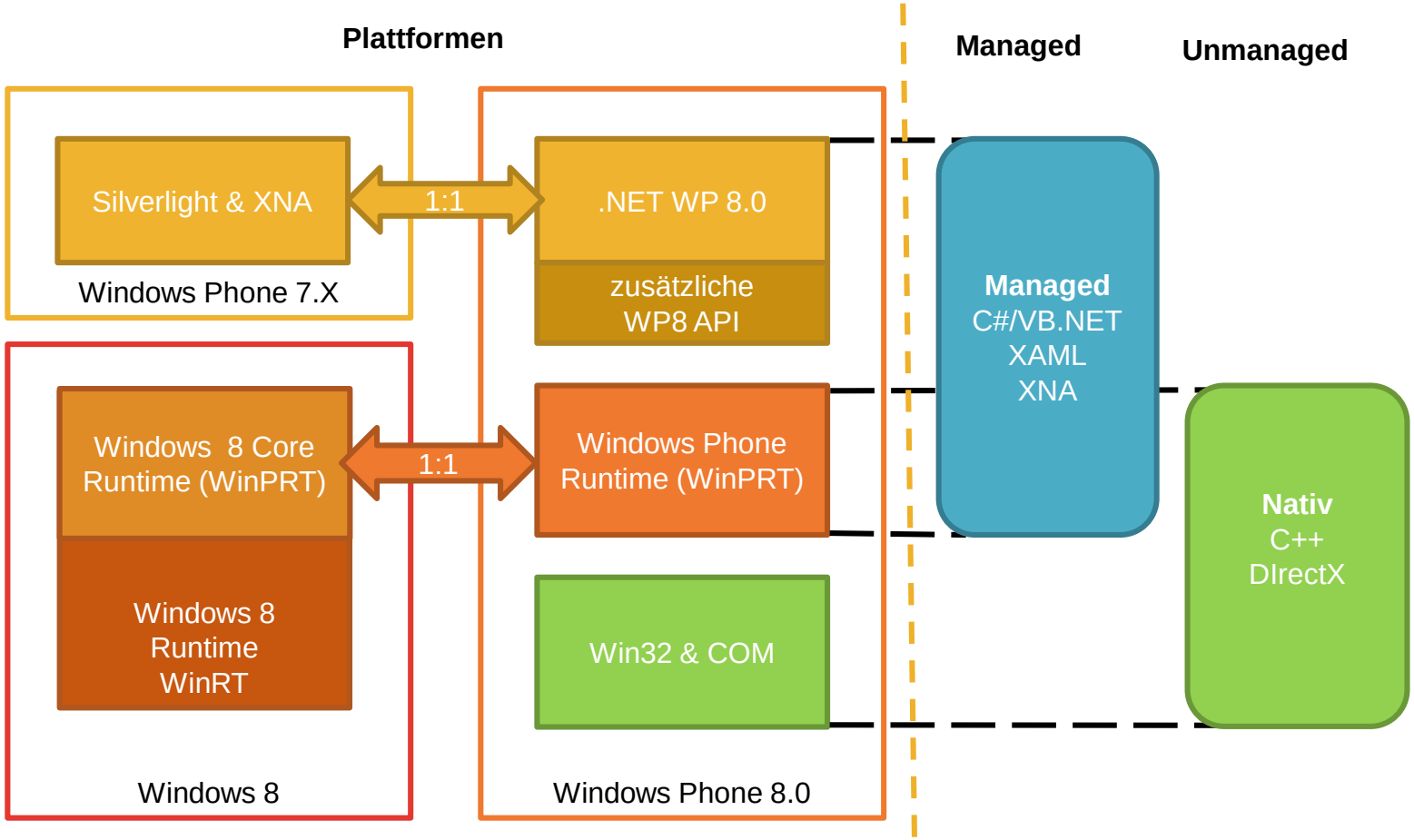
VoIP
Integrated VoIP API

... and many more

Windows Phone 8 Apps

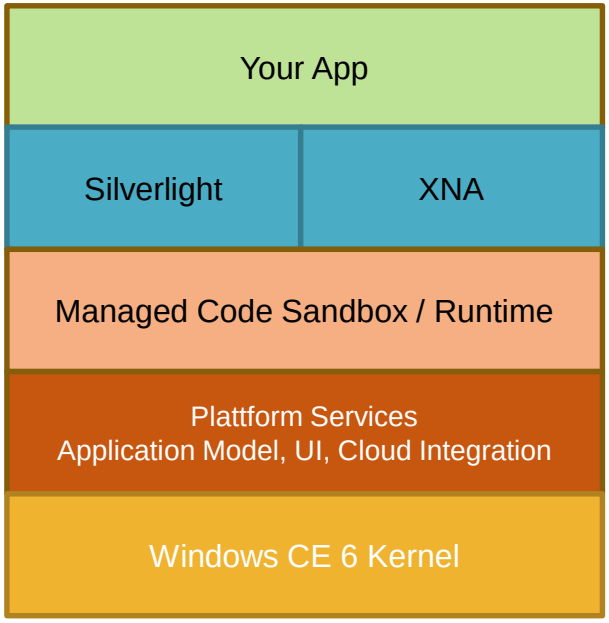


Windows Phone 8 - Entwicklersicht

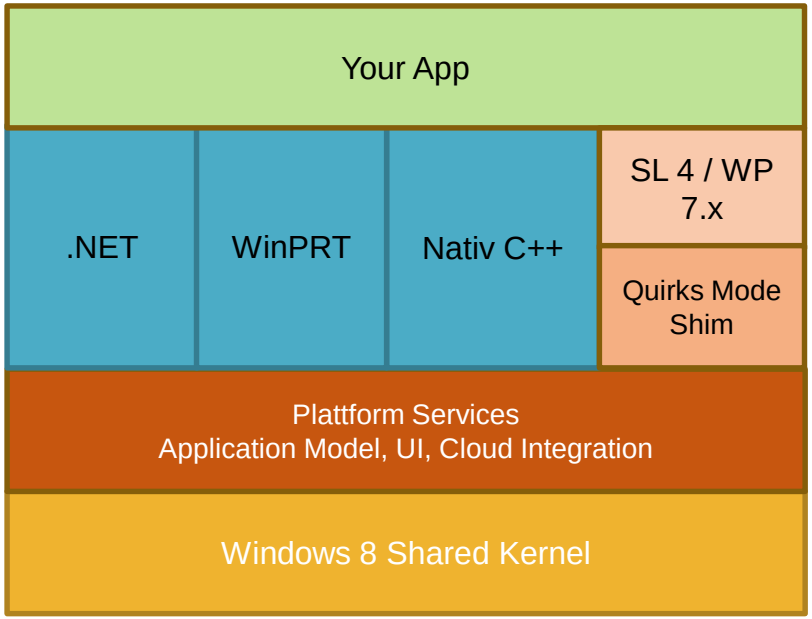


Die neue alte Plattform

Windows Phone 7.x



Windows Phone 8



WP8 Features Overview

BASTA!

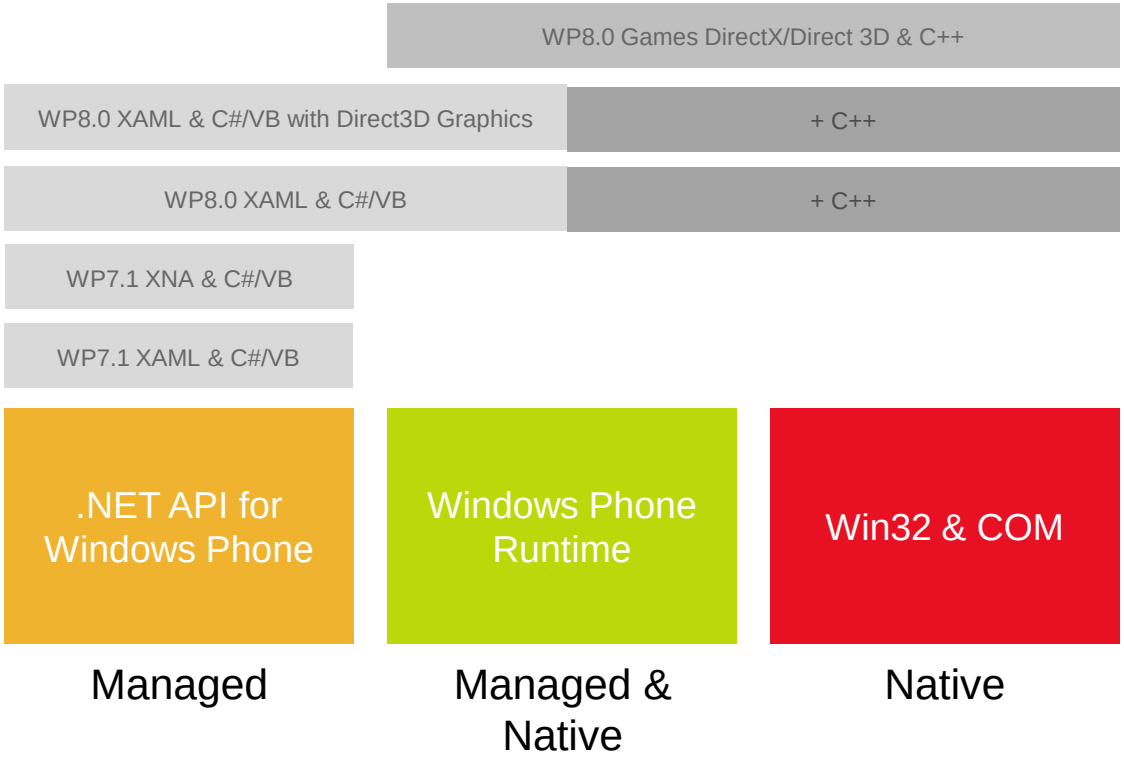
New features

Camera low level access	Lenses	New Maps	In-app purchase	Enterprise Apps
NFC	SD Card	Speech	File & URI Associations	Data Sense
Wallet	Internet Explorer 10	VoIP	Lockscreen	DirectX

Extended features

Managed	480 × 800	Tiles	Bluetooth	Launchers	Background Agents	Controls
Native	720 × 1280	New Sizes	App 2 App	Maps	Location	Long List Selector
HTML5	768 × 1280	New Types	App 2 device	Appointment	Audio improvements	New Pivot & Panorama
VB.NET				Share Media	Photo upload	

Windows Phone 8 APIs



- Windows Phone 8
  - Managed Programmierung WP7.1, WP8.0 .NET und Windows Phone Runtime
  - Native Programmierung WinPRT und Win32
  - Spieleprogrammierung mit dem WP7.1 XNA Framework
  - Spieleprogrammeirung mit Direct3D / DirectX

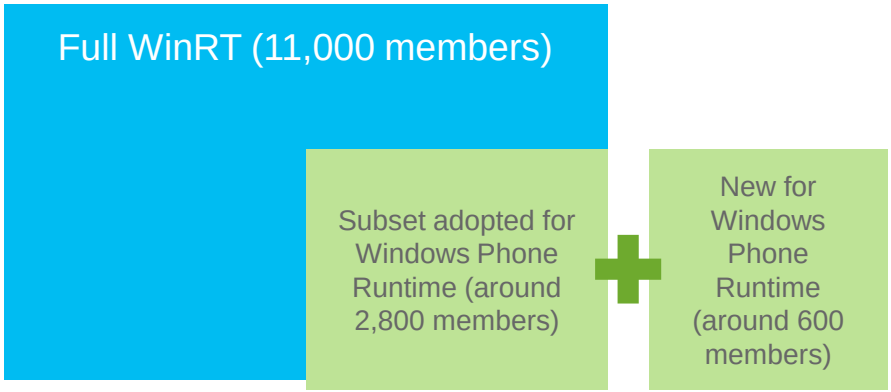
.NET API for Windows Phone

- Die **.NET API für Windows Phone** stellt die primäre API dar
  - Schießt *alle* Typen und API's von Windows Phone OS 7.* ein
  - Enthält Klassen und Typen aus dem **System** und **Microsoft.Phone** Namensraum

- Für Windows Phone 8.0 wurden neue Klassen und Namensräume eingefügt z.B.
  - Microsoft.Phone.Wallet
  - Microsoft.Phone.Tasks.Sharemediatask
  - Microsoft.Phone.Tasks.Mapstask
  - Microsoft.Phone.Storage.Externalstorage
  - Microsoft.Phone.Networking.Voip
  - u.v.m.

Windows Phone Runtime API

- Die **Windows Phone Runtime** ist eine Untermenge der WinRT plus spezielle Phone-Erweiterungen
  - Die Win(P)RT ist nativ in C/C++ implementiert und wird nach C#, VB.NET, und C++ „projiziert“
  - Für HTML5/JavaScript ist z.Z. (noch) keine Projektion verfügbar unter Windows Phone 8



- Die Phone-Erweiterungen der WinRT enthalten unter anderem :
  - Sprachsynthese und -erkennung
  - Windows.Phone.PersonalInformation
  - LockScreen
  - LockScreenManager
  - u.v.m.

API Choices for Managed Code

- Viele APIs der Windows Phone Runtime stellen neue Funktionalitäten bereit
- Andere APIs stellen die Kompatibilität zwischen native und managed Code sicher, diese enthalten äquivalente Funktionalitäten wie die .NET APIs

.NET API	Windows Phone Runtime API
System.IO.IsolatedStorage	Windows.Storage
System.NET.Sockets	Windows.Networking.Sockets
System.Threading.ThreadPool	Windows.System.Threading.ThreadPool
Microsoft.Devices.Sensors	Windows.Devices.Sensors

- Entwickler von managed Code können beide gleichwertige APIs nutzen.
  - Programmcode soll vor allem für WP7.* und WP8.0 kompatibel sein → **.NET API**
  - Programmcode soll zw. WP8.0 & Win8 geteilt werden→ **Windows Phone Runtime API**

Win32 and COM API

- Zusätzlich zu .NET und Windows Phone Runtime, gibt es die Win32 API
  - Zugriff auf die WinSock-Lib für low-level Netzwerkprogrammierung
  - Kamera APIs für native Code Apps
  - COM APIs z.B. CoInitializeEx, CoTaskMemAlloc, CoTaskMemFree, CreateFile2, ReadFile, WriteFile, HeapAlloc, CreateMutexExW, WaitForSingleObjectW, u.v.m.
- Hauptsächlich für Entwickler von nativem C / C++ Code interessant
- Managed Anwendungen können diese aufrufen, indem ein natives Code Projekt in die Solution eingefügt und darüber auf diese APIs zugegriffen wird (selten erforderlich)
- **Wird in diesem Workshop nicht weiter behandelt!**

Emulator

- Unterstützt verschiedene Auflösungen.
  - Simulation für Lagesensor
  - Simulation für GPS-Position
  - Screenshot
- Der Windows Phone 8 Emulator läuft mit Windows Hyper-V
- Der Windows Phone 8 Emulator verhält sich wie ein Eigenständiges Gerät innerhalb des LANs.

Emulator Voraussetzungen

- Hardware:
  - DirectX 10 or later graphics card
  - 4 GB or more of RAM
  - 64-bit processor
  - Hardware-assisted virtualization supported **Second Level Address Translation (SLAT)** supported by the BIOS
  - Hardware-based Data Execution Prevention (DEP) supported
- Software:
  - 64-bit version of Windows 8 Pro or higher

Changes in WP8 Emulator

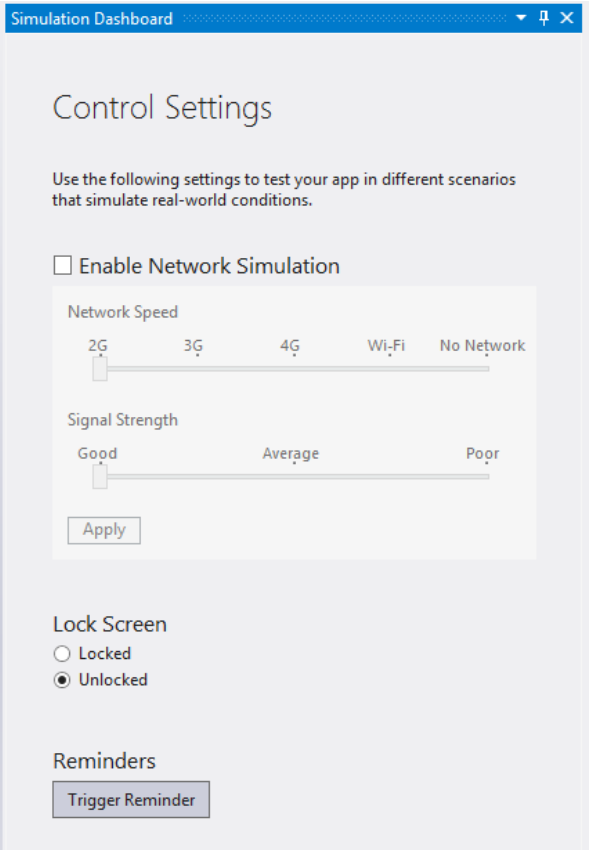
BASTA!

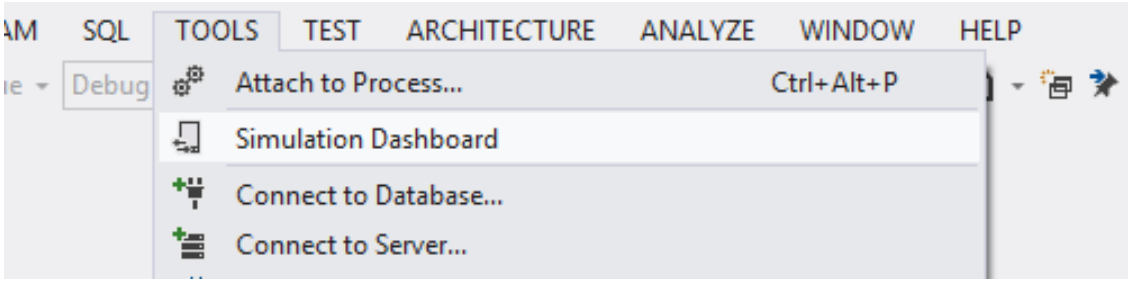
Emulator feature	WP 7.1 Emulator	WP8 Emulator	Effects
Graphics rendering	Uses the computer's hardware graphics card.	Uses software emulation.	Graphics in WP 8 Emulator could be either faster or slower than on the actual phone.
Networking	Uses the Windows network connection.	Connects directly to the network as a separate device.	WP8 Emulator is connected directly to the network with its own IP address. Depending on your firewall or proxy settings, the emulator might not be able to reach some network destinations.
Sample photos	The media library was pre-populated with sample photos.	Sample photos are added to the media library when you open the Photos Hub for the first time in the emulator.	If you want to test an app that uses the photo chooser task, or an app that uses the MediaLibrary class, open the Photos Hub to populate an album with sample photos before you test your app.

New features of VS 2012

- Debugging native code
- Attached debugger
  - Debugging apps launched from a tile or notification.
- Manifest designer
- Isolated Storage Explorer
- Project templates for Direct3D apps
- Enhanced localization support in project templates.
- XNA Framework support
- DirectX support
- App monitoring and profiling.
- Simulation Dashboard.

