Android, iOS and Hybrid Applications

Mobile-Development

Day 2

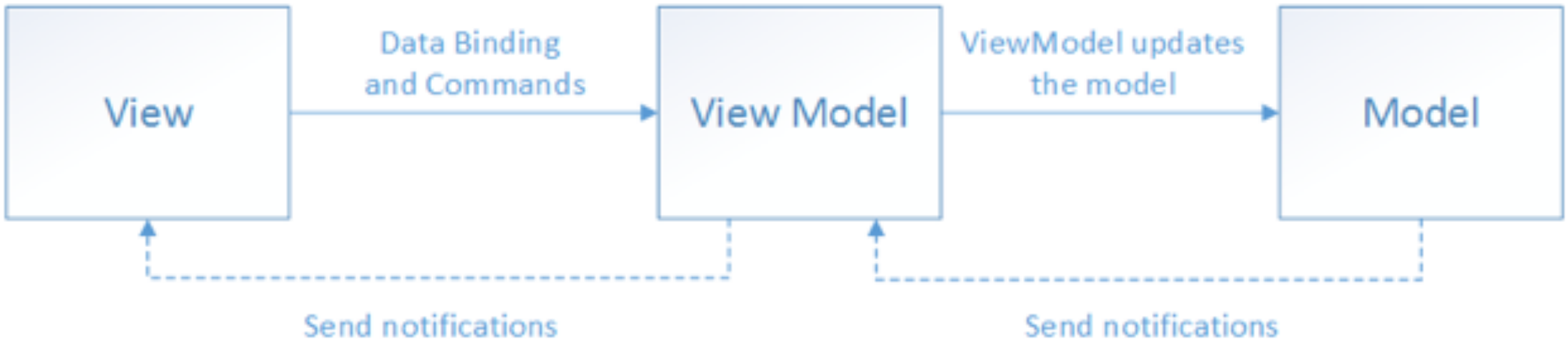
- MVVM
- XAML for Forms
- Controls and differences to WPF
- Bindings
- Commands

MVVM Pattern

- Model View ViewModel
- Decouple the logic from the views
- Use Bindings to connect them
- It’s hard to test code-behind or controllers in UnitTests
- A ViewModel generally doesn’t know his view

https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm

MVVM PATTERN



https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm

ViewModels

- The logical part should be in here
  - whether a control is enabled or not
  - what selections are available in lists
  - what is executed when clicking something

ViewModels – Notifying the View

- INotifyPropertyChanged
- Create a base class for ViewModels

```
protected void RaisePropertyChanged([CallerMemberName] string property = "")
{
  PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(property));
}

public bool IsSelected
{
  get => _isSelected;
  set
  {
    _isSelected = value;
    RaisePropertyChanged();
  }
}
```

https://docs.microsoft.com/en-us/dotnet/api/system.componentmodel.inotifypropertychanged?view=netstandard-2.0

ViewModels – Lists

- ObservableCollection<T>
- INotifyCollectionChanged

```
public ObservableCollection<TodoItemViewModel> Items
{
  get => _items;
  set
  {
    _items = value;
    RaisePropertyChanged();
  }
}
```

https://docs.microsoft.com/en-us/dotnet/api/system.collections.specialized.inotifycollectionchanged?view=netstandard-2.0

Setup the Basics

- Setup your ViewModels
- Add the necessary properties
- Raise PropertyChanged where necessary

Xamarin Forms

- Page - represents a Screen
  - ContentPage
  - MasterDetailPage
  - TabbedPage

https://docs.microsoft.com/en-us/xamarin/xamarin-forms/user-interface/controls/pages

XAMARIN FORMS – LAYOUT

- Layouts - Subtypes of View
  - ContentView - ContentControl
  - Frame - Border
  - ScrollView - ScrollViewer
  - StackLayout - StackPanel
  - Grid
  - FlexLayout - based on the CSS flexbox-model

https://docs.microsoft.com/en-us/xamarin/xamarin-forms/user-interface/controls/layouts

Xamarin Forms – Controls

- Label
- Button
- Entry - TextBox
- ListView - List
- Picker - ComboBox
- DatePicker

https://docs.microsoft.com/en-us/xamarin/xamarin-forms/user-interface/controls/index

