Universal Windows Platform

Intro

- Slide deck: Desktop was yesterday
- Slide deck: Windows 10 for developers
- Slide deck: The world has changed

.NET Core, .NET Standard

- .NET and Universal Windows Platform Development
- Introducing .NET Standard
- .NET API Port

Downloadlink for Emulators

- Additional Emulators for Visual Studio

UWP Community Toolkit

- Additional controls, helpers, etc.
- Github
- Documentation

Create a UWP app

- Discuss the created files
  - package.appxmanifest
- Add MVVMLight

XAML-Changes to WPF

- TextBlock:
  - Header property
  - Placeholder property
  - Mode = TwoWay! (default for all bindings: OneWay)
- x:Bind syntax
- New controls
  - AutoSuggestBox
  - GridView

    - ProgressRing
    - Hub
    - SemanticZoom
- Responsive layout (more details below)

AutoSuggest-Box

- use x:Bind to bind to a method in ViewModel

```
<AutoSuggestBox ItemsSource="{Binding SearchResult}"
        Text="{Binding SearchText, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay}"
        TextChanged="{x:Bind ViewModel.Search}"
        PlaceholderText="Search..."
        QueryIcon="Find"
        Width="200">
    <AutoSuggestBox.ItemTemplate>
        <DataTemplate>
            <TextBlock Text="{Binding LastName}" />
        </DataTemplate>
    </AutoSuggestBox.ItemTemplate>
</AutoSuggestBox>
```

```
private string searchText;

public string SearchText
{
    get { return searchText; }
    set { this.Set(ref searchText, value); }
}

private IEnumerable<Person> searchResult;

public IEnumerable<Person> SearchResult
{
    get { return searchResult; }
    set { this.Set(ref searchResult, value); }
}

public void Search()
{
    this.SearchResult = this.People.Where(p => p.LastName.ToLower().Contains(this.SearchText
}
```

GridView

```
<GridView ItemsSource="{Binding People}" >
    <GridView.ItemTemplate>
        <DataTemplate>
            <Grid>
                <Image Source="{Binding ImageUrl}" />
                <Border Background="Black"
                VerticalAlignment="Bottom"
                HorizontalAlignment="Left"
                    Padding="3" Margin="3">
                    <TextBlock Text="{Binding LastName}"
                            Foreground="White"/>
                </Border>
            </Grid>
        </DataTemplate>
    </GridView.ItemTemplate>
    <GridView.ItemContainerStyle>
        <Style TargetType="GridViewItem">
            <Setter Property="Width" Value="150" />
            <Setter Property="Height" Value="150" />
        </Style>
    </GridView.ItemContainerStyle>
</GridView>
```

Hub with SemanticZoom

```
<SemanticZoom ScrollViewer.ZoomMode="Enabled">
    <SemanticZoom.ZoomedInView>
        <Hub Name="hub">
            <HubSection MinWidth="600" Header="Aktuell">
                <HubSection.Background>
                    <ImageBrush ImageSource="http://softaware-new.azurewebsites.net/about-us
                                Stretch="UniformToFill" />
                </HubSection.Background>
            </HubSection>
            <HubSection Header="GridView">
                <DataTemplate>
                    <GridView ItemsSource="{Binding People}" >
                        <!-- ... -->
                    </GridView>
                </DataTemplate>
            </HubSection>
            <HubSection Header="Test">

            </HubSection>
        </Hub>
    </SemanticZoom.ZoomedInView>
    <SemanticZoom.ZoomedOutView>
        <ListView
            ItemsSource="{Binding SectionHeaders, ElementName=hub}"
            HorizontalAlignment="Center" VerticalAlignment="Center">
        </ListView>
```

```
    </SemanticZoom.ZoomedOutView>
</SemanticZoom>
```

Offline scenarios

- Storage API
- SQLite
  - VS Extensions: SQLite for Universal App Platform
- Create an interface for data retrieval
  - Offline implementation: uses online implementation if possible, otherwise Offline
- Note that images or other files have to be made available too

```
private async static Task UpdateCache(IList<Recipe> recipes)
{
  string filename = "recipes.json";
  var file = await ApplicationData.Current.LocalFolder.CreateFileAsync(
    fileName,
    CreationCollisionOption.ReplaceExisting);

  await FileIO.WriteTextAsync(file, JsonConvert.SerializeObject(recipes));
}
```

```
private async static Task<IList<Recipe>> ReadCache()
{
  string fileName = "recipes.json";

  var file = await ApplicationData.Current.LocalFolder.TryGetItemAsync(fileName) as StorageF

  if (file != null)
  {
    var json = await FileIO.ReadTextAsync(file);
    return JsonConvert.DeserializeObject<List<Recipe>>(json);
  }
  else
  {
    return new List<Recipe>();
  }
}
```

Navigation with Behaviors SDK

- Add namespaces

```
xmlns:interactivity="using:Microsoft.Xaml.Interactivity"
xmlns:core="using:Microsoft.Xaml.Interactions.Core"
```

```
<Image Source="{Binding ImagePath}" [...]>
  <interactivity:Interaction.Behaviors>
    <core:EventTriggerBehavior EventName="Tapped">

      <core:NavigateToPageAction TargetPage="Cookbook.Views.RecipeDetailView"
        Parameter="{Binding Id}" />

    </core:EventTriggerBehavior>
  </interactivity:Interaction.Behaviors>
</Image>
```