Simulation Dashboard





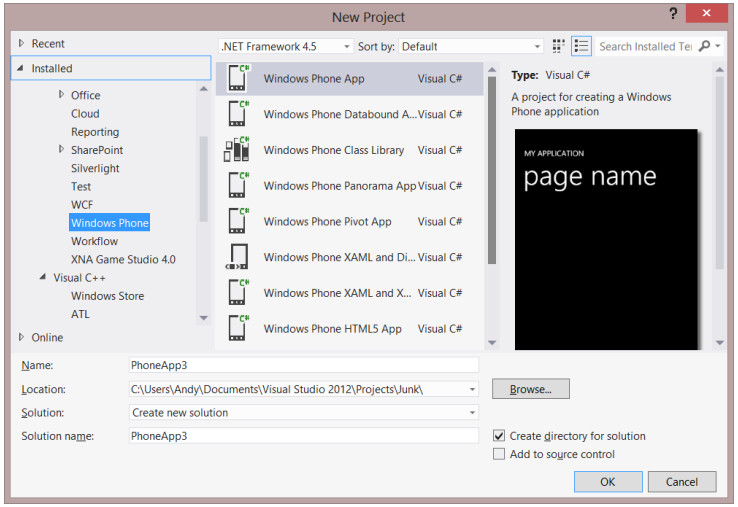
- Your app's slow startup time
- Slow response time to input, such as scrolling or zooming
- High battery drain
- Network latency
- High cost of network data
- Poor performance as the quality of the network signal changes
- Out of memory errors caused by high resource usage

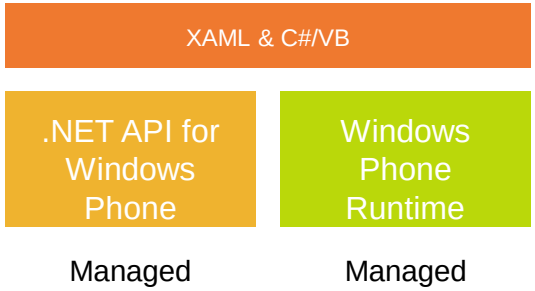
APP MODELS

Windows Phone 8 offers many additional ways of building apps compared to Windows Phone OS 7.1

XAML UI Mit Managed Code

- Am häufigsten verwendeter Weg eine Windows Phone App zu erstellen
- Die UI wird mit XAML und die
- Logik in C# oder Visual Basic .NET entwickelt
- Zugriff auf die .NET APIs und die Windows Phone Runtime APIs

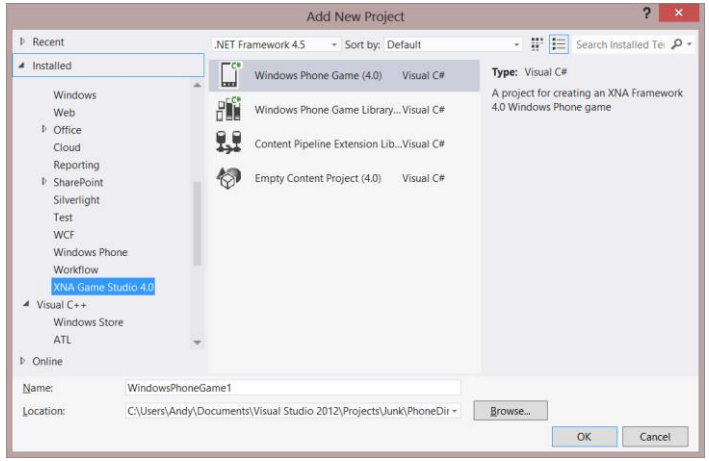


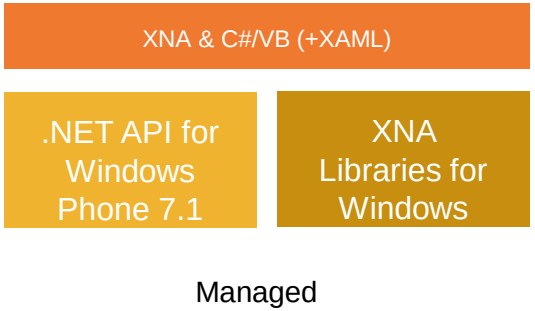


Demo 1: XAML and Managed Code

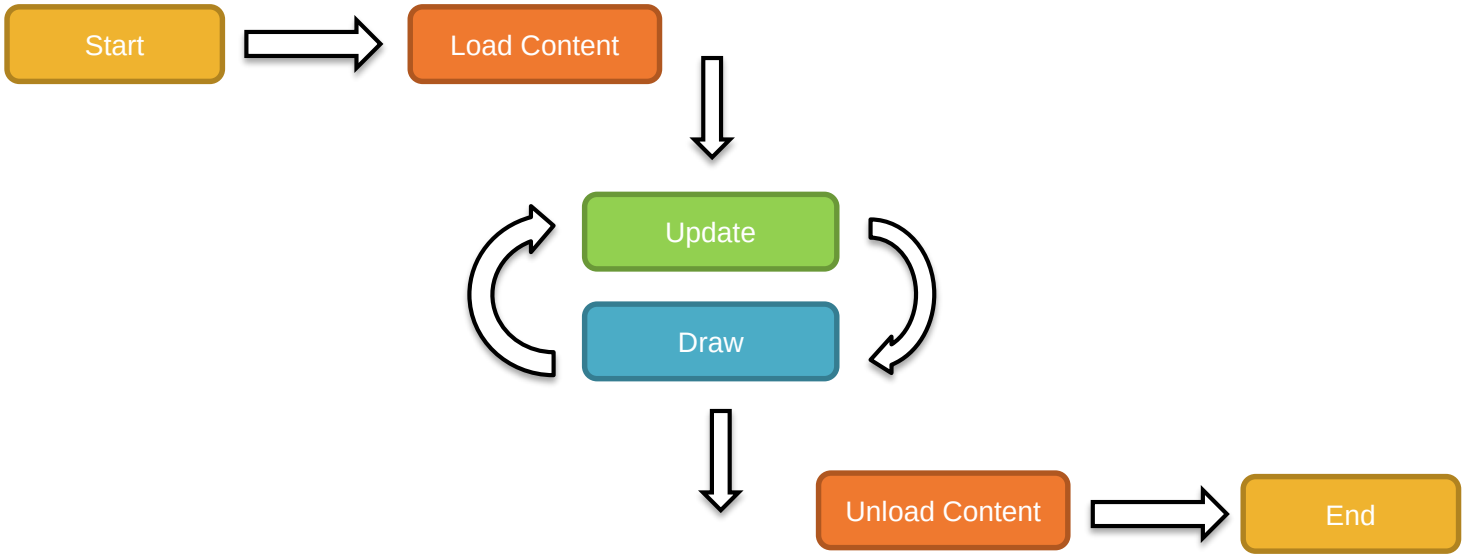
XNA Games using Managed Code

- Entwickeln von Spielen für Windows Phone unter Verwendung des XNA Frameworks
- Die gleichen Funktionen wie in Windows Phone OS 7.X
- Logik wird in C# oder Visual Basic .NET entwickelt
- Zugriff auf die WP 7.X APIs
- KEIN Zugriff auf die Windows Phone 8 APIs
- Verwendet die gleiche Projekt-Vorlage wie Visual Studio 2010





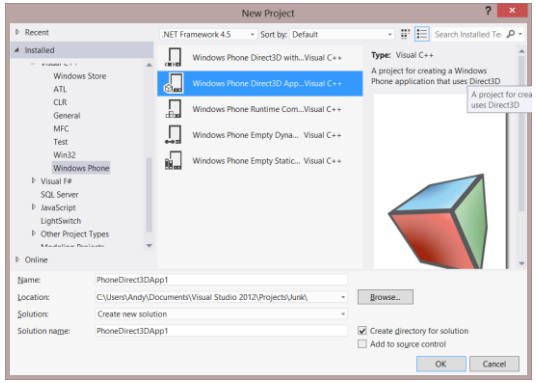
- Wichtige Methoden in der GAME Klasse
  - Initialisierung
    - `Initalize()` -> Initialisiert alle Elemente (z.B. StartPositionen)
    - `LoadContent()` -> Lädt Resourcen (z.B. Images, Sounds usw.)
  - `Hauptschleife`
    - `Update()` -> Aktualisiert die Spielewelt
    - `Draw()` -> zeichnet die Spielewelt

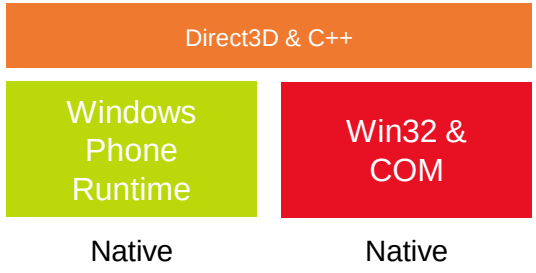


Demo 2: XNA and Managed Code

Direct3D App

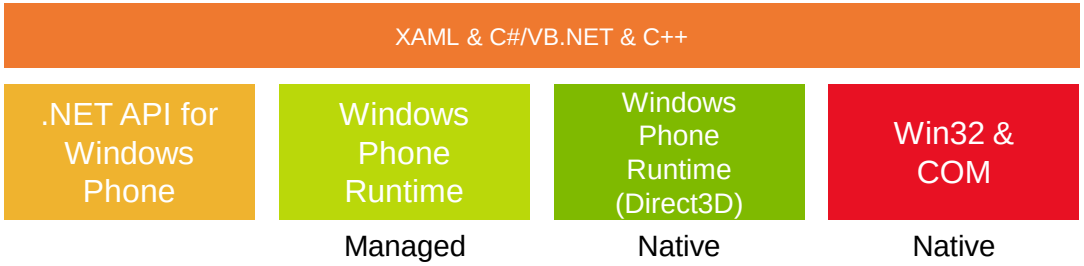
- Direct3D Apps komplett in C++ nativem Code,
- Verwendet nur **Direct3D** für die UI
- Vor allem für **Games development**
  - große Codeteile mit dem PC geschared
  - Erleichtert das gemeinsame Nutzen von Komponenten in C++
  - z.B. compute engines, graphic libraries
  - and API sets
- Zugriff auf die Windows Phone Runtime APIs





Direct3D and XAML

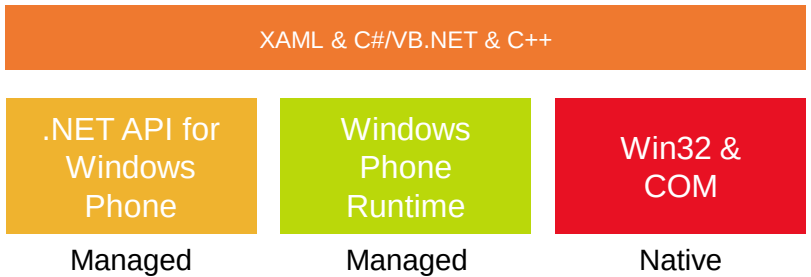
- Kombination aus allen Technologien
- Oberfläche ist teilweise in XAML und Teilweise mit DirectX entwickelt
- Programmcode besteht teilweise aus C# und aus C++
  - Wird verwendet, wenn in einer XAML Anwendung anspruchsvolle Grafik dargestellt werden soll.
- Eigene Projektvorlage: **Windows Phone Direct3D with XAML App**
  - Diese ist in Visual C#, Visual Basic und Visual C++ in dem **Add New Project Dialog** verfügbar



Demo 3: Direct3D Games

XAML/Managed plus Native Code

- Managed Apps können auch mit nativen Bibliotheken interagieren
  - Hinzufügen einer C++ Dynamic Link Library oder eines Windows Phone Runtime Component Projekts zu einer managed XAML Projektmappe
- Win32 API stellt unter anderem Funktionen der Bibliotheken:
  - Winsock, File I/O, usw.
- Einfache Möglichkeit C/C++ Code für Windows Phone Plattform zu portieren
- Rechenintensive Vorgänge können explizit in nativen Code ausgelagert werden
  - z.B. Bildbearbeitung, Berechnungsmodule allgemein, Dokumentenerstellung…

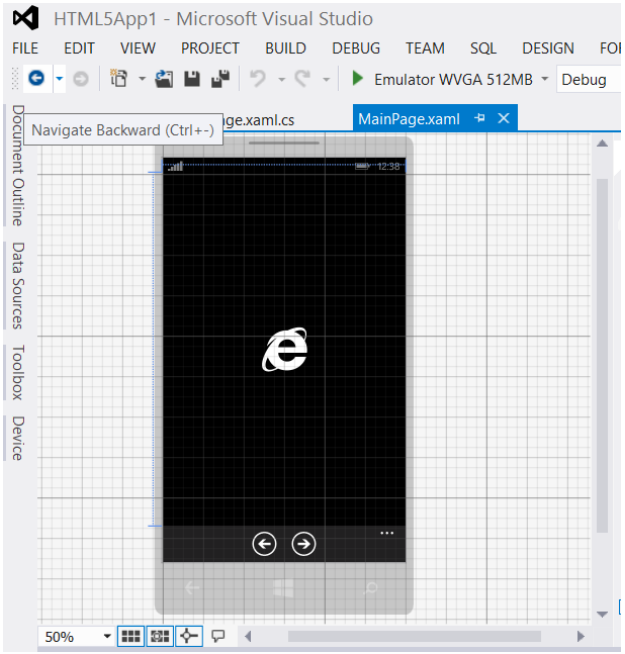


Demo 4: Managed and Native Interop

- Native HTML5/JavaScript Entwicklung wird unter Windows Phone 8 (noch ?) **nicht unterstützt**.
  - Windows Phone Runtime nicht verwendbar
- Trotzdem: Windows Phone 8 hat den Internet Explorer 10
  - Gleiche Version des IE wie unter Windows 8
  - Sehr gute HTML5 Unterstützung

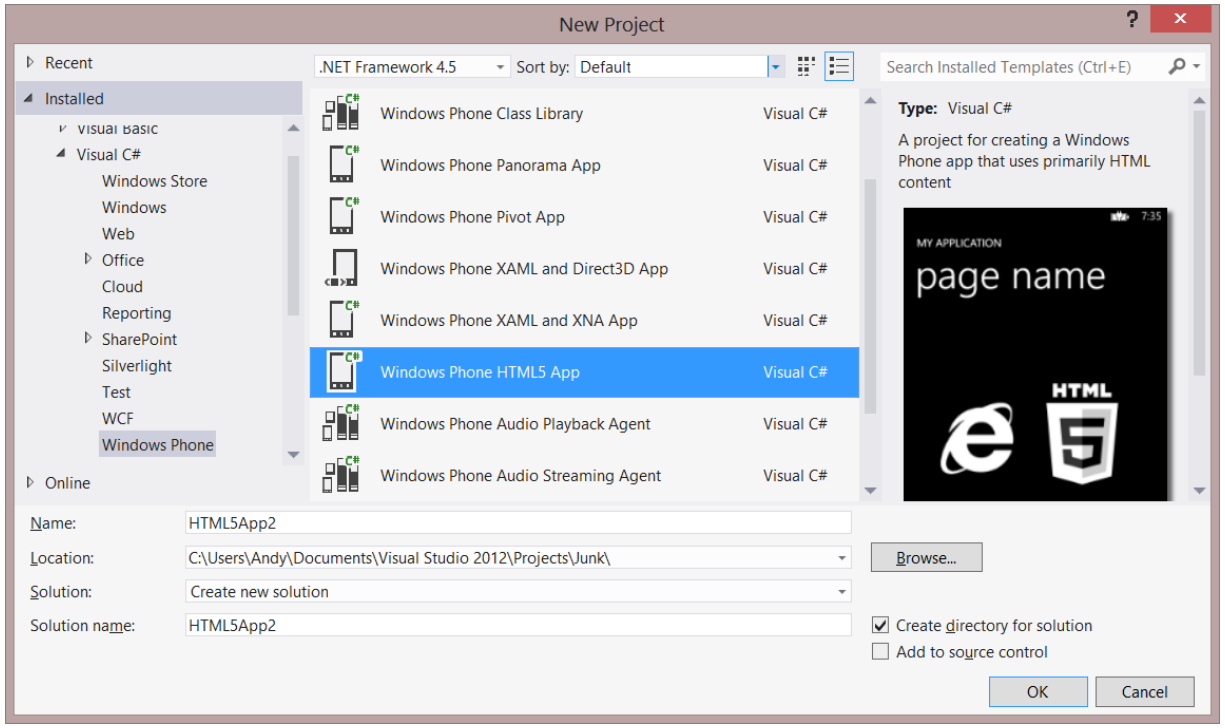
    (2 x feature support compared to WP 7.X)
  - Inklusive der neuen JavaScript Engine

    (4 x faster than Windows Phone 7.X)



- Dieser Browser steht innerhalb des Browser-Steuer-elementes zu Verfügung
  - Hiermit lassen sich HTML5-basierende Anwendungen innerhalb des WebBrowsers entwickeln
  - Inhalte können lokal oder auf einem WebServer liegen

HTML5 App Project Template



Demo 5: Managed App Displaying HTML 5 Content

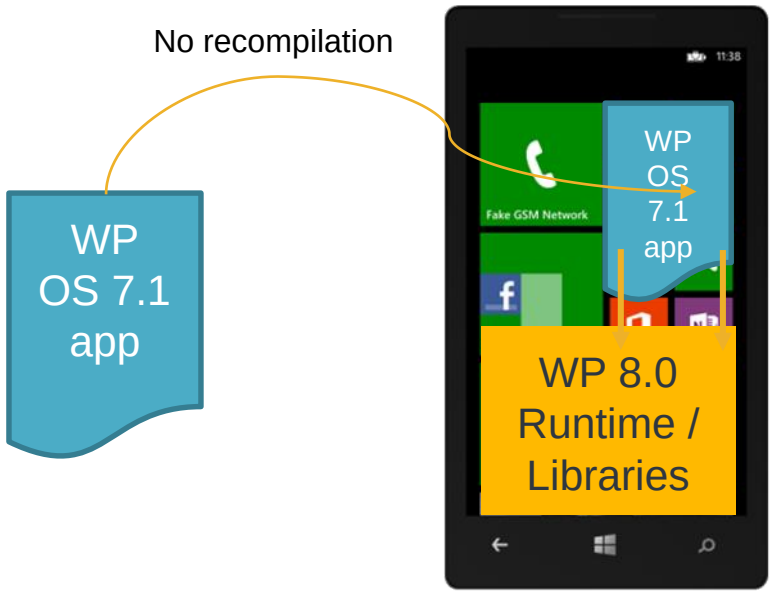
WINDOWS PHONE 7.X APP COMPATIBILITY

Laufen WP 7.X APPs auf WP 8.0 ?

Was ist dabei zu beachten ?