Xamarin Forms – Controls Examples

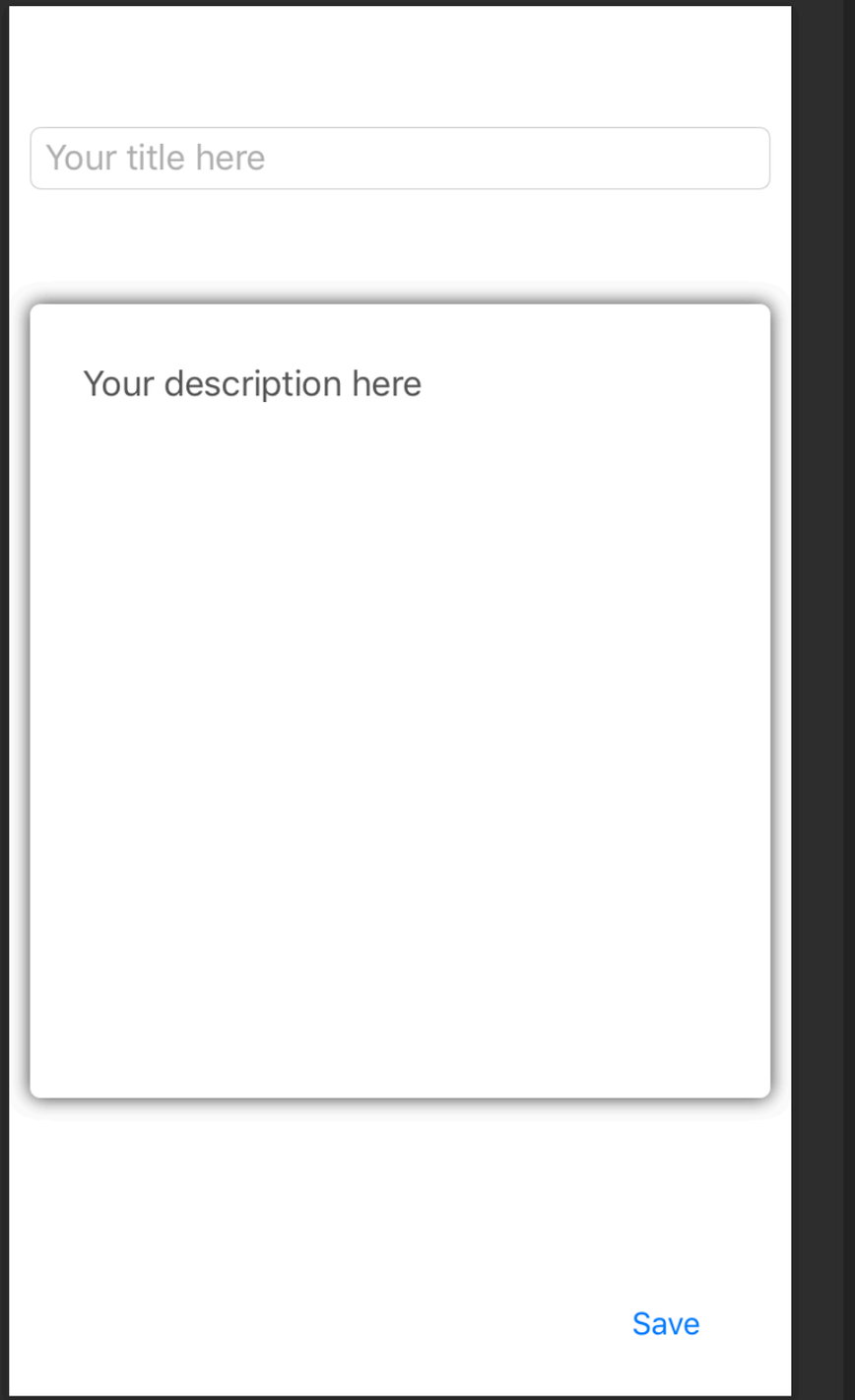
```
<Grid Margin="10">
    <Grid.RowDefinitions>
        <RowDefinition Height="*" />
        <RowDefinition Height="3*" />
        <RowDefinition Height="*" />
    </Grid.RowDefinitions>

    <Entry Grid.Row=“0" VerticalOptions=“Center" Placeholder="Your title here” />

    <Frame Grid.Row="1">
        <Editor VerticalOptions=“FillAndExpand" Placeholder="Your description here” />
    </Frame>

    <Button Grid.Row=“2" HorizontalOptions=“End" VerticalOptions=“End"
            WidthRequest=“100" HeightRequest=“50" Text="Save" />
</Grid>
```

