Diving into MVVM

Gill Cleeren
ARCHITECT

@gillcleeren www.snowball.be

Agenda

The MvvmCross framework

Data binding

The MVVM pattern

The MvvmCross Framework

MvvmCross

is a cross-platform MVVM framework that enables developers to create cross platform apps

From mvvmcross.com

Hello MvvmCross!

Open-source MVVM framework

Promotes code-sharing up until the ViewModel layer

Covers Android, iOS, Windows, Mac and Xamarin.Forms

Core Features of MvvmCross

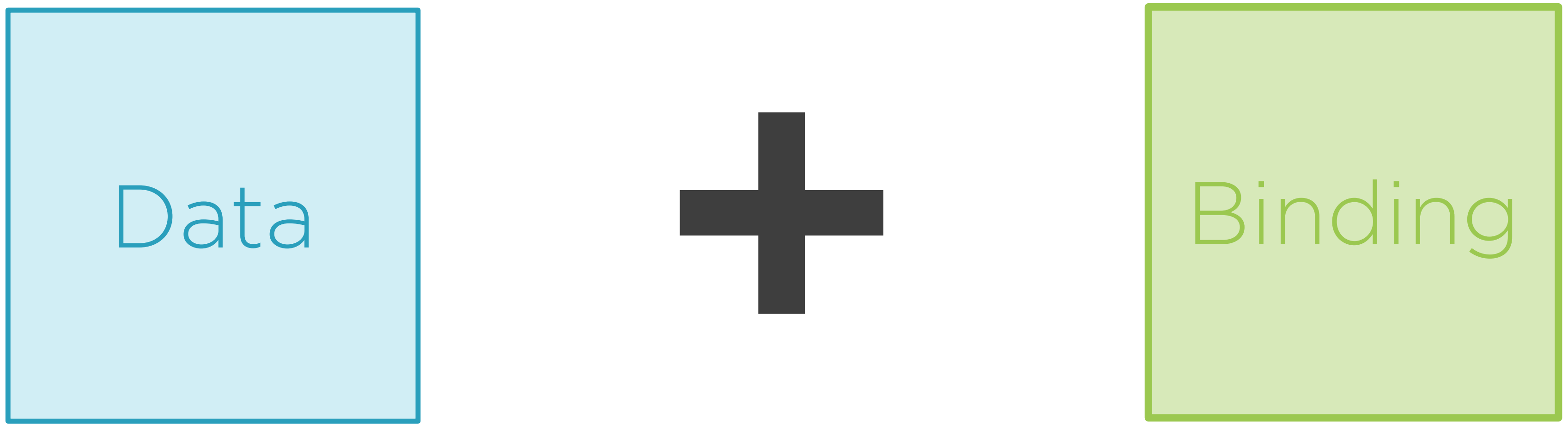
- Data binding
- IOC
- Plug-ins

Demo

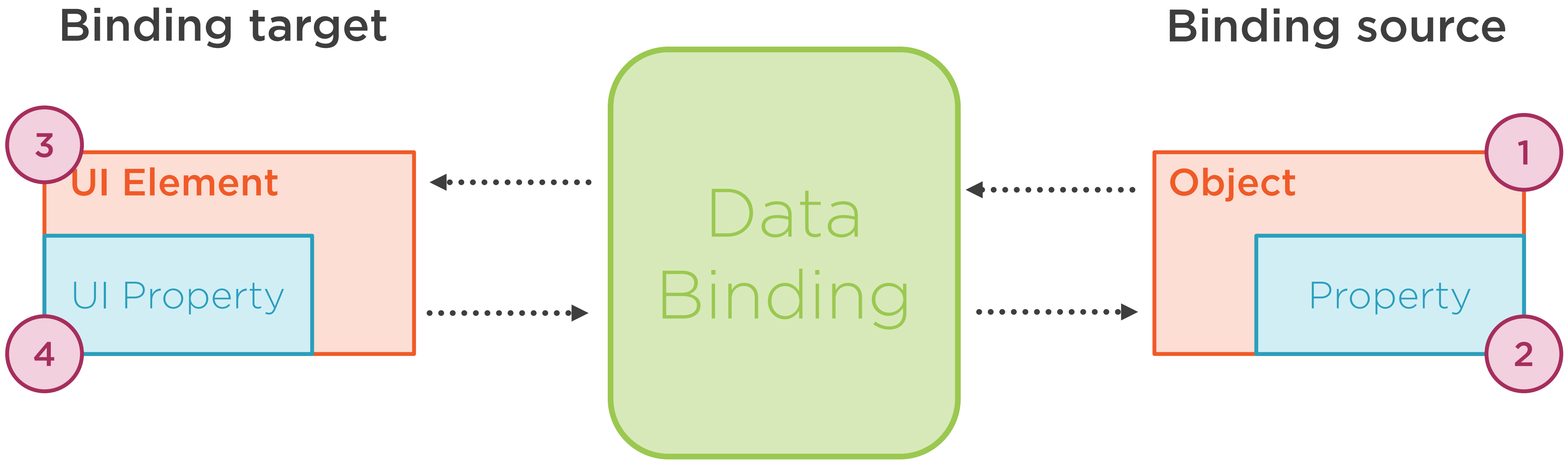
Taking a Look at a Basic MvvmCross Application

Data Binding

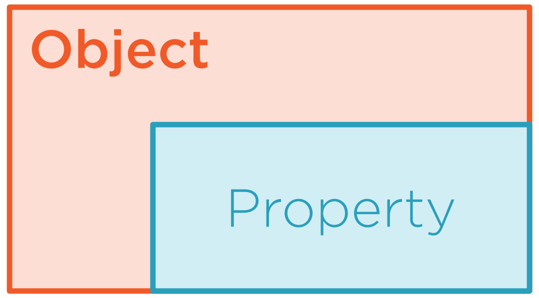
Data Binding



Data Binding



Binding source



Single object or collection

Not related to database

Part of shared code

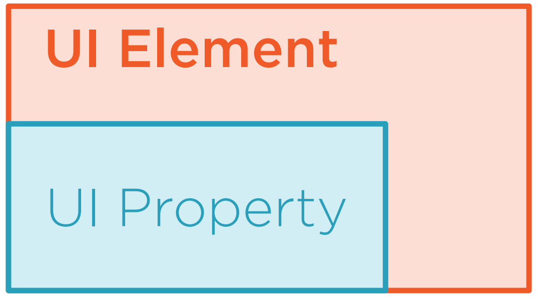
Data Binding

Removes need for boiler-plate code

Synchronization

Conversion