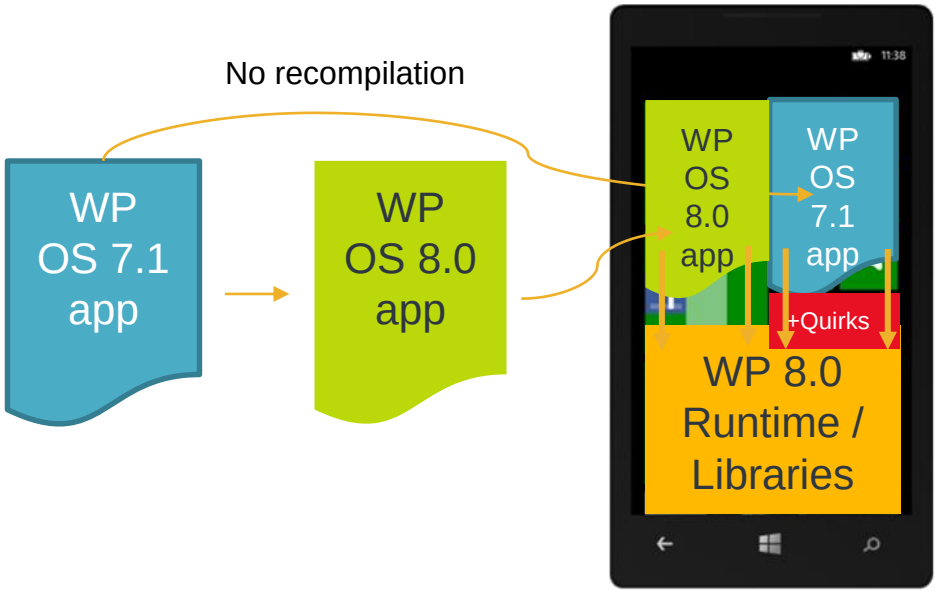
Windows Phone 7.1 / Windows Phone 8

- Grundsätzlich bietet die neue Windows Phone Plattform die Möglichkeit Windows Phone OS 7.X App ohne Neu-Übersetzung oder Anpassung laufen zu lassen
- Die API ist prinzipiell gleich, bis auf geringe Verhaltensunterschiede zwischen WP8.0 und WP7.X
  - Feature improvements
  - Behavior changes



How does it work ? WP 7.X Apps on WP 8

- Speziell für WP 7.X Apps wurde eine **quirks shim** eingeführt, welche die WP 7.X API auf die WP 8.0 API abbildet.
- Apps welche ge-upgraded wurden auf WP 8.0 (recompiled) laufen ohne die quirks shim



Upgrade von WP 7.X auf WP 8.0

- Stellen Sie bei der Aktualisierung von WP7.1 auf WP 8.0 sicher, dass Ihr Code nicht auf Verhalten der Version 7.X aufbaut, Bedenken Sie folgende zwei Szenarien:
  - **Quellcode-Inkompatibilität** – Nach der Neuübersetzung des gleichen Quellcodes mit WP8 verhallt sich dieser Quellcode anders als zuvor unter WP 7.X
  - **Beispiel 1**: Assembly.GetType sucht unter WP 7.X in der mscorlib.dll sowie in der Assembly aus der diese Funktion aufgerufen wurde. Im Gegensatz dazu sucht die WP8 Laufzeit NUR in der Assembly aus der diese Funktion aufgerufen wurde.
  - **Binäre-Inkompatibilität** – mache Verhalten können nicht „quirked“ werden, so kann es sein, dass eine App sich zur Laufzeit anders verhallt als zuvor unter WP 7.5 ausgetestet.
  - **Beispiel 2**: Der Garbage-Collector unter WP8 ist komplett anders implementiert als unter WP 7.1. Sollte Ihre App davon abhängen, dass Objekte in einer festen Reihenfolge finalisiert werden kann dies zu Problemen führen.

Source Code Incompatibility Examples

BASTA!

- Programmcode der sich beim Übersetzen andersverhällt zwischen WP7 und WP8

Item	Windows Phone OS 8.0	Windows Phone OS 7.1
IsolatedStorageFile.FileExists(String) method	If passed a null, the method throws an ArgumentNullException exception.	The method returns false.
Mutex class	Mutex names cannot include a backslash character	A backslash character is allowed in a mutex name and is replaced with another character at run time
BeginRead , BeginWrite, EndRead, EndWrite, and subclassed methods	Input/output operations are performed asynchronously	Input/output operations are performed synchronously
Thread.CurrentCulture and Thread.CurrentUICulture properties	Changes to the current culture and current UI culture affect only the current thread	Changes to the current culture and the current UI culture affect all app threads
XmlSerializer class	Serialized types must have a default (parameterless) constructor	It is not necessary for serialized types to have a default constructor
….many more…. !	See the topic **Windows Phone app platform compatibility** in the documentation	

Binary App Incompatibility Examples

- Programmcode der sich auf nach dem Überstzen auf WP7 anders verhällt als auf WP8

Item	Windows Phone OS 8.0 runtime behavior change	Impact on code that assumes Windows Phone OS 7.1 behavior
Background File Transfers	The limit on the number of concurrent file transfers has increased from 5 to 25	Code that assumed that the maximum number of transfers cannot exceed 5 will fail
Networking	Because the Windows Phone 8 Developer Preview client can handle the Vary header and cache responses, a Web service call may complete much faster than in previous versions	If you are making a Web service call your code must not rely on the download taking more than a second. When checking the response time it could be less than 1 second
Access to private nested classes	Windows Phone OS 7.1 allows a class to access its private nested classes; Windows Phone 8 Developer Preview does not	Access to private nested classes is unsupported
….many more…. !	See the topic **Windows Phone app platform compatibility** in the documentation	

MEINE ERSTE WINDOWS PHONE 8 APP

Anwendungsvorlagen

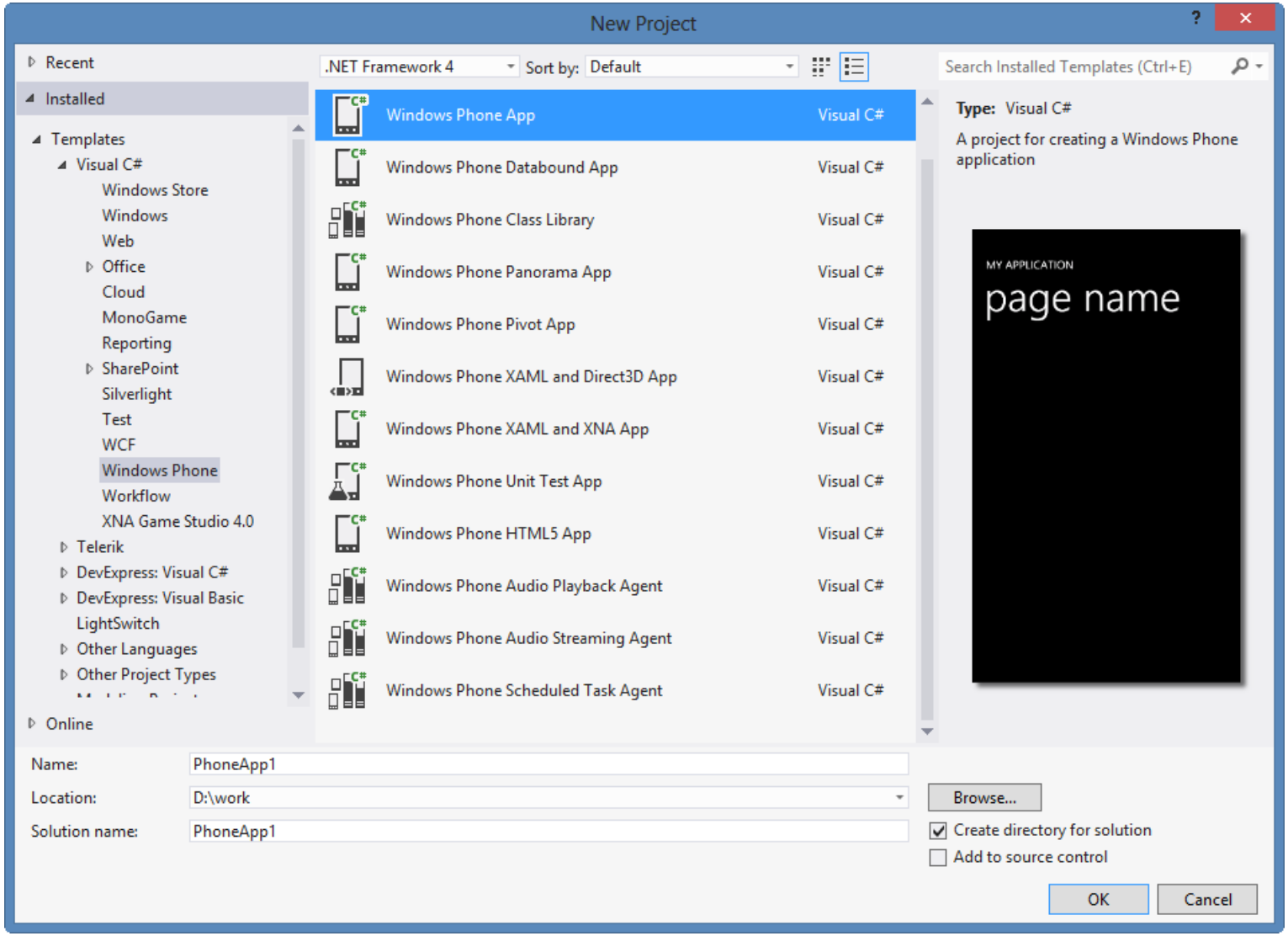
- Drei Vorlagen für .NET für Windows Phone Apps

Windows Phone Anwendung

Windows Phone Pivot Anwendung

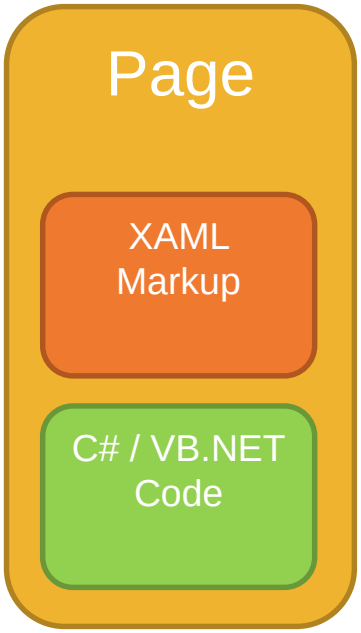
Windows Phone Panorama Anwendung

Visual Studio 2012 Projektvorlgen



Silverlight – Code Behind

- .NET für Windows Phone trennt den **programm code** von dem **markup code**
  - Der markup code wird in XAML formatiert angegeben, beschreibt das Design der Seite und enthält Steuerelemente (TextBox, Button, usw.)
  - Der programm code wird als C# oder VB.NET code in der code behind Datei angeben, enthält Funktionen, Ereignishandler usw.
  - Die Klasse wird von der Basisklasse Page abgeleitet und durch zusammenfügen der partiellen Anteile aus XAML und C#/VB.NET code gebildet

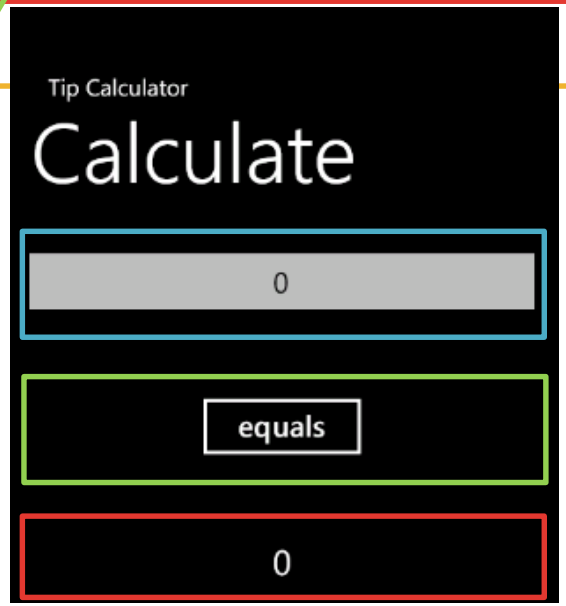


XAML für TipCalculator APP

```
 <!--ContentPanel - place additional content here-->
<Grid x:Name="ContentGrid" Grid.Row="1">
    <TextBox Height="72" HorizontalAlignment="Left" Margin="8,19,0,0" Name="firstNumberTextBox"
             Text="0" VerticalAlignment="Top" Width="460" TextAlignment="Center" />
    <Button Content="equals" Height="72" HorizontalAlignment="Left" Margin="158,144,0,0„
            Name="calcButton" VerticalAlignment="Top" Width="160" Click="calcButton_Click" />
    <TextBlock Height="46" HorizontalAlignment="Left" Margin="158,276,0,0"
               Name="resultTextBlock" Text="0" VerticalAlignment="Top" FontSize="30"
               Width="160" TextAlignment="Center" />
</Grid>
```

`calcButton` ist der `Name` unter dem der Button aus dem Programmcode erreichbar ist

`Click` ist der Ereignisname des mit der Methode `calcButton_Click` zu verbindenen Ereignisses



CodeBehind für TipCalculator APP

- Button im XAML

```
<Button Content="equals" Height="72" HorizontalAlignment="Left" Margin="158,144,0,0"
        Name="calcButton" VerticalAlignment="Top" Width="160" Click="calcButton_Click" />
```

- Ereignishandler im C# Code

```
public partial class MainPage : PhoneApplicationPage {
 // Constructor
 public MainPage() {
    InitializeComponent();
 }

 private void calcButton_Click(object sender, RoutedEventArgs e) {
   resultTextBlock.Text = (decimal.Parse(firstNumberTextBox.Text)*0.10m).ToString();
 }
}
```

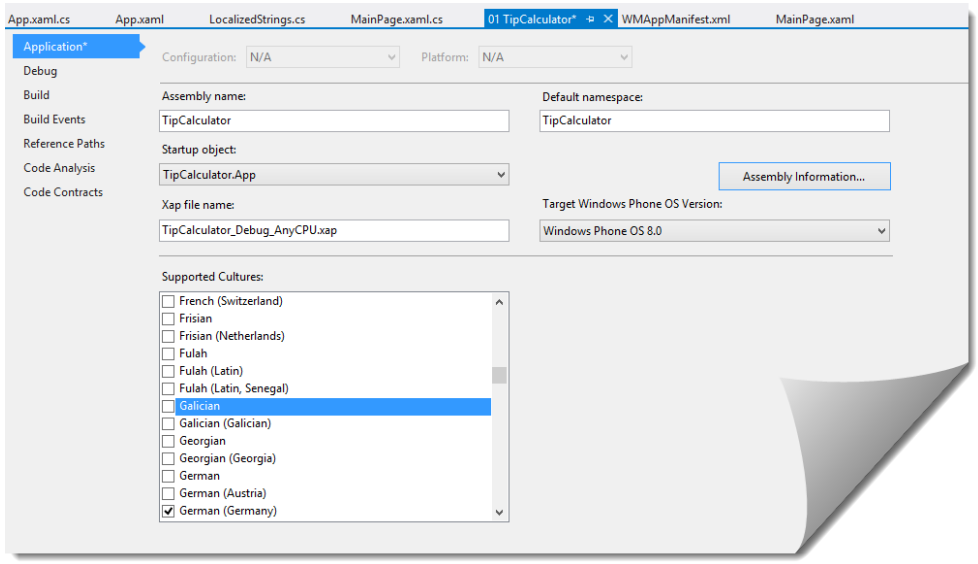
- TextBlock im XAML

```
<TextBlock Name="resultTextBlock" Text="0" VerticalAlignment="Top" FontSize="30"
           Height="46" HorizontalAlignment="Left" Margin="158,276,0,0"
           Width="160" TextAlignment="Center" />
```

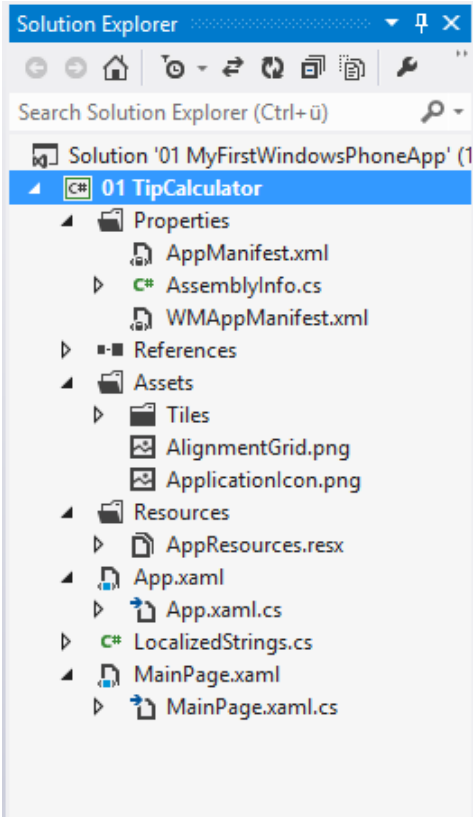
- Erstellen der TipCalculator App
  - Rechner für „Trinkgeld“ mit einem oder mehreren Eingabe-feldern erstellen, Anwendung starten
- Emulator verwenden

Projektaufbau

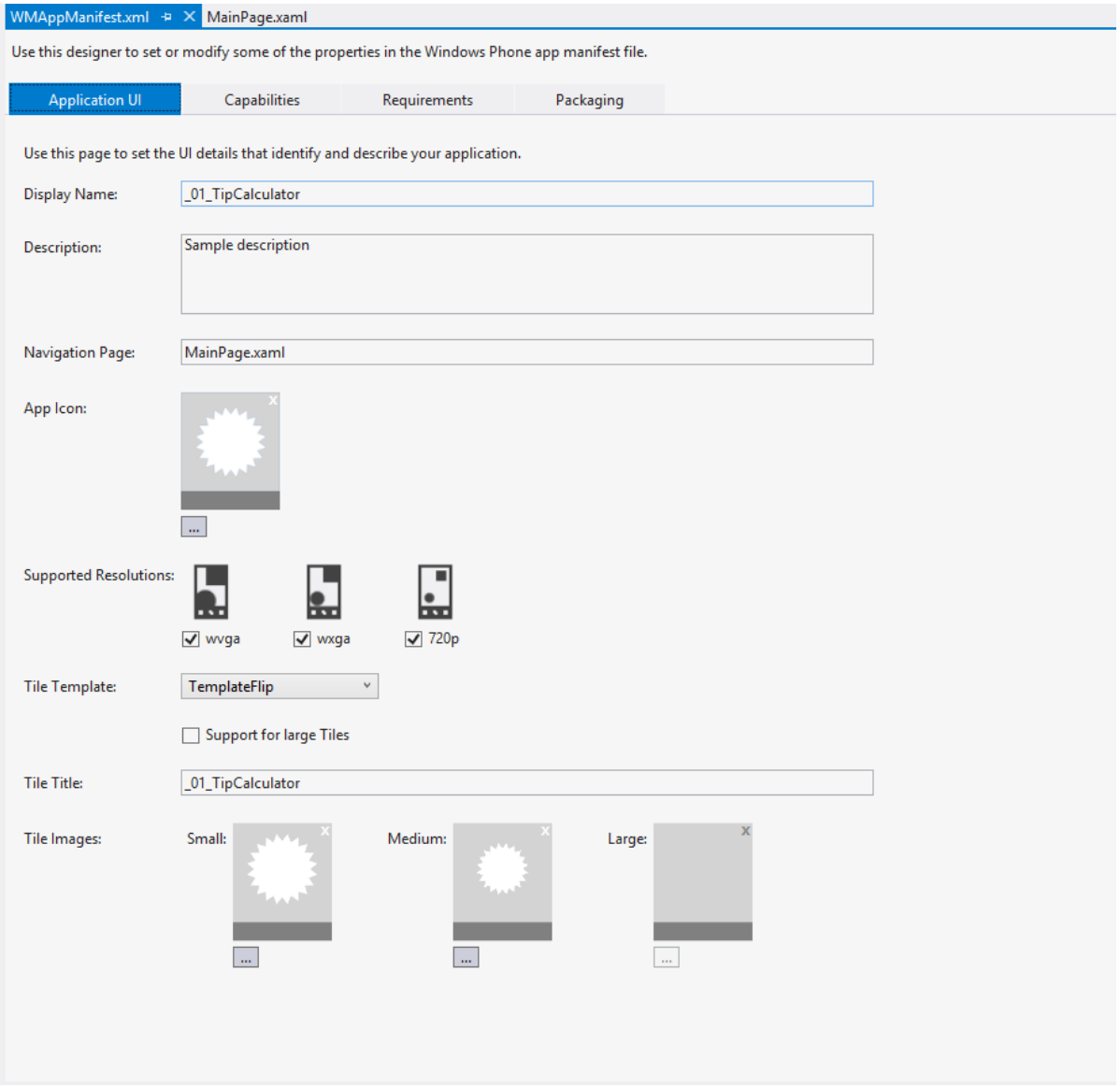
- Projektmappe
  - Projekt
    - Properties