XAMARIN FORMS – CONTROLS EXAMPLES



XAML Standard

- Currently in preview
- Unify XAML syntax between platforms
- WPF, Xamarin and UWP

https://docs.microsoft.com/en-us/xamarin/xamarin-forms/xaml/standard/

https://github.com/Microsoft/xaml-standard

Practice

- Example
- Extend your app with the necessary controls

XAMARIN FORMS – BINDINGS

- Similar to WPF
- Define the “BindingContext” (ViewModel)
- {Binding Path=Value} Syntax
- BindingMode like WPF (Default, One, Two)
- INotifyPropertyChanged
- IValueConverter to convert “on the fly”

https://docs.microsoft.com/en-us/xamarin/xamarin-forms/app-fundamentals/data-binding/basic-bindings

XAMARIN FORMS – BINDINGS

- No support for RelativeSource
- BindingContext is the “Source”
- BindingContext is inherited
- x:Reference to reference a control by name

Xamarin Forms – "Compiled Bindings"

- Will trigger compile errors
- Improve performance
- Use the x:DataType on a Visual to set the context

https://docs.microsoft.com/en-us/xamarin/xamarin-forms/app-fundamentals/data-binding/compiled-bindings

Xamarin Forms – Commands

- If you want to execute something
- Use the Command class with a delegate
- IsEnabled will be set by CanExecute

```
<Button Command="{Binding Path=MyCommand}" Text="Click me" />
```

```
AddTodoCommand = new Command(() => ..., () => ...);
```

https://docs.microsoft.com/en-us/xamarin/xamarin-forms/app-fundamentals/data-binding/commanding

XAMARIN FORMS – COMMANDS WITH PARAMETERS

- Use the generic Command class
- Make sure to set the CommandParameter in the code

```
<Button Command="{Binding Path=MyCommand}" CommandParameter="1" Text="Click" />

AddTodoCommand = new Command<int>((parameter) => ..., (parameter) => ...);
```

XAMARIN FORMS – VIEWMAPPER

- Simple Service to map a ViewModel to a View

```
public Page Map<TViewModel>(TViewModel viewModel) where TViewModel : class
{
  Page result;
  switch (viewModel)
  {
    case MainViewModel mainViewModel:
      result = new MainPage { BindingContext = mainViewModel };
      break;
    case TodoListViewModel listViewModel:
      result = new TodoListPage { BindingContext = listViewModel };
      break;
    case TodoItemViewModel detailViewModel:
      result = new TodoDetailPage { BindingContext = detailViewModel };
      break;
    default:
      throw new InvalidNavigationException($"Could not map type [{typeof(TViewModel)}]");
  }

  return result;
}
```

XAMARIN FORMS – IVALUECONVERTER

- Change the value of a binding for the control
- Implement the IValueConverter interface

```
<ContentPage.Resources>
  <ResourceDictionary>
    <local:IntToBoolConverter x:Key="Converter" />
  </ResourceDictionary>
</ContentPage.Resources>

<Button
    IsEnabled="{Binding Path=SomeValue, Converter={StaticResource Converter}}”
    Text="Click me" />
```

https://docs.microsoft.com/en-us/xamarin/xamarin-forms/app-fundamentals/data-binding/converters

XAMARIN FORMS – LISTVIEW

- ItemsSource –> use ObservableCollection<T>
- ItemTemplate to override appearance
- SelectedItem to listen for selection

https://docs.microsoft.com/en-us/xamarin/xamarin-forms/user-interface/listview/

Xamarin Forms – ListView

```
<ListView ItemsSource="{Binding Path=MyList}">
    <ListView.ItemTemplate>
        <DataTemplate>
            <ViewCell>
                <StackLayout>
                    <Label Text="{Binding Firstname}" />
                    <Label Text="{Binding Lastname}" />
                </StackLayout>
            </ViewCell>
        </DataTemplate>
    </ListView.ItemTemplate>
</ListView>
```

Practice

- Example
- Extend your app with the basic bindings
- Apply commands for the navigation