Back Button

- App.xaml.cs

```
// after Window.Current.Content  = rootFrame
SystemNavigationManager.GetForCurrentView().BackRequested += App_BackRequested;

// in event handler App_BackRequested:
Frame rootFrame = Window.Current.Content as Frame;

if (rootFrame.CanGoBack)
{
    e.Handled = true;
    rootFrame.GoBack();
}
```

- rootFrame.Navigated +=

```
SystemNavigationManager.GetForCurrentView().AppViewBackButtonVisibility =
    ((Frame)sender).CanGoBack ?
    AppViewBackButtonVisibility.Visible :
    AppViewBackButtonVisibility.Collapsed;
```

Live Tiles

- Mention the different type of Live Tiles
- See Tile Template Catalog
- Adaptive Tiles, see Notifications Visualizer

```
var updater = TileUpdateManager.CreateTileUpdaterForApplication();
//updater.EnableNotificationQueue(true);

XmlDocument xml = TileUpdateManager.GetTemplateContent(TileTemplateType.TileSquare150x150Ima
var all = xml.GetXml();
var image = xml.GetElementsByTagName("image")[0] as XmlElement;
```

```
image.SetAttribute("src", person.Photo);
updater.Update(new TileNotification(tileDefinition));
```

Sensors

- Windows.Devices.Sensor.Accelerometer
- Null-Check
- ReadingChanged-Event

```
var captureUI = new Windows.Media.Capture.CameraCaptureUI();

try
{
    var file = await captureUI.CaptureFileAsync(Windows.Media.Capture.CameraCaptureUIMode.Ph

    var folder = Windows.Storage.ApplicationData.Current.LocalFolder;

    var picturesFolder = Windows.Storage.KnownFolders.PicturesLibrary;
    await file.CopyAsync(picturesFolder);

    var dialog = new Windows.UI.Popups.MessageDialog("Saved");
    await dialog.ShowAsync();


}
catch (Exception)
{
    throw;
}
```

Responsive Layout and Device awareness

SplitView

```
<SplitView IsPaneOpen="True" DisplayMode="Inline" Name="splitView" >
```

- Hamburger-Button
- See (Segoe MDL2 FontFamily)[https://msdn.microsoft.com/en-us/windows/uwp/style/segoe-ui-symbol-font]
- (UWP-App for Segoe MDL 2)[https://www.microsoft.com/en-us/store/p/uwp-segoe-mdl2-assets/9nblggh5fzpm]
- Switch IsPaneOpen in Event-Handler or ViewModel

```
<Button FontFamily="Segoe MDL2 Assets" Content="&#xE700;" Click="Button_Cl" Width="48"/>
```

Visual State Triggers

- SplitView

```
<SplitView IsPaneOpen="False" DisplayMode="Overlay" Name="splitView" >
```

```
<VisualStateManager.VisualStateGroups>
    <VisualStateGroup x:Name="WindowSizeStates">
        <VisualState x:Name="WideState">
            <VisualState.StateTriggers>
                <AdaptiveTrigger MinWindowWidth="1024" />
            </VisualState.StateTriggers>
            <VisualState.Setters>
                <Setter Target="splitView.IsPaneOpen" Value="True" />
                <Setter Target="splitView.DisplayMode" Value="Inline" />
            </VisualState.Setters>
        </VisualState>
    </VisualStateGroup>
</VisualStateManager.VisualStateGroups>
```

Create custom state Triggers

```
public class ContinuumStateTrigger : StateTriggerBase
{
    public ContinuumStateTrigger()
    {
        Window.Current.SizeChanged += Current_SizeChanged;
    }

    private void Current_SizeChanged(object sender, WindowSizeChangedEventArgs e)
    {
        var mode = UIViewSettings.GetForCurrentView().UserInteractionMode;
        this.SetActive(mode == UserInteractionMode.Touch);
    }
}
```

RelativePanel

Capability Check

- Add Reference > Universal Windows > Extensions > Mobile Extensions
- ApiInformation.IsTypePresent
  - Windows.Phone.UI.Input.HardwareButtons

- Windows.Phone.Devices.Notification.VibrationDevice

```
Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.UI.Input.HardwareBut
```

DeviceFamily

- Create sub-folder "DeviceFamily-Mobile"
  - Mobile, Desktop, Team, Xbox
- Create XAML View instead of Page, Codebehind is shared
- Resources with Style depending on device family
- Resources: "Logo.DeviceFamily-Mobile.png"
  - ms-appx:///Images/Logo.png

```
var deviceFamily = AnalyticsInfo.VersionInfo.DeviceFamily;
```

Hosted Web App

```
function cameraCapture() {

  alert('Hi!');

  if(typeof Windows != 'undefined') {
   var captureUI = new Windows.Media.Capture.CameraCaptureUI();

   //Set the format of the picture that's going to be captured (.png, .jpg, ...)
   captureUI.photoSettings.format = Windows.Media.Capture.CameraCaptureUIPhotoFormat.png;

   //Pop up the camera UI to take a picture
   captureUI.captureFileAsync(Windows.Media.Capture.CameraCaptureUIMode.photo).then(functior
   });
  }
}
```

Suspend / Resume

- Register for **Application.Current.Suspending** event in OnNavigatedTo method of page - and unregister in OnNavigatedFrom
- use ApplicationData.Current.LocalSettings.Values["..."] or file storage
- react to stored state in App.xaml.cs (OnLaunched)
- Test suspend/resume with Visual Studio Debug feature

Cortana

- Video from Cortana talk

Design and User Experience Guidelines

- Microsoft Downloads