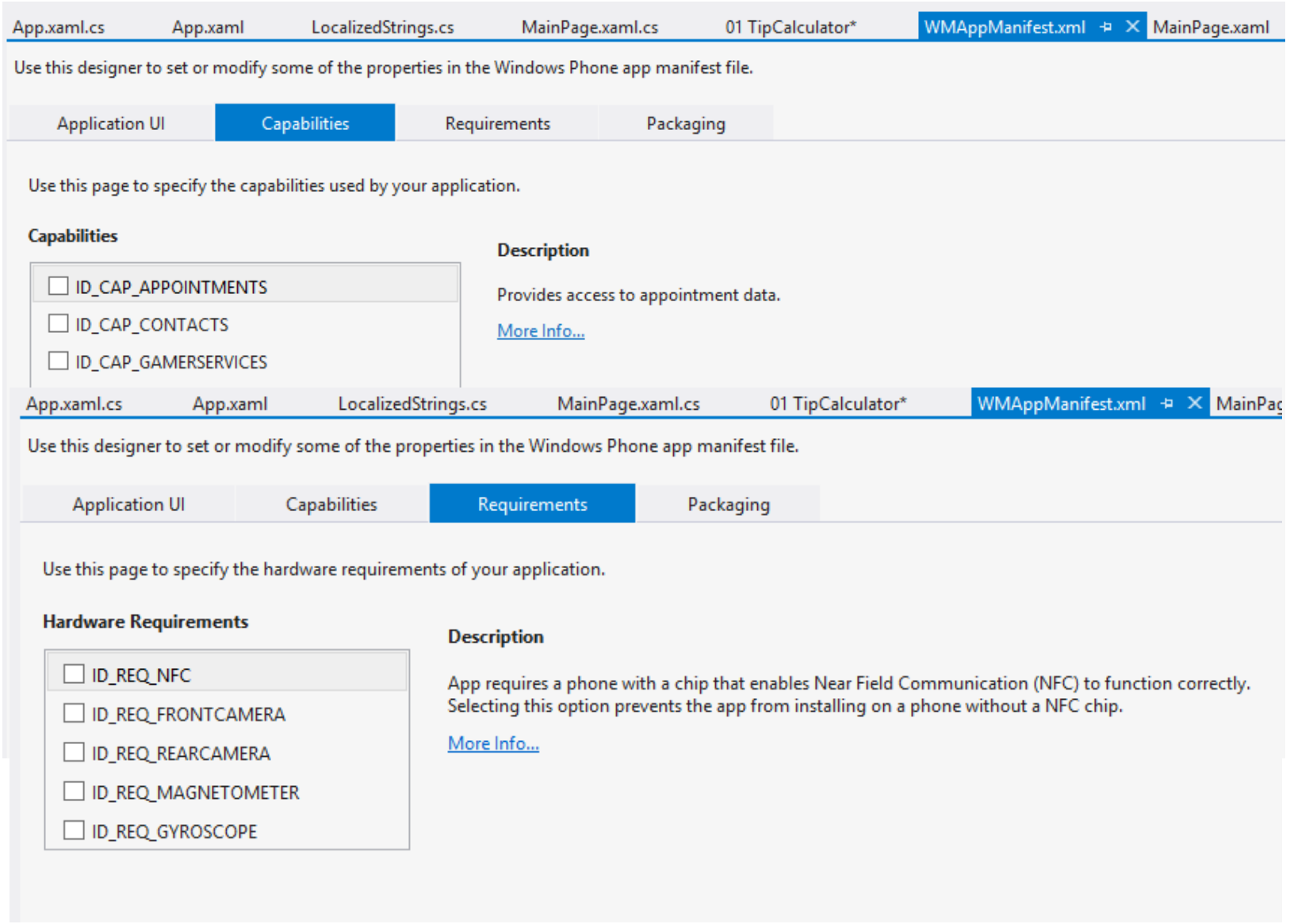
    - WMAppMainfest
      - Eigenschaften der Anwendung
  - Anwendungsklasse (App.xaml)
  - Seiten (MainPage.xaml)
  - Ressourcen
  - Quellcode



WMAppManifest

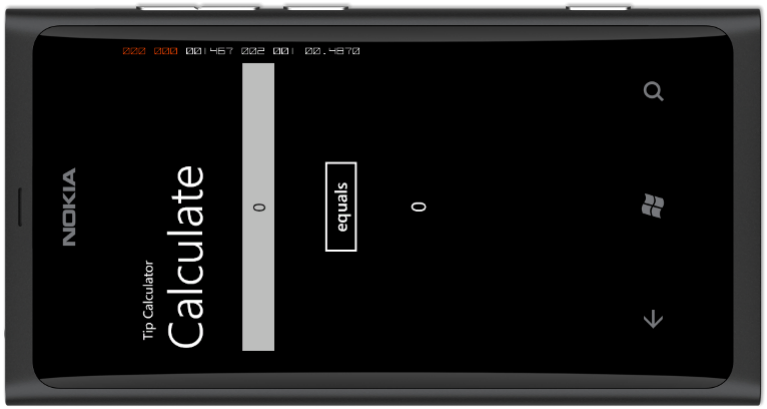


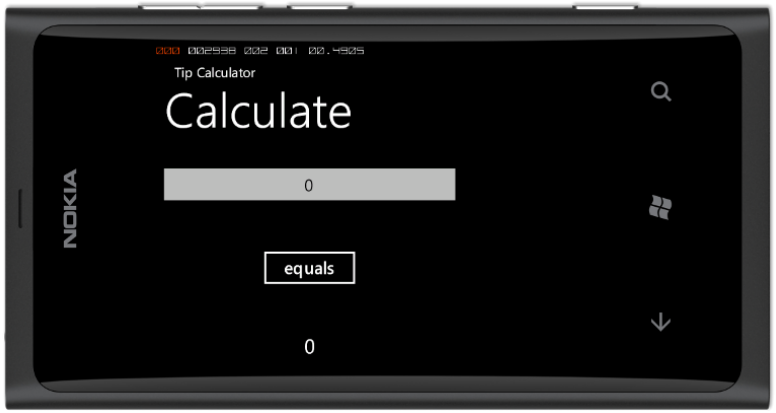
WMAppManifest



Landscape Mode

- Die Standardeinstellung für die Orientierung einer Anwendung ist **Portrait**, wenn der **LandscapeMode** unterstützt werden soll, muss diese explicit mit der entsprechenden Eigenschaft der Page zugelassen werden.






```
SupportedOrientations="Portrait"
```

```
SupportedOrientations="PortraitOrLandscape"
```

Fehlerbehandlung

- Benutzer können in der aktuellen Version alle Daten eingeben, z.B. Texte, welche sich nicht in dezimal-Werte konvertieren lassen

  - **Schritt 1)** Die Konvertierung mit einem Try-Catch-Block absichern

```
try {
  resultTextBlock.Text =
     (decimal.Parse(firstNumberTextBox.Text) * 0.10m).ToString();
} catch {
  MessageBox.Show("Falsches Zahlenformat")
}
```

  - **Schritt 2)** Den Eingabebereich der Tastatur begrenzen

```
<TextBox InputScope="Number"  ... />
```

- Jede Eingabe sollte immer auf Richtigkeit überprüft werden, vor der Verarbeitung
- Der Eingabebereich sollte nur Eingaben zulassen, welche in dem aktuellen Kontext gültig sind (z.B. keine Buchstaben in einem Zahlenfeld)
- Gebräuchliche Eingabebereichsbegrenzungen sind:

Bezeichner	Funktion
Url	Webadressen
FullFilePath	Dateinamen und -Pfade
MailUserName	eMail-Adressen
PostalCode	Postleitzahlen
Digits	Ziffern
Numbers	Zahlen
TelephoneNumber	Telefonnummern

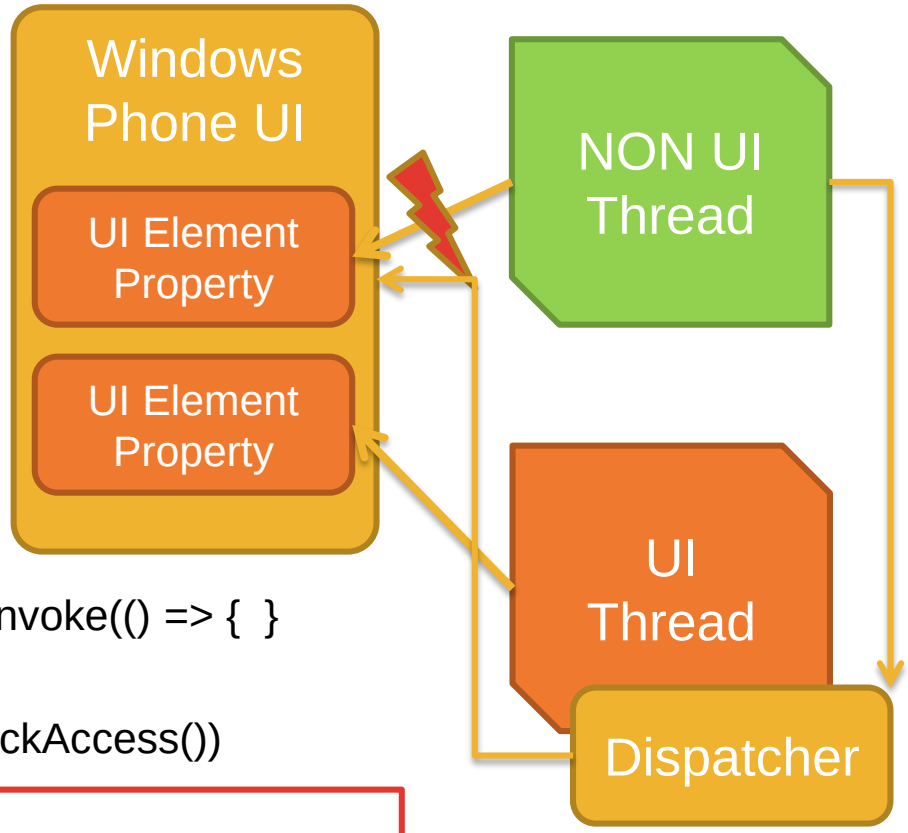
http://msdn.microsoft.com/en-us/library/system.windows.input.inputscopenamevalue(v=vs.95).aspx

WINDOWS PHONE APIS

Dispatcher

- Wird verwendet wenn ein NICHT UI Thread Änderungen an einem Element der UI Vornehmen soll

```
private void button1_Click(object sender, RoutedEventArgs e) {
  var t = new Thread(MyWorker);
  t.Start();
}

private void MyWorker() {
  button1.IsEnabled = false;
  Thread.Sleep(2000);
  button1.IsEnabled = true;
}
```

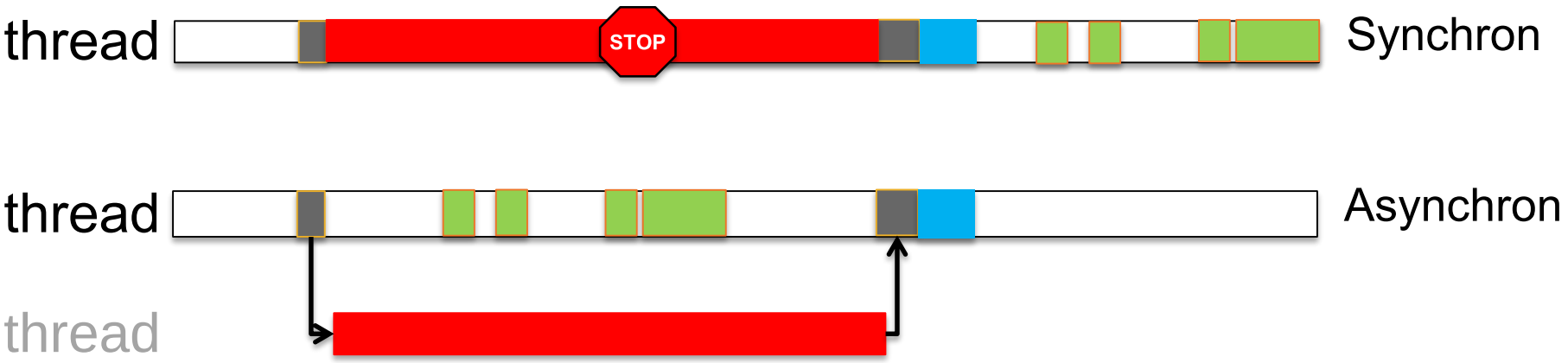


- Wird aufgerufen mit
  - Deployment.Current.Dispatcher.BeginInvoke(() => { })
- ggf. testen mit
  - if (Deployment.Current.Dispatcher.CheckAccess())

```
private void MyWorker() {
  Deployment.Current.Dispatcher.BeginInvoke(() => { button1.IsEnabled = false; };
  Thread.Sleep(2000);
  Deployment.Current.Dispatcher.BeginInvoke(() => { button1.IsEnabled = true; };
}
```

Asynchrone Funktionen
Was ist das?

- Sind Nebenläufig
- meist Ereignisorientiert
- Nutzen mehrere Prozessorkerne / Threads
- Entwickler startet asynchrone Verarbeitung unter Angabe einer Ruckruffunktion durch Aufrufen der Start-Funktion. Stoppen der Verarbeitung ist oft mit Hilfe eine Ende-Funktion möglich.



Asynchrone Funktion

- Asynchrones laden einer WebAPI, mit Hilfe eines WebRequests
  - Vorteil: UI Thread Blockiert nicht
  - Nachteil: Viele verschachtelte Rückruffunktionen bei komplexerer Logik

```
private void useAsyncRes_Click(object sender, RoutedEventArgs e) {
   var webRequest = (HttpWebRequest)WebRequest.Create("http://api.open... ");
   webRequest.Method = "GET";
   webRequest.BeginGetResponse(new AsyncCallback(OnGotWebRequest), webRequest);
}
```

UI Thread

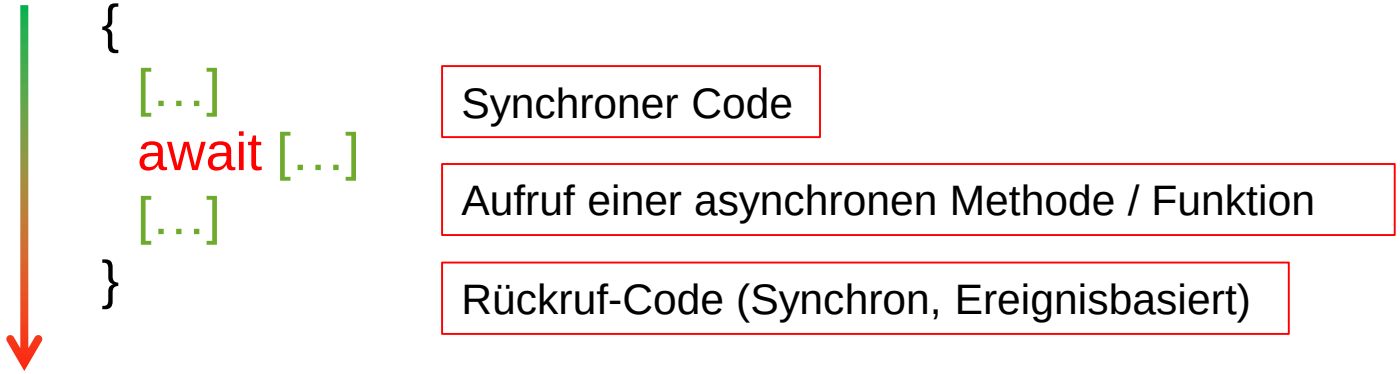
```
private void OnGotWebRequest(IAsyncResult asyncResult) {
  var webRequest = (HttpWebRequest)asyncResult.AsyncState;
  var httpResponse = (HttpWebResponse)webRequest
        .EndGetResponse(asyncResult);

  using (var streamReader = new StreamReader(httpResponse.GetResponseStream())) {
    string responseText = streamReader.ReadToEnd();
    Dispatcher.BeginInvoke(delegate {
      result.Text = responseText;
    });
  }
}
```

UI Thread

Worker Thread

Wie heute?

- Lineare Programmlogik wird mit zusätzlichen Schlüsselworten versehen
  - private async Task BerechneAsync()



- Der Compiler generiert zusätzliche Ablauf-Logik.
- Rückgabe mit Task<T> möglich
- Wechsel von synchron in Asynchron mit:
  - private async void MeineFunktion(…) { await … }

Async / Await Aufruf

- Asynchroner Funktionsaufruf mit Hilfe von **Async / Await**
  - Vorteil : UI Thread wird nicht blockiert
  - Vorteil: nach Gewöhnung an Syntax linearer Programmablauf auch bei komplexer Logik

- ThreadMarsheling erfolgt automatisch

```
private async void useAsyncAwait_Click(object sender, RoutedEventArgs e) {
  var webRequest = WebRequest.CreateHttp("http://api.open...de");
  var response = await webRequest.GetResponseAsync();
  using (var stream = response.GetResponseStream()) {
    using (var reader = new StreamReader(stream)) {
       string responseText = await reader.ReadToEndAsync();
       result.Text = responseText;
    }
  }
}
```

UI Thread

WorkerThread

UI Thread

IAsyncResult → async/await

- Verwendung der statischen Funktion
  `Task<TResult>.Factory.FromAsync<TParameter>(begin, end)`

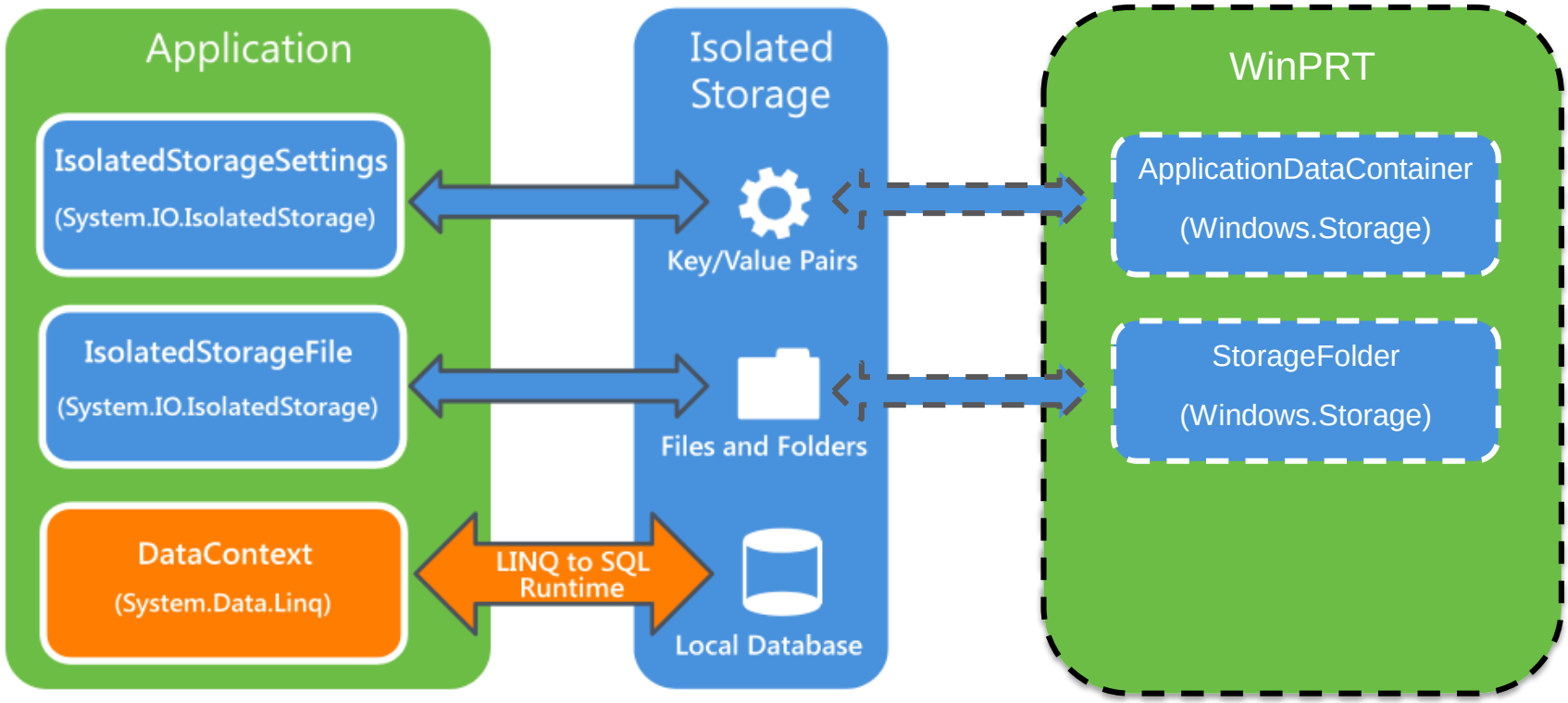
```
var response = await Task<WebResponse>.Factory.FromAsync(
  webRequest.BeginGetResponse,
  webRequest.EndGetResponse,
  null);
```

- Alternative: eigene Erweiterungsmethode & TaskCompletionSource

```
public static Task<HttpWebResponse> GetResponseAsync(this HttpWebRequest request) {
  var taskComplete = new TaskCompletionSource<HttpWebResponse>();
  request.BeginGetResponse(asyncResponse => {
    try {
      var responseRequest = (HttpWebRequest)asyncResponse.AsyncState;
      var someResponse = (HttpWebResponse)responseRequest
        .EndGetResponse(asyncResponse);
      taskComplete.TrySetResult(someResponse);
    } catch (WebException webExc) {
      var failedResponse = (HttpWebResponse)webExc.Response;
      taskComplete.TrySetResult(failedResponse);
    }
  }, request);
  return taskComplete.Task;
}
```

Datenpersistenz

- **Settings**: speichert Daten als key/value Paare in der **IsolatedStorageSettings** Klasse.
- **Files and folders**: Speichert Dateien und Ordner mit Hilfe der **IsolatedStorageFile** Klasse.
- **Relational data**: Speichert relationale Daten in einer lokalen Datenbank unter Verwendung von LINQ to SQL.



WP 7 & WP 8

Win8 & WP 8

z.Z. Nicht Implementiert

Applicationsettings WP7/WP8

- Schreiben von Anwendungseinstellungen

```
var settings = IsolatedStorageSettings.ApplicationSettings;
settings["LoggerState"] = LoggerStatus;
settings["TraceName"] = CurrentName;
settings["TraceId"] = CurrentTraceId;
```

- Lesen von Anwendungseinstellungen

```
var settings = IsolatedStorageSettings.ApplicationSettings;

string traceName;
TraceName = settings.TryGetValue("TraceName", out traceName) ? traceName : "";

Guid traceId;
CurrentTraceId = settings.TryGetValue("TraceId", out traceId) ? traceId : Guid.Empty;
```

- Siehe http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff769510(v=vs.105).aspx

Dateien und Ordner WP7/WP8

BASTA!

- Ordner und Datei anlegen

```
IsolatedStorageFile myStore = IsolatedStorageFile.GetUserStoreForApplication();
myStore.CreateDirectory("MyFolder");

using (var isoFileStream = new IsolatedStorageFileStream("MyFolder\\myFile.txt", FileMode.OpenOrCreate, myStore))
{
   using (var isoFileWriter = new StreamWriter(isoFileStream))
   {
      isoFileWriter.WriteLine(txtWrite.Text);
   }
}
```

- Ordner und Datei auslesen

```
IsolatedStorageFile myStore = IsolatedStorageFile.GetUserStoreForApplication();
try {
   using (var isoFileStream = new IsolatedStorageFileStream("MyFolder\\myFile.txt", FileMode.Open, myStore))
   {
      using (var isoFileReader = new StreamReader(isoFileStream))
      {
        txtRead.Text = isoFileReader.ReadLine();
      }
   }
}
catch
{
   txtRead.Text = "Bim Lesen ist ein Fehler aufgetreten";
}
```

Applicationsettings WP8/Win8

- Windows RT stellt lokale und geteilte Settings bereit
  - in WP 8.0 (noch) nicht implementiert

```
private static ApplicationDataContainer _local =
  ApplicationData.Current.LocalSettings;

private static ApplicationDataContainer _roaming =
  ApplicationData.Current.RoamingSettings;

public string LoggerName {
   get { return _local.Values["LoggerName"] as string ?? ""; }
   set { _local.Values["LoggerName"] = value;
}
```

- Diese WinRT API sollte immer dann zum Einsatz kommen, wenn die App gemeinsame Codeanteile mit Windows 8 enthält.

Dateien & Ordner WP8/Win8

- Für zukünftige Verwendung , in WP 8.0 (noch) nicht implementiert

```
 private static StorageFolder _local = ApplicationData.Current.LocalFolder;
 private static StorageFolder _roaming = ApplicationData.Current.RoamingFolder;

private async Task Save(string filename, byte[] data) {
  var roamingFile = await _roaming.CreateFileAsync(filename,
     CreationCollisionOption.ReplaceExisting);
  using (Stream stream = await roamingFile.OpenStreamForWriteAsync()) {
    await stream.WriteAsync(data, 0, data.Length);
  }
}

private async Task<byte[]> Load(string filename) {
  var file = await _roaming.GetFileAsync(filename);
  var prop = await file.GetBasicPropertiesAsync();
  var data = new byte[prop.Size];
  using (var roamingFile = await _roaming.OpenStreamForReadAsync(filename)) {
    await roamingFile.ReadAsync(data, 0, data.Length);
  }
  return data;
}
```

Launcher & Chooser

- **Launcher** - A “fire and forget” action, where a specific Windows Phone functionality is launched, for example, sending an SMS message, opening a webpage, or placing a phone call

- **Chooser** - An “open file dialog” action, where information is selected from a specific phone application’s storage area, for example, selecting an email address, contact, or picture

Launcher & Chooser

BASTA!

Launcher	Chooser
ConnectionSettingsTask (ab WP8)	EmailAddressChooserTask
EmailComposeTask (ab WP8)	PhoneNumberChooserTask
MarketplaceDetailTask	CameraCaptureTask
MarketplaceHubTask	PhotoChooserTask
MarketplaceReviewTask	Contacts (ab WP8)
MarketplaceSearchTask	Appointments (ab WP8)
MediaPlayerLauncher	
PhoneCallTask	
SaveContactTask (ab WP8)	
SaveEmailAddressTask (ab WP8)	
SavePhoneNumberTask (ab WP8)	
SearchTask	
ShareLinkTask (ab WP8)	
ShareStatusTask (ab WP8)	
SmsComposeTask	
WebBrowserTask	

Übung

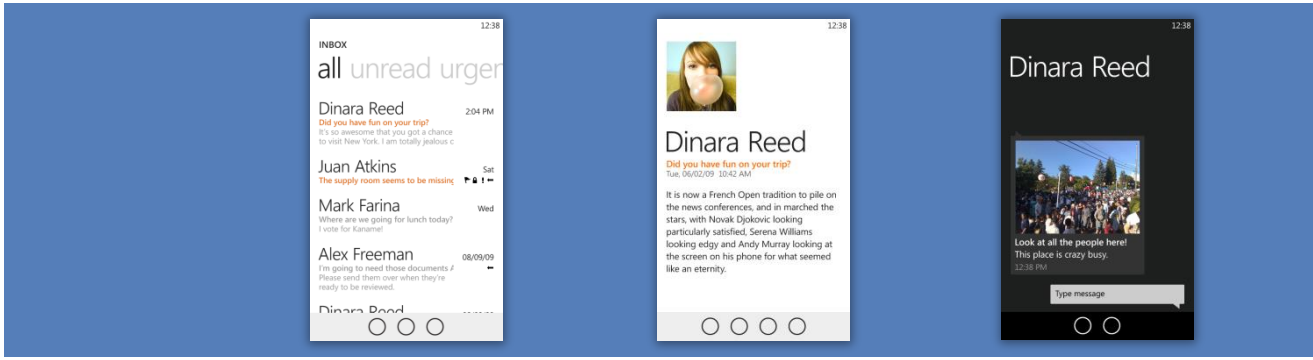
- Erstellen einer Anwendung die einen Kontakt auswählt und die Möglichkeit bietet, diesen Anzurufen, eine eMail oder eine SMS zu senden.
- Alternativ: eine App, welche einen neuen Status postet

NAVIGATION

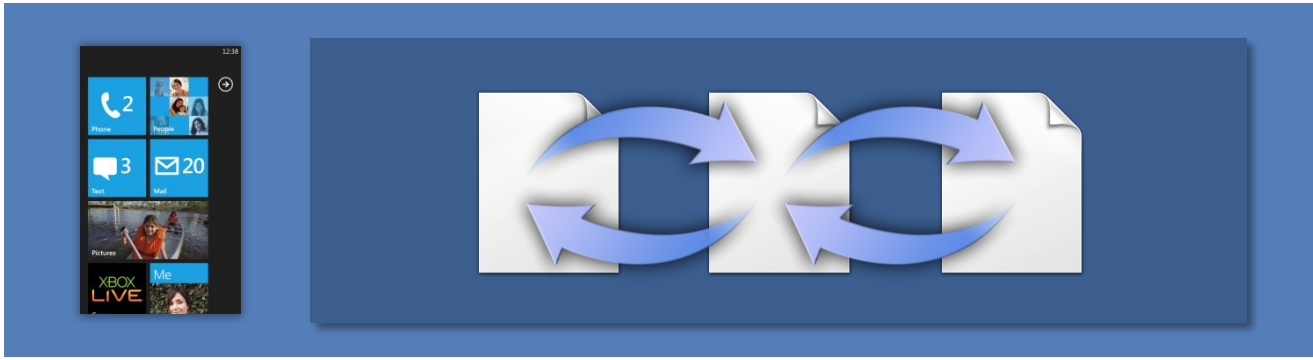
UI Model Concepts

Application
UI and logic for functionality exposed through pages

Page
A single screen of user interaction elements

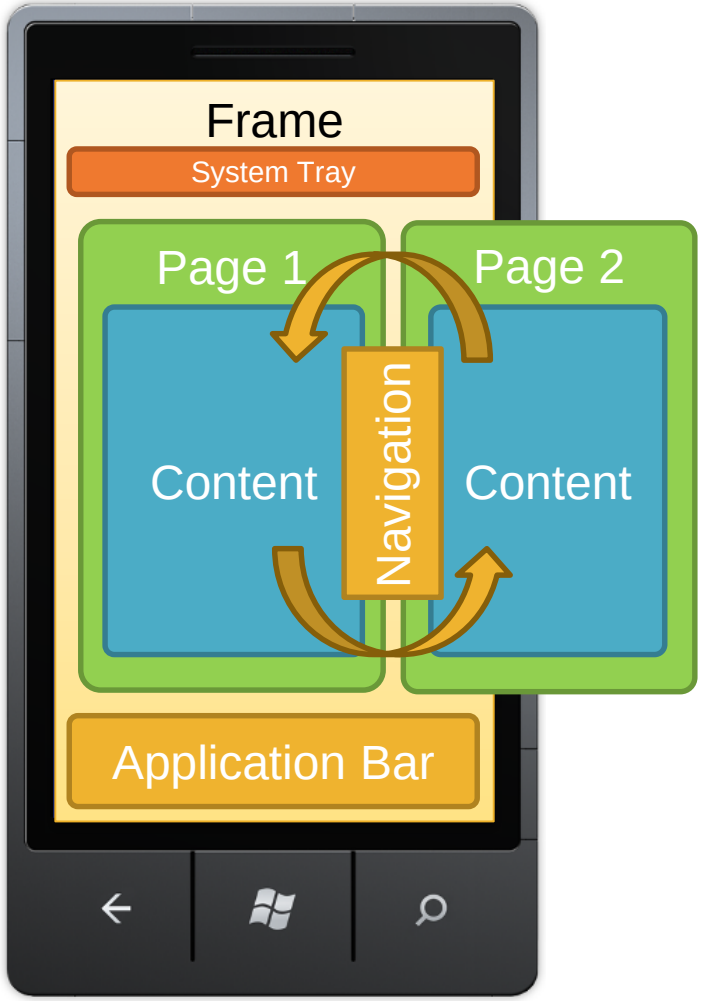


Session
An ordered workflow of user interactions spanning applications



Rahmen vs. Seite

- **Rahmen** (frame)
  - stellt das oberste Steuerelement dar
  - Ist von der Klasse PhoneApplicationFrame
  - Enthält die Seitensteuerung, Systemelement wie den Tray oder die AppBar
- **Seite** (page)
  - Füllt den gesamten Inhaltsbereich (Content) des Rahmens
  - Ist von der Klasse PhoneApplicationPage
  - Enthält Titel
  - Kann benutzerdefinierte AppBar enthalten



Seiten Navigation

- Seitenorientiertes Navigationsmodel
  - Angelehnt an WebSeitenNavigation
  - Seiten werden durch URI unterschieden
  - Seiten sind „zustandslos“


```
private void hyperlinkButton1_Click(
   object sender, RoutedEventArgs e) {
     NavigationService.Navigate(
       new Uri("/SecondPage.xaml",
         UriKind.Relative));
   }
```

```
private void hyperlinkButton2_Click(
   object sender, RoutedEventArgs e) {
     NavigationService.GoBack());
   }
```

Den Back-Key verbiegen

- „Finger weg von der Back Taste !“ – *Microsoft*
  - *Ausnahmen:*
    - Eigene Aktionen auslösen vor dem Zurück navigieren, aber Vorsicht !!!

```
<phone:PhoneApplicationPage
 x:Class="Navigation.SecondPage"
 ...
 shell:SystemTray.IsVisible="True"
 BackKeyPress="PhoneApplicationPage_BackKeyPress" >

private void PhoneApplicationPage_BackKeyPress(object sender,
  System.ComponentModel.CancelEventArgs e) {
  ...
  e.Cancel = true;
}
```

Datenübergabe zwischen den Seiten

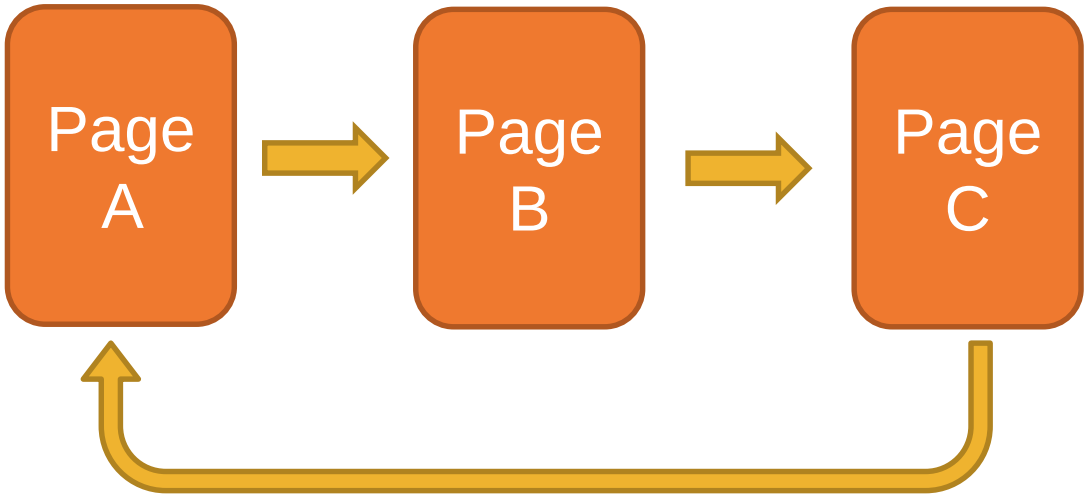
- Wie bei WebSeiten kann ein QueryString verwendet werden, um Parameter zu übergeben, dieser muss manuell erstellt werden

```
private void hyperlinkButton1_Click(object sender, RoutedEventArgs e) {
     NavigationService.Navigate(new Uri("/SecondPage.xaml?msg=" +
     textBox1.Text, UriKind.Relative));
   }
```

- Auf der Zielseite muss die Nachricht halb manuell ausgepackt werden

```
protected override void OnNavigatedTo(NavigationEventArgs e) {
  base.OnNavigatedTo(e);
  string msg = "";
  if (NavigationContext.QueryString.TryGetValue("msg", out msg)) {
    textBlock1.Text = msg;
  }
}
```

Navigationshistorie manipulieren




```
NavigationService.Navigate(new Uri("/PageA.xaml?commingFrom=PageC",  UriKind.Relative));
```

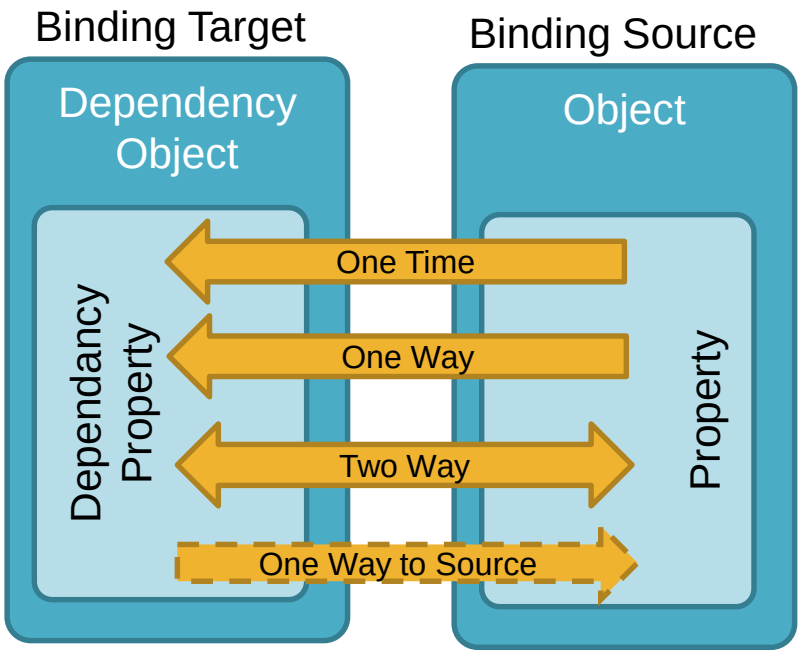
```
protected override void OnNavigatedTo(NavigationEventArgs e) {
  if (e.NavigationMode == NavigationMode.New &&
      NavigationContext.QueryString.ContainsKey( "commingFrom" )) {
        switch (NavigationContext.QueryString["commingFrom"]) {
          case "PageC":
            NavigationService.RemoveBackEntry(); // Remove PageC
            NavigationService.RemoveBackEntry(); // Remove PageB
          break;
        }
    }
}
```

Datenbindung

Datenbindung

- Datenbindung (data binding) ist ein Mechanismus, um eine lose Kopplung zwischen der UI und der Anwendungslogik herzustellen
- Als Ziel für eine Datenbindung können nur Depandency Objekts verwendet werden.
- Als Quelle kann jedes Objekt verwendet werden. Wenn Änderungen automatisch an das Bindungsziel übermittelt werden sollen, muss
  - Entweder die Schnittstelle IPropertyChanged
  - Oder ein Dependency Property

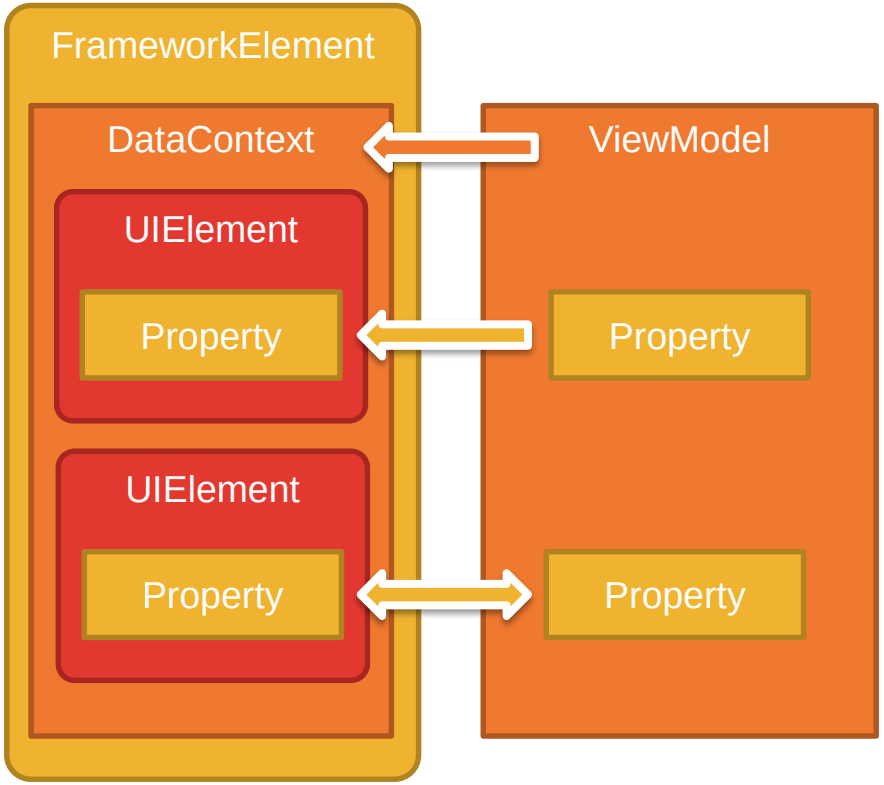
  verwendet werden



- Die Bindung kann einseitig, beidseitig oder einmalig erfolgen.
- Die Bindung kann in XAML oder im CodeBehind angelegt werden.

Datenbindung Funktionsweise

- Eine Datenklasse, welche als Quelle für die Datenbindung verwendet wird, wird **ViewModel** genannt.
- Die Datenquelle wird festgelegt, indem der Eigenschaft DataContext eines UI-Containers (FrameworkElement) eine Instanz der Datenklasse (des ViewModels) zugewiesen wird.
- Eigenschaften von UI Elementen innerhalb des Containers, sowie Eigenschaften des Containers können nun an Eigenschaften des ViewModels gebunden werden.



Datenbindung im XAML

- Eine ViewModell-Klasse

```
public class MyViewModel {
  public string TextProperty { get; set; }
}
```

- wird der Eigenschaft DataContext des Containers zugewiesen.

```
  MainContainer.DataContext = new MyViewModel {TextProperty = "Hello Word"};
```

- In der XAML Datei erfolgt dann die Bindung einer Eigenschaft des ViewModels zu einer Eigenschaft des UI-Elements

```
<Grid Name="MainContainer" >
  <TextBlock Name="ContentText" Text="{Binding TextPropety, Mode=OneWay}"
             TextWrapping="Wrap" Style="{StaticResource PhoneTextTitle3Style}" />
</Grid>
```

Data Binding Cheat Card

BASTA!

Binding Mode	
OneTime	Bind only one time
OneWay	Bind only from Model To Control
TwoWay	Bind both directions
OneWayToSurce	Bind only from Control to Model

Binding Parameter	
Converter	Converter for value conversion
Source	Object to be used as binding source
StringFormat	Formats to String
Path	Source Property

Basic Binding	
{Binding}	Bind to current DataContext.
{Binding Name}	Bind to the "Name" proeprty of the current DataContext.
{Bindind Name.Length}	Bind to the Length property of the object in the Name property of the current DataContext
{Binding ElementName=SomeTextBox, Path=Text}	Bind to the "Text" property of the element XAML element with name="SomeTextBox" or x:Name="SomeTextBox".

Collection Current Item Binding	
{Binding /}	Bind to the current item in the DataContext (when DataContext is a collection)
{Binding AllItems/}	Bind to the current item in the "AllItems" property of the DataContext
{Binding AllItems/Name}	Bind to the "Name" property of the current item in the "AllItems" property of the DataContext

XML Binding	
{Binding Source={StaticResource BooksData} XPath=/books/book}	Bind the result of XPath query "/books/book" from the XML in the XmlDataProvider in a parent's "Resources" elememt with x:Key="BooksData".
{Binding XPath=@name}	Bind to the result of an XPath query run on the XML node in the DataContext (for example in an ItemControl's DataTemplate when the ItemsControl.ItemsSource is bound to an XML data source).

Relative Source Binding	
{Binding RelativeSource={RelativeSource Self}}	Bind to the target element.
{Binding RelativeSource={RelativeSource Self}, Path=Name}	Bind to the "Name" property of the target element.
{Binding RelativeSource={RelativeSource FindAncestor, AncestorType={x:Type Window}}, Path=Title}	Bind to the title of the parent window.
{Binding RelativeSource={RelativeSource FindAncestor, AncestorType={x:Type ItemsControl}, AncestorLevel=2}, Path=Name}	Bind the the name of the 2nd parent of type ItemsControl.
{Binding RelativeSource={RelativeSource TemplatedParent}, Path=Name}	Inside a control template, bind to the name property of the element the template is applied to.
{TemplateBinding Name}	Shortcut for the previous example.

ValueConverter

- Ist von der Schnittstelle IValueConverter abgeleitet, verfügt über zwei Methoden Convert und ConvertBack. Daten werden als object übergeben müssen ge-casted werden.

```
public interface IValueConverter   {
  object Convert(object value, Type targetType, object parameter, CultureInfo culture);
  object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture);
}
```

```
public class DoubleConverter : IValueConverter {
  public object Convert(object value, Type targetType, object parameter, CultureInfo culture) {
     if (value is double) { return value.ToString();  }
     return value;
  }
  public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture) {
     var s = value as string;
     if (s != null) {
       double res;
       if (double.TryParse(s, out res)) {
        return res;
       }
     }
     return null;
   }
 }
```

```
<phone:PhoneApplicationPage.Resources>
  <local:DoubleConverter x:Key="doubleConverter" />
</phone:PhoneApplicationPage.Resources>

 <StackPanel Margin="0,350,0,0">
   <Slider Name="SliderControl" Minimum="0" Maximum="1000" Value="500" />
   <TextBox Name="TextControl" Text="{Binding ElementName=SliderControl, Path=Value,
              UpdateSourceTrigger=Default,  Mode=TwoWay,
              Converter={StaticResource doubleConverter}}"  />
 </StackPanel>
```

Schnittstelle INotifyPropertyChanged

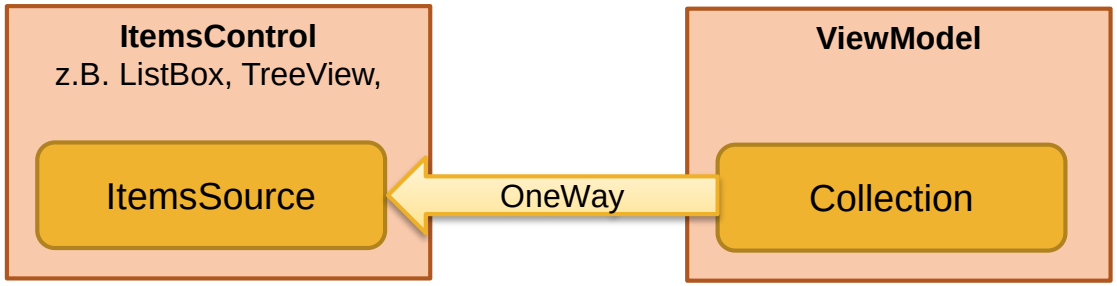
- Implementiert eine Klasse die INotifyPropertyChanged-Schnittstelle, können Benachrichtigungen über Änderungen von Eigenschaften an ein gebundenes Steuerelement übermittelt werden.
- Implementierung:

```
public class Artikel : INotifyPropertyChanged {

  public event PropertyChangedEventHandler PropertyChanged;

  protected void OnPropertyChanged([CallerMemberName] string propName = null ) {
    if (PropertyChanged != null) {
      PropertyChanged(this, new PropertyChangedEventArgs(propName));
    }
  }

  private string _headline;
  public string Headline {
    get { return _headline; }
    set {
      if (_headline == value) return;
      _headline = value;
      // OnPropertyChanged("Headline");
      OnPropertyChanged();
    }
  }
}
```

- Ändert sich die Eigenschaft `Headline`, wird das Ereignis PropertyChanged ausgelöst, falls ein oder mehrere Empfänger diese Ereignis abonniert (subscribe) haben.
- Um keine unnötigen Ereignisse auszulösen, empfiehlt es sich immer zuvor zu überprüfen, ob der gleiche Wert zugewiesen wurde, wie bereits schon vorhanden war.
- Ist ein Modell gebunden / der Eigenschaft DataContext zugewiesen, wird automatisch erkannt, dass dieses Modell die Schnittstelle IPropertyChanged implementiert und ein Abonnent für die entsprechenden Bindungen der einzelnen Werte hinzugefügt.
- Das Attribut CallerMemberName setzt automatisch den Namen der übergeordneten Funktion oder Eigenschaft ein .
-

Binden von Collections

- Anzeige für Datenlisten: ItemsControl und abgeleitete Steuerelemente
  - Zum Binden von ItemsControl an ein Kollektion Objekt dient die ItemsSource Eigenschaft.

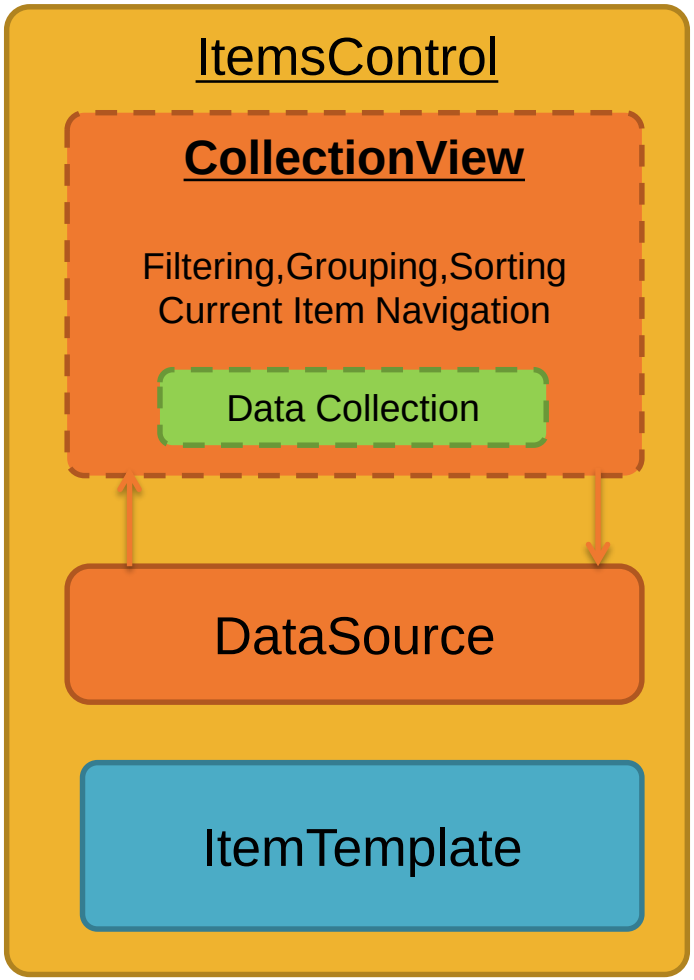


- Es kann jede Kollektion verwandt werden, die die IEnumerable-Schnittstelle implementiert.
- Um eine automatische Aktualisierung zu erreichen, muss die Kollektion die INotifyPropertyChanged-Schnittstelle implementieren.
  - Dafür stellt WPF/Win(P)RT die ObservableCollection<T>-Klasse zur Verfügung
  - Zur vollständigen Aktualisierung aller Datenwerte muss auch ein <T>-Objekt in der Kollektion die INotifyPropertyChanged-Schnittstelle implementieren.

CollectionViews und DataTemplates

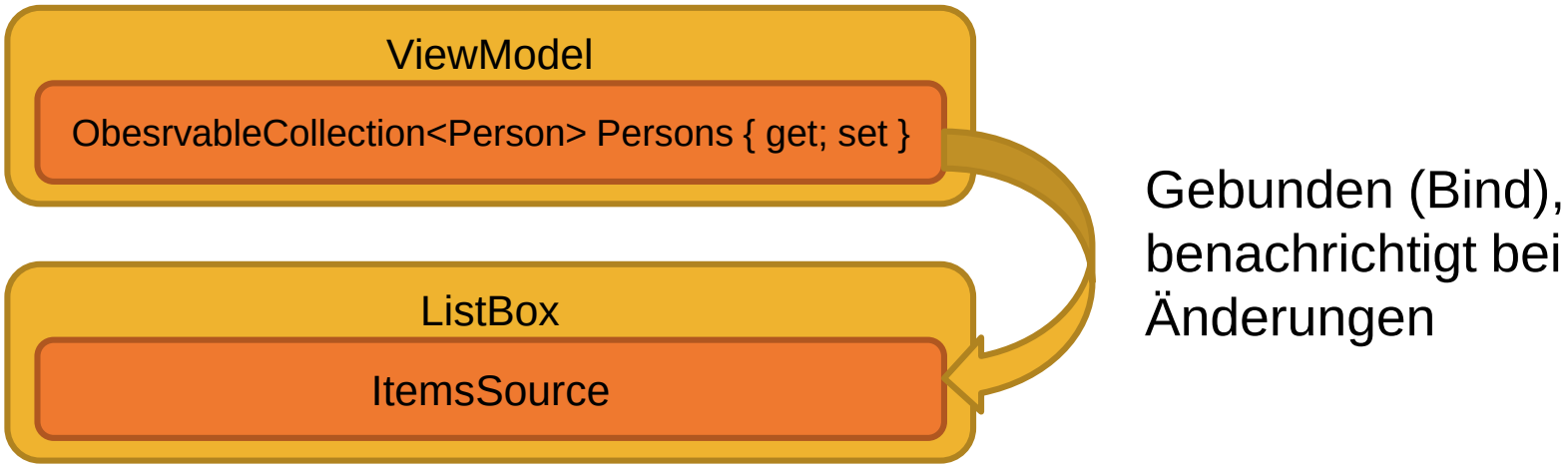
- Zwischen der Quell-Kollektion und dem ItemsControl liegt eine CollectionView.
- Die CollectionView verwendet eine DataCollection, welche mit Hilfe von einem DataTemplate angezeigt wird.
  - Der DataConntext innerhalb des Templates wird jeweils auf das aktuelle Item der Collection gesetzt.
  - Zugriff auf die Eigenschaften erfolgt via Binding
  - ItemTemplate kann als Ressource gespeichert werden

```
<ListBox Width="400" Margin="10"
         ItemsSource="{Binding Source={StaticResource myTodoList}}">
  <ListBox.ItemTemplate>
    <DataTemplate>
      <StackPanel>
        <TextBlock Text="{Binding Path=TaskName}" />
        <TextBlock Text="{Binding Path=Description}"/>
        <TextBlock Text="{Binding Path=Priority}"/>
      </StackPanel>
    </DataTemplate>
  </ListBox.ItemTemplate>
</ListBox>
```



ObservableCollection<T>

- Stellt eine dynamische Datenauflistung dar, die Benachrichtigungen bereitstellt, wenn Elemente hinzugefügt oder entfernt werden oder wenn die gesamte Liste aktualisiert wird
  - Wenn auch Änderungen der Elemente signalisiert werden, müssen diese jeweils die Schnittstelle INotifyPropertyChanged implementieren.

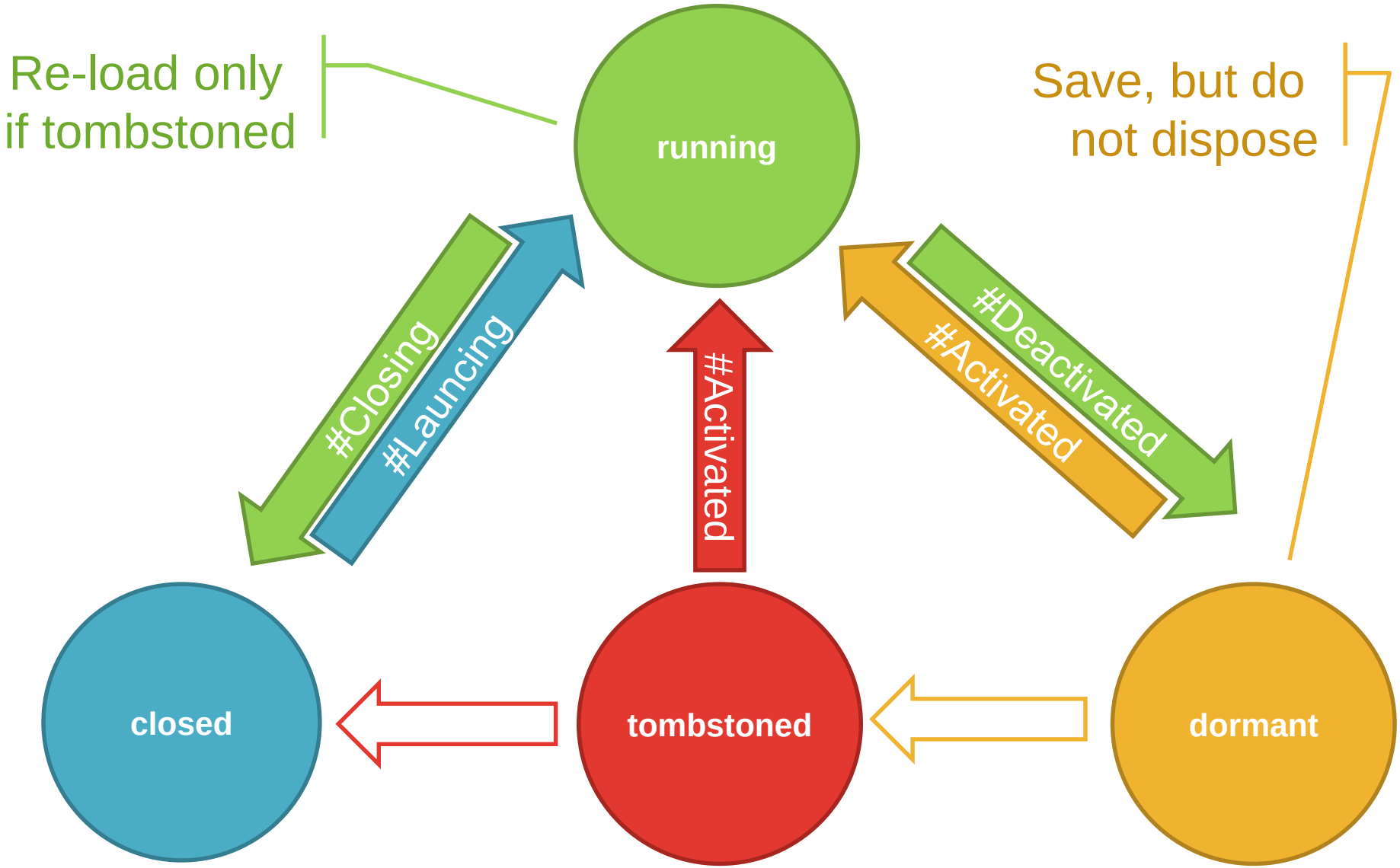


Demo / Übung

- EasyTodo
  - Eine einfache Aufgabenliste implementieren
  - Nutzung von DataTemplates und DataContext

APPLICATION LIVE CYCLE

Application Livecycle

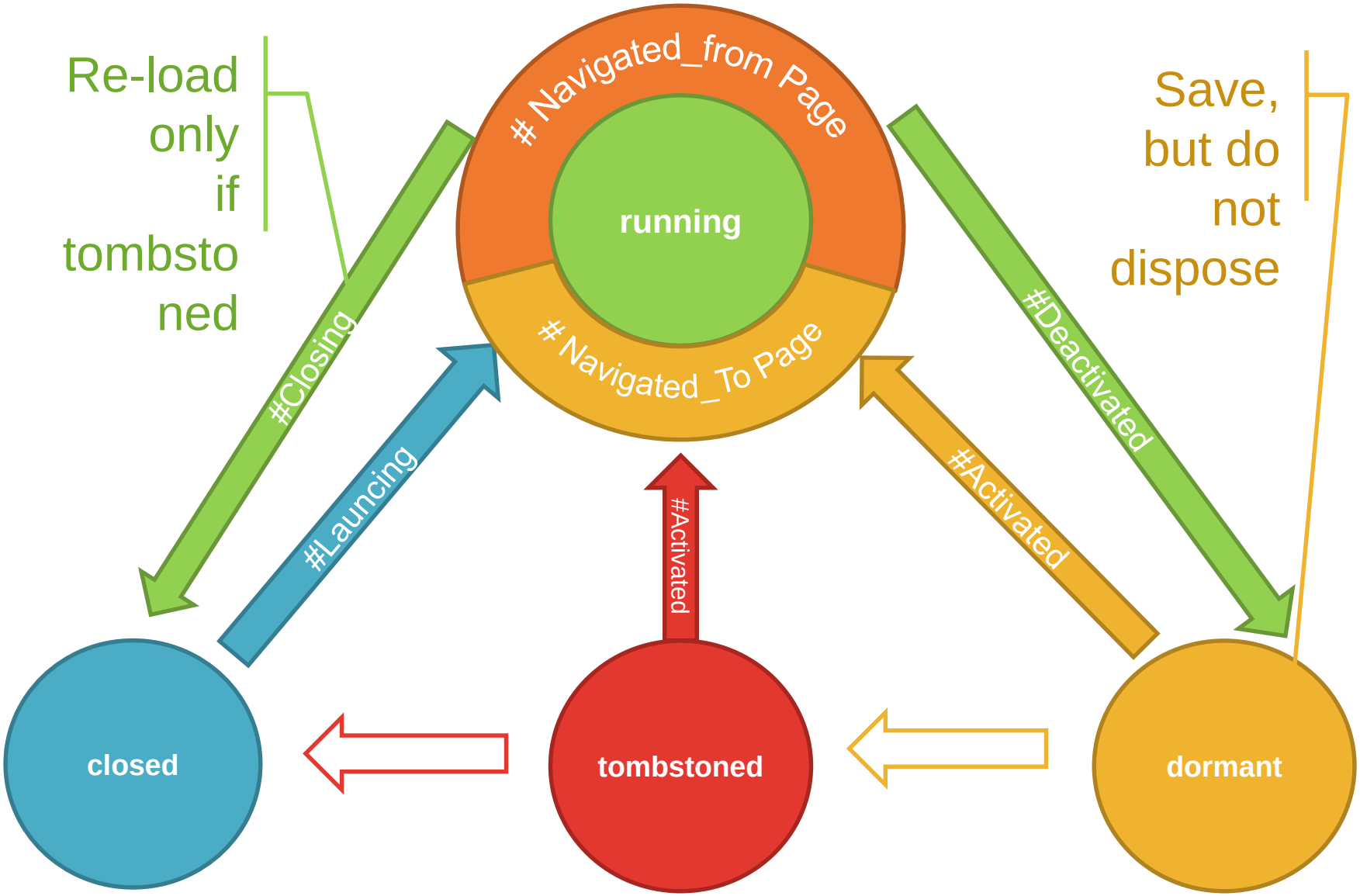


Lifecycle Events (Mango)

- **Application_Launching**
  - Wenn die App aus dem geschlossenen Zustand startet
  - Wird nicht geworfen bei Reaktivierung
- **Application_Activated**
  - Wenn die App reaktiviert wird z.B. in den Vordergrund kommt
  - Wird nicht beim Starten der App geworfen
  - Siehe (IsApplicationInstancePreserved. Variable)
- **Application_Deactivated**
  - Wenn die App deaktiviert wird z.B. in den Hintergrund geschickt
  - Wird nicht geworfen beim Schießen der App
- **Application_Closing**
  - Wenn die App geschlossen wird z.B. drücken der Zurück-Taste
  - Wird nicht geworfen beim Deaktivieren der App

- Mit dem Ereignis Deactivated sollten immer alle Status-Informationen einer App gesichert werden, weil nicht sicher ist, ob die App aus dem Zustand Dormant oder Tombstoned zurückkehren wird.
- Der Zustand **Dormant** wurde mit dem Mango-release eingeführt, um das sog. Fast-Application-Switching zu unterstützen. Dieser Zustand hält alle Daten der App im Speicher. Im Gegensatz zu dem Zustand Tombstoned werden die Threads nicht zerstört, jedoch stoppen alle Threads sowie alle Ressourcen (z.B. Sockets oder GeoPostionWatcher) werden frei gegeben.
- Werden weniger als (ca.) 3-5 Apps geladen, bevor die eigene App wieder in den Vordergrund kommt, wird die App direkt aktiviert. In dem Fall ist kein laden des Anwendungszustandes erforderlich. Windows Phone 7 kann in der aktuellen Version nicht mehr als 5 Apps im Zustand Dormant halten.
- Innerhalb des **Application_Activated** Ereignises kann mit Hilfe der Variablen IsApplicationInstancePreserved geprüft werden, ob die App aus dem Zustand Dormant oder Tombstoned kommt.

Application Livecycle Ext



Erweitern der Seite

- Seitennavigationsereignisse verwenden um die TodoItems von und zum DataManager zu synchronisiren
- Zustand der Seite in dem State-Dicttionary sichern
  - **Hinweis**: Dieser Zustand wird mit dem Schießen der App gelöscht.

```
protected override void OnNavigatedTo(NavigationEventArgs e) {
    // Wenn die Anwendung im Hintergrund in den Zustand Tombstoned läuft
    newToDoTextBox.Text = (State.ContainsKey("NewTitemText"))
      ? (string)State["NewTitemText"]
      : "";

    _viewModel.Items =
       new ObservableCollection<TodoItem>(DataManager.Instance.TodoItems);

     base.OnNavigatedTo(e);
  }

protected override void OnNavigatedFrom(NavigationEventArgs e) {
    DataManager.Instance.TodoItems = _viewModel.Items.ToList();
    State["NewTitemText"] = newToDoTextBox.Text;
    base.OnNavigatedFrom(e);
  }
```

Erweitern des DataMangers

```
 public static bool SerializeToFile<T>(string path, T serializationSource) {
   try {
     using (var store = IsolatedStorageFile.GetUserStoreForApplication())
     using (var stream = store.OpenFile(path, FileMode.OpenOrCreate, FileAccess.ReadWrite)) {
#if UseDataContract
          var serializer = new DataContractSerializer(typeof(T));
          serializer.WriteObject(stream, serializationSource);
#else
           var serializer = new XmlSerializer(typeof(T));
           serializer.Serialize(stream, serializationSource);
#endif
       }
       return true;
     } catch { return false; }
   }

 public static T DeserializeFromFile<T>(string path) {
     try {
       T returnable;
       using (var store = IsolatedStorageFile.GetUserStoreForApplication())
       using (var stream = store.OpenFile(path, FileMode.Open, FileAccess.Read)) {
#if UseDataContract
          var serializer = new DataContractSerializer(typeof(T));
          returnable = (T)serializer.ReadObject(stream);
#else
          var serializer = new XmlSerializer(typeof(T));
          returnable = (T)serializer.Deserialize(stream);
#endif
       }
       return returnable;
     } catch { return default(T); }
   }
```

Load und Save Funktion

BASTA!

DataManager

```
public void Load() {
  TodoItems = DeserializeFromFile<List<TodoItem>>("Items.dat") ?? new List<TodoItem>();
}

public void Save() {
  SerializeToFile("Items.dat", TodoItems);
}
```

Application → App.xaml.cs

```
private void Application_Launching(object sender, LaunchingEventArgs e) {
      DataManager.Instance.Load();
    }

private void Application_Activated(object sender, ActivatedEventArgs e) {
      if (!e.IsApplicationInstancePreserved) {
        DataManager.Instance.Load();
      }
    }

private void Application_Deactivated(object sender, DeactivatedEventArgs e) {
      DataManager.Instance.Save();
    }

private void Application_Closing(object sender, ClosingEventArgs e) {
      DataManager.Instance.Save();
    }
```

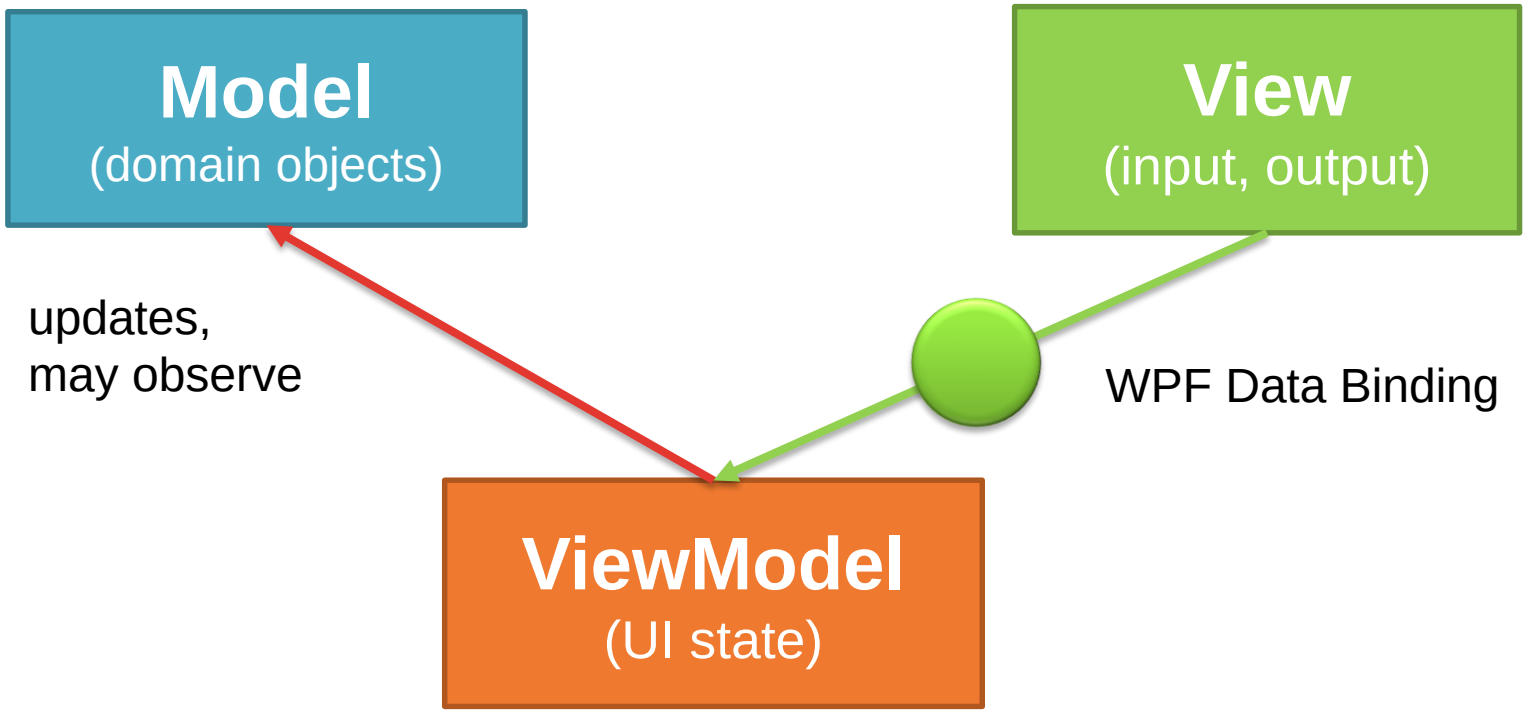
- Allgemeine Hinweise zur Nutzung der Zustände

Ereignis oder Methode	Aktionen
Launching Ereignis	Nur kleine Code-Blöcke, keine ressourceintensiven Aktionen. ggf. Laden von Einstellungen sowie des initialen Zustandes der APP aus dem isolated storage
OnNavigatedFrom Methode	Wenn es keine Back-Navigation war, den UI Zustand im state dictionary sichern.
Deactivated Ereignis	Den Zustand der Anwendung im state dictionary sichern für den Fall, dass die App **tombstoned** ist. Zusätzlich wird der Anwendungszustand persistent im isolated storage gesichert, falls die Anwendung **terminated** wird. Der Anwendungszustand im Speicher muss erhalten werden, falls die Anwendung **dormant** ist.
Activated Ereignis	IsApplicationInstancePreserved prüfen. Wenn **true**, keine Aktion erforderlich. Wenn **false**, den Anwendungszustand mit Hilfe des state dictionarys wiederherstellen.
OnNavigatedTo Methode	Prüfen ob die Seite eine neue Instanz ist, falls nicht, ist der Zustand der Seite in Ordnung. Ansonsten ggf. aus dem state dictionary wiederherstellen.
Closing Ereignis	Persistenten Anwendungszustand im isolated storage sichern.

- EasyTodo erweitern
  - Anwendungs- und Seitenzustände verwenden

MVVM

Model-View-ViewModel



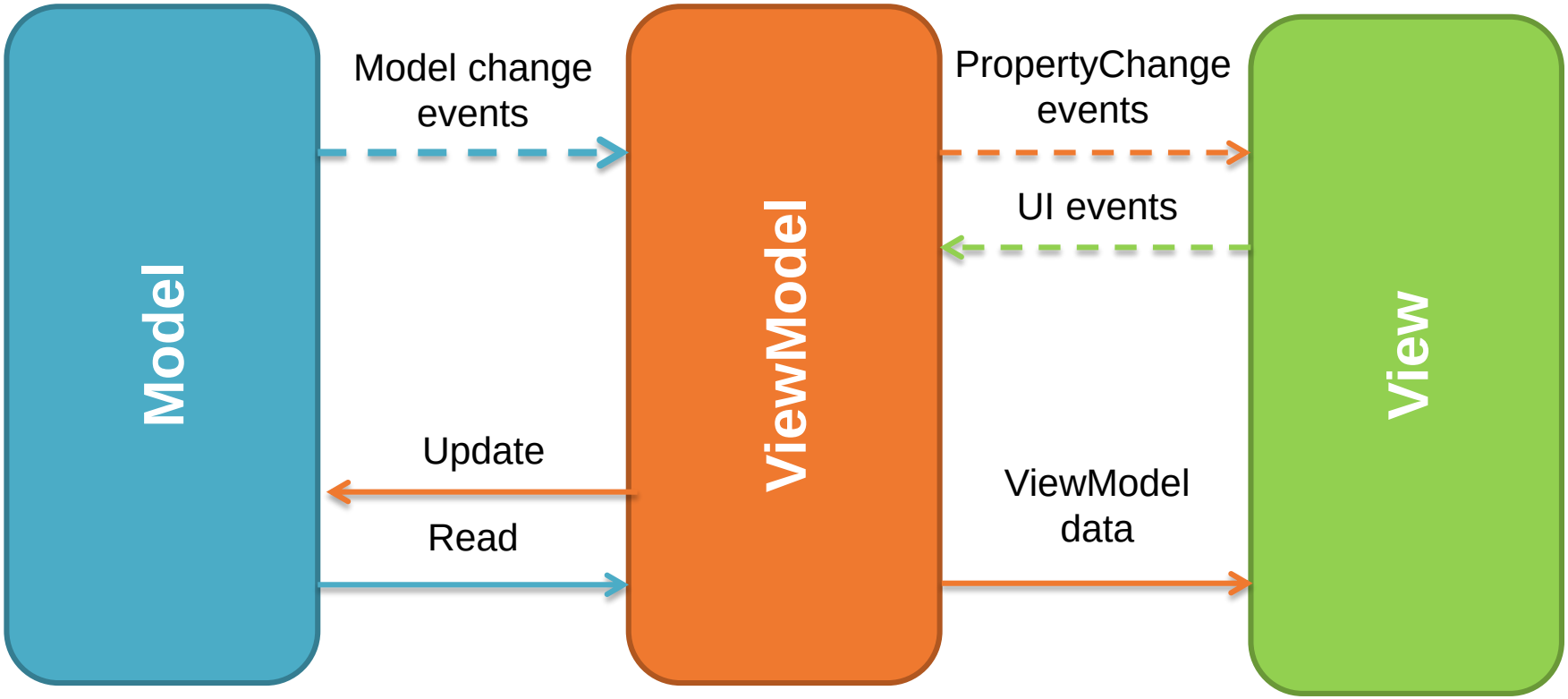
View.DataContext = ViewModel;

http://blogs.msdn.com/b/johngossman/archive/2005/10/08/478683.aspx

Aufgabentrennung

Model	View	ViewModel
Greift auf Datenbank zu	Positionierung der UI Elemente auf dem Bildschirm	Eingaben prüfen und Fehler ggf. zur Anzeige bringen
Erzeugt neue Einträge	Verantwortliche für das visuelle Erscheinen der UI Elemente , z.B. Font, Size, Color, usw.	Ruft das Model auf, um ggf. neue Elemente zu erzeugen
	Tastaturereignisse in Navigations- und Bearbeitungsaktionen umsetzen	Steuert Aktivierung und Deaktivierung von UI Elementen
	Mouse-Ereignisse entgegennehmen und in Anwendungsereignisse wandeln	

Model-View-ViewModel



- Quelle:
  - http://msdn.microsoft.com/en-us/library/ff798384.aspx

MVVM Charakterisierung

- **View** und **Model** sind isoliert
- **ViewModel** ändert keine Steuerelemente direkt
- Die **meisten** Interaktionen zwischen **View** und **ViewModel** erfolgen mittels Datenbindungen
- Codebehind wird auf ein **Minimum** reduziert bis auf wenige Ausnahmen:
  - Programmcode für reine Anzeigezwecke
  - Programmcode zur Verarbeitung oder Weitergabe von Ereignissen (Messaging)
- Modelle werden mit Hilfe des ViewModelLocator gebunden.

DelegateCommand

- Die Implementierung für Kommandos mit Hilfe der Schnittstelle ICommand ist immer gleich. Um nicht jedes Mal eine eigene Klasse für jedes Kommando ausprogrammieren zu müssen, kann die folgende Klasse DelegateCommand als allgemeine Implementierung verwendet werden.

```
public class DelegateCommand : ICommand
{
  Func<object, bool> canExecute;
  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) {
    this.executeAction(parameter);
  }
}
```

uMVVM Framework

- Komponenten für eine MVVM Anwendung
  - DelegateCommand
  - ViewModels (von ViewModelBase abgeleitet)
  - DataBinding und CommandBinding

- Zur Vereinfachung der Erstellung der ViewModell-Klassen
  - die Schnittstelle `INotifyPropertyChanged` in der Klasse ViewModelBase implementieren.

```
public abstract class ViewModelBase : INotifyPropertyChanged {

    public event PropertyChangedEventHandler PropertyChanged;
    protected void OnPropertyChanged ([CallerMemberName] string propName = null) {
      if (PropertyChanged != null) {
        PropertyChanged(this, new PropertyChangedEventArgs(propName));
      }
}
```

Verwenden des DelegateCommands

```
public class PersonViewModel {
   private ObservableCollection<Person> _personDataSource;
   private ICommand _loadDataCommand;

   public PersonViewModel()     {
      _personDataSource = new ObservableCollection<Person>();
      _loadDataCommand = new DelegateCommand(this.LoadDataAction);
   }

   private void LoadDataAction(object p) {
     DataSource.Add(new Person() { Name = "John"});
     DataSource.Add(new Person() { Name = "Kate"});
     DataSource.Add(new Person() { Name = "Sam"});
   }

  public ICommand LoadDataCommand {
    get { return this.loadDataCommand;  }
  }

  public ObservableCollection<Person> DataSource {
    get { return this.personDataSource; }
  }
}
```

```
public class Person {
   public string Name { get; set; }
}
```

```
public MainPage() {
  InitializeComponent();
  // simple way to bind the view to the view model
  this.DataContext = new ViewModelBase.PersonViewModel();
}
```

```
<StackPanel x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
  <Button Content="LoadData" Command="{Binding LoadDataCommand}" />
  <ListBox ItemsSource="{Binding DataSource}">
    <ListBox.ItemTemplate>
      <DataTemplate>
       <TextBlock Text="{Binding Name}" />
      </DataTemplate>
    </ListBox.ItemTemplate>
  </ListBox>
</StackPanel>
```

MVVM hinzufügen / Projekt umstellen

- MVVMLight oder anderes Framework hinzufügen
- ViewModel Locator erstellen (falls nicht vorhanden)
- ViewModel erstellen und mit Hilfe des ViewModel Locators verwenden

```
DataContext="{Binding Todo, Source={StaticResource Locator}}"
```

- Hilfsklassen aus Code-Behind in ViewModell übernehmen z.B. DataContext
- Öffentliche Eigenschaften für Bindung unter Berücksichtigung von INotifyPropertyChanged im ViewModel erstellen

```
private string _newItemText;
public string NewItemText {
  get { return _newItemText; }
  set {
      if (_newItemText == value) return;
      _newItemText = value;
      RaisePropertyChanged();
  }
}
```

```
<TextBox
  x:Name="newToDoTextBox"  Grid.Column="0"
  Text="{Binding NewItemText, Mode=TwoWay}"
  FontFamily="{StaticResource PhoneFontFamilyLight}"
/>
```

- Commandos erstellen und binden

```
public RelayCommand AddButtonCommand { get; set; }
```

```
<Button
  Content="add"  Grid.Column="1"
  Command="{Binding AddButtonCommand}"
/>
```

ViewModelBase

- Stellt Basis-Implementierung für die Schnittstelle INotifyPropertyChanged bereit

```
private string _newItemText;
public string NewItemText {
  get { return _newItemText; }
  set {
    if (_newItemText == value) return;
    _newItemText = value;
    RaisePropertyChanged();
  }
}
```

```
<TextBox
  x:Name="newToDoTextBox"  Grid.Column="0"
  Text="{Binding NewItemText, Mode=TwoWay}"
  FontFamily="{StaticResource PhoneFontFamilyLight}"
>
```

- Stellt diverse Funktionen und Eigenschaften bereit z.B. Unterscheidung on Design oder Runtime-Mode

```
if (IsInDesignMode) {
  // Code runs in Blend --> create design time data.
}
else

{
// Code runs in Application --> create real data model.
}
```

```
public bool IsInDesignMode { get { return DesignerProperties.IsInDesignTool; } }
```

ViewModelLocator

- Stellt eine statische Instanz der ViewModels bereit

```
public class ViewModelLocator     {
   private static TodoViewModel _todo;

   public ViewModelLocator()         {
     ////if (ViewModelBase.IsInDesignModeStatic)
     ////{
     ////    // Create design time services and viewmodels
     ////}  else {
     ////    // Create run time services and view models
     ////}
   }

   public static TodoViewModel TodoStatic {
     get { return _todo ?? (_todo = new TodoViewModel()); }
   }

   [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Performance", "CA1822:MarkMembersAsStatic",
            Justification = "This non-static member is needed for data binding purposes.")]
   public TodoViewModel Todo {
     get { return TodoStatic; }
   }
}
```

- Im Zusammenspiel mit einer Statischen Ressource in den View verwendet

```
<Application.Resources>
   <vm:ViewModelLocator x:Key="Locator" d:IsDataSource="True" />
</Application.Resources>
```

```
DataContext="{Binding Todo, Source={StaticResource Locator}}"
```

RelayCommand

- Implementierung der ICommand-Schittstelle als DelegateCommand mit und ohne Parameter

```
public DelegateCommand AddButtonCommand { get; set; }

public DelegateCommand<ToDoItem> DelButtonCommand { get; set; }
```

```
DelButtonCommand = new RelayCommand<ToDoItem>(
   (o) => {
            ToDoItems.Remove(o);
            _toDoDB.ToDoItems.DeleteOnSubmit(o);
            _toDoDB.SubmitChanges();
           });


AddButtonCommand = new RelayCommand(
   () => {
            // Create a new to-do item based on the text box.
            var newToDo = new ToDoItem { ItemName = NewItemText };
            // Add a to-do item to the observable collection.
            ToDoItems.Add(newToDo);
            // Add a to-do item to the local database.
            _toDoDB.ToDoItems.InsertOnSubmit(newToDo);
         });
```

Hinweis: Die Bindung erfolgt nicht direakt an die Eigenschaft DelButtonCommand, weil der DataConntext innerhalb des ItemTemplates nicht auf das MainViewModell sondern auf jeweils ein ToDo Item gesetzt ist

- Bindung mit und ohne Parameter

```
<Button Content="add" Grid.Column="1" x:Name="newToDoAddButton"  Command="{Binding AddButtonCommand}" />
```

```
<Button Grid.Column="2" x:Name="deleteTaskButton" BorderThickness="0" Margin="0"
        Command="{Binding Path=Main.DelButtonCommand, Source={StaticResource Locator}}" CommandParameter="{Binding}"
```

Übung / Demo µMVVM

- Demonstration des µMVVM Frameworks
  - Commands, Databinding
  - ViewModelLocator

KAMERA APIIEW

Windows Phone Camera Classes

- Windows Phone 7/8
  - CameraCaptureTask
  - PhotoCamera
- Windows Phone 8 Only
  - PhotoCaptureDevice
  - AudioVideoCaptureDevice

Vergleich WP7.X vs. WP 8.0

	CamerCaptureTask	PhotoCamera	PhotoCaptureDevice
Namespace	Microsoft.Phone.Tasks	Microsoft.Devices	Windows.Phone.Media.Capture
Compatibility	WP7 / WP8	WP7 / WP8	WP8
Live Pixel Buffer Available	No	Yes	Yes
Controllable Camera Parameters	None	Camera Type, Flash Mode, Autofocus, Resolution	Camera Type, Exposure Compensation, Exposure Time, Flash Mode, Flash Power, Focus Illumination Mode, Iso, Auto Focus, Manual White Balance, Scene Mode, White Balance Preset, Resolution
Support for native code	No	No	Yes (Win32)

CameraCaptureTask

- Einfache schnelle Lösung Bilder aus App aufzunehmen
- Speicherverbrauch der CameraCaptureTask wird nicht dem Speicher der App zugerechnet
  - Interessant für Geräte mit 256Mb RAM
- Keine Live Manipulation möglich
- Bevorzugte Variante, wenn verwendbar

- Weitere Information

PhotoCamera & PhotoCaputereDevice

- Für die Erstellung von „Custom Camera Apps“
- Erstellen eines Benutzerdeferierten ViewFinders
- Anpassen der Camera-Parameter
- Zugriff auf den „Shutter-Button“
- Zugriff auf den „Live Preview Buffer“

- WPAppManifest.xaml

- <Capability Name="ID_CAP_ISV_CAMERA"/>
  - Erforderlich für Verwendung der Camera in der APP
- <Requirement Name="ID_REQ_FRONTCAMERA"/>
  - Optional: Wenn eine App eine ForntKamera erfordert
- <Requirement Name="ID_REQ_REARCAMERA"/>
  - Optional: Wenn eine App eine RearCamera erfordert

Vorraussetungen

- WPAppManifest.xaml

- <Capability Name="ID_CAP_ISV_CAMERA"/>
  - Erforderlich für Verwendung der Camera in der APP
- <Requirement Name="ID_REQ_FRONTCAMERA"/>
  - Optional: Wenn eine App eine ForntKamera erfordert
- <Requirement Name="ID_REQ_REARCAMERA"/>
  - Optional: Wenn eine App eine RearCamera erfordert

Cameracapturetask Beispiel

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
    <Button Content="Capture!" HorizontalAlignment="Left" Height="83" Margin="75,25,0,0"
            VerticalAlignment="Top" Width="295" Click="Button_Click_1"/>
    <Image x:Name="image1" HorizontalAlignment="Left" Height="424" Margin="10,140,0,0"
            VerticalAlignment="Top" Width="436"/>
</Grid>
```

```
private void Button_Click_1(object sender, RoutedEventArgs e) {
    var cameraTask = new CameraCaptureTask();
    cameraTask.Completed += (o, args) => {
       if (args.TaskResult == TaskResult.OK) {
         var bmp = new BitmapImage();
         bmp.SetSource(args.ChosenPhoto);
         image1.Source = bmp;
       }
    };
    cameraTask.Show();
 }
```

Microsoft.Devices.PhotoCamera

Properties

- AvailableResolutions
- CameraType
- FlashMode
- IsFocusAtPointSupported
- IsFocusSupported
- Orientation
- PreviewResolution
- Resolution
- YCbCrPixelLayout

Methods

- CancelFocus
- CaptureImage
- Focus
- FocusAtPoint
- GetPreviewBufferArgb32
- GetPreviewBufferY
- GetPreviewBufferYCbCr
- IsFlashModeSupported

Events

- AutoFocusCompleted
- CaptureCompleted
- CaptureImageAvailable
- CaptureStarted
- CaptureThumbnailAvailable
- Initialized

http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh202956(v=vs.105).aspx

Benutzerdefinierte Viewfinder

BASTA!

- System.Windows.Media.VideoBrush

```
<Canvas Height="273" HorizontalAlignment="Left" Margin="17,18,0,0"
        Name="canvas1" VerticalAlignment="Top" Width="426">
  <Canvas.Background>
      <VideoBrush x:Name="previewCanvas" Stretch="UniformToFill" />
   </Canvas.Background>
</Canvas>
```

```
PhotoCamera _cam = new PhotoCamera();
. . .
previewCanvas.SetSource(_cam);

previewCanvas.RelativeTransform = new CompositeTransform() {
        CenterX = 0.5, CenterY = 0.5, Rotation = 90
};
```

Hardwarebuttons & Events

- Microsoft.Devices.CameraButtons

```
CameraButtons.ShutterKeyPressed += (sender, args) => { _cam.CaptureImage(); };
CameraButtons.ShutterKeyHalfPressed += (sender, args) => { _cam.Focus(); };
CameraButtons.ShutterKeyReleased += (sender, args) => { . . . };
```

- ImageAvaiable und CaptureComplete

```
_cam.CaptureImageAvailable += (sender, args) => {
    var library = new MediaLibrary();
    string fileName = "nokia_training_" + _imgCounter + ".jpg";
    library.SavePictureToCameraRoll(fileName, args.ImageStream);
};

_cam.CaptureCompleted += (sender, args) => { _imgCounter++; };
```

PhotoCaptureDevice

Properties

- AvailableSensorLocations
- CaptureResolution
- FocusRegion
- PreviewResolution
- SensorLocation
- SensorRotationInDegrees

Events

- PreviewFrameAvailable
- VendorSpecificDataAvailable

Methods

- Close
- CreateCaptureSequence
- FocusAsync
- GetAvailableCaptureResolutions
- GetAvailablePreviewResolutions
- GetPreviewBufferArgb
- GetPreviewBufferY
- GetPreviewBufferYCbCr
- GetProperty
- GetSupportedPropertyRange
- GetSupportedPropertyValues
- IsFocusRegionSupported
- IsFocusSupported
- OpenAsync
- PrepareCaptureSequenceAsync
- ResetFocusAsync
- SetCaptureResolutionAsync
- SetPreviewResolutionAsync
- SetProperty

http://msdn.microsoft.com/en-us/library/windowsphone/develop/jj662940(v=vs.105).aspx

Eigenschaften

KnownCameraGeneralProperties

- AutoFocusRange
- EncodeWithOrientation
- IsShutterSoundEnabledByUser
- IsShutterSoundRequiredForRegion
- ManualFocusPosition
- PlayShutterSoundOnCapture
- PreviewFrameRate
- SpecifiedCaptureOrientation

KnownCameraPhotoProperties

- ExposureCompensation
- ExposureTime
- FlashMode
- FlashPower
- FocusIlluminationMode
- Iso
- LockedAutoFocusParameters
- ManualWhiteBalance
- SceneMode (CameraSceneMode)
- WhiteBalancePreset

Auslesen und setzen von Parametern

BASTA!

- SceneMode

```
var sceneMode = _cam.GetProperty(KnownCameraPhotoProperties.SceneMode);
_cam.SetProperty(KnownCameraPhotoProperties.SceneMode, CameraSceneMode.Macro);
```

- WhiteBalancePreset

```
var whiteBalance = _cam.GetProperty(KnownCameraPhotoProperties.WhiteBalancePreset);
_cam.SetProperty(KnownCameraPhotoProperties.WhiteBalancePreset, WhiteBalancePreset.Cloudy);

// Default Wert ist NULL
_cam.SetProperty(KnownCameraPhotoProperties.WhiteBalancePreset, null);
```

ENUM to Listpickerhelper

```
public class EnumListItem<T> where T : struct, IComparable {
  public T? Value { get; set; }

  public string Name { get; private set; }

  public static IEnumerable<EnumListItem<T>> CreateList(string defaultName = null) {
    var res = from t in typeof (T).GetFields()
              where t.IsLiteral
              select new EnumListItem<T> {
                Value = (T?) t.GetValue(null),
                Name = t.Name
              };
    return (!string.IsNullOrEmpty(defaultName))
      ? (res.Union(new [] {new EnumListItem<T>{ Name = defaultName, Value = null} }))
      : res ;
  }
}
```

LiVE Effekte erzeugen

- Timer und Bitmap anlegen und initialisieren

```
_bmp = new WriteableBitmap((int) _cam.PreviewResolution.Width, (int) _cam.PreviewResolution.Height);

_timer = new DispatcherTimer {Interval = TimeSpan.FromMilliseconds(10)};
_timer.Tick += timer_Tick;
_timer.Start();
```

```
private void timer_Tick(object sender, EventArgs e) {
  var pixelData =
    new int[(int)(_cam.PreviewResolution.Width*_cam.PreviewResolution.Height)];

   _cam.GetPreviewBufferArgb(pixelData);
   for (int i = 0; i < pixelData.Length; i++) {
      pixelData[i] = ColorToGray(pixelData[i]);
    }

    pixelData.CopyTo(_bmp.Pixels, 0);
    imgFilter.Source = _bmp;
}
```

Lense-APPS

http://msdn.microsoft.com/en-us/library/windowsphone/develop/jj206990(v=vs.105).aspx

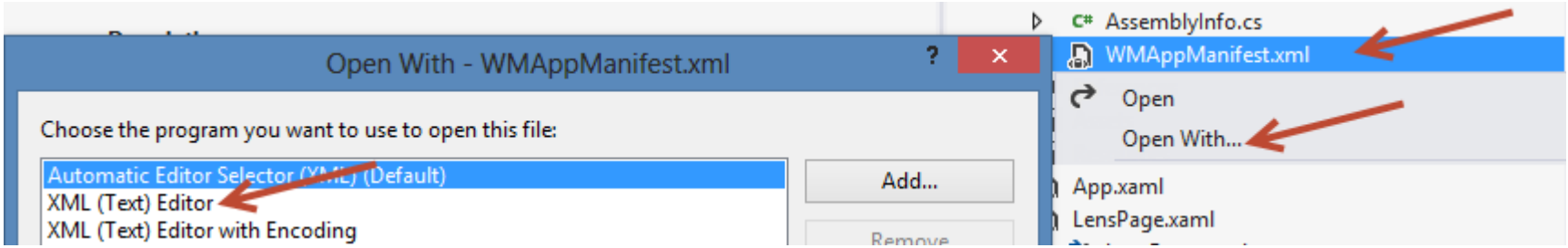
Creating a lens app

1. Die Anwendung mit einer Seite starten, welche einen **viewfinder** anzeigt
   - Alternativ kann eine separate Startseite für den Lens-Mode mit Hilfe eines URI-Mappers angebenden werden
2. Kann nur auf einem Telefon (nicht im Emumator) entwickelt werden
3. ID_CAP_ISV_CAMERA und ID_CAP_MEDIALIB_PHOTO setzen
   - ggf. ID_REQ_REARCAMERA setzen
4. Die Erweiterung **Camera_Capture_App** reginstrieren
5. Spezielle Icons für den „lens picker“ im ***/Asset*** Ordner bereitstellen

Dateiname	Auflösung
Lens.Screen-WVGA.png	173 × 173
Lens.Screen-720p.png	259 × 259
Lens.Screen-WXGA.png	277 × 277

Lens-Erweiterung registrieren

In der Datei WMAppManifest.xml, manuell einfügen !!!




```
<Extensions>
  <Extension ExtensionName="Camera_Capture_App"
             ConsumerID="{5B04B775-356B-4AA0-AAF8-6491FFEA5631}
             TaskID="_default" />
</Extensions>
```

Erstellen eines URI-Mappers

```
public class LensUriMapper : UriMapperBase {
   private string _tempUri;

   public override Uri MapUri(Uri uri) {
     _tempUri = uri.ToString();

     // Look for a URI from the lens picker.
     if (_tempUri.Contains("ViewfinderLaunch")) {
       // Launch as a lens, launch viewfinder screen.
       return new Uri("/LensPage.xaml", UriKind.Relative);
     }

     // Otherwise perform normal launch.
     return uri;
   }
 }
```

LensPage erstellen

BASTA!

```
<Canvas Name="viewFinderCanvas" >
    <Canvas.Background>
        <VideoBrush x:Name="viewFinderBrush" />
    </Canvas.Background>
</Canvas>
```

```
protected override void OnNavigatedTo(NavigationEventArgs e){
    cam = new PhotoCamera();
    cam.CaptureImageAvailable += cam_CaptureImageAvailable;
    viewFinderBrush.SetSource(cam);
}
```

```
void cam_CaptureImageAvailable(object sender, ContentReadyEventArgs e){
    using (e.ImageStream)    {
        var library = new MediaLibrary();
        library.SavePictureToCameraRoll("photo.jpg", e.ImageStream);
    }
}
```

Richtilienen für eine Lens App

- Der “Lens splash screen” erscheint immer in “landscape orientation”
- Die Lens icons sollten WVGA, HD720p, und WXGA unterstützen
- Konsistentes Verhalten mit der WP 8 „Camera“-Anwendung.
  - Gestenunterstützung: „swipe left“ um den „preview“-Screen zu öffnen.
  - Portrait und Landscape unterstützen.
  - Buttons:
    - Half press (focus), Hardware capture
    - Touch to capture (with focus)
  - Flash icons und Zustandsanzeigen „On, Off, Auto“, ggf. „Front Facing Camera“ Button
  - Focus brackets
- Je Aufnahme maximal ein Bild in der **Cameraroll** speichern.
- Alle weiteren Bilder werden in dem **App-Local-Storag**e abgelegt.
- Wenn ein „**Capture-And-Confirm**“ Dailog intergiert wird:
  - Verwenden von konsistenten Icons für : **Save**, **Save copy**, und **Delete**.
  - **Delete** und **Save** müssen in den **ViewFinder** zurückkehren .

Fragen?

Brauchen Sie Unterstützung bei .NET, Silverlight, Lightswitch, WCF, WPF, ASP.NET, IIS, Windows 8 oder Windows Phone 8 ?

- Beratung bei Einführung, Migration und Betrieb
- (Vor-Ort-)Schulungen, Workshops
- Coaching (Vor-Ort | Telefon | E-Mail | Online-Meeting)
- Support (Vor-Ort | Telefon | E-Mail | Online-Meeting)
- Entwicklung von Prototypen und Lösung

Matthias Fischer IT Consult
http://www.dotnetautor.de.de
Telefon +49 1520 1920 708
matthias@dotnetautor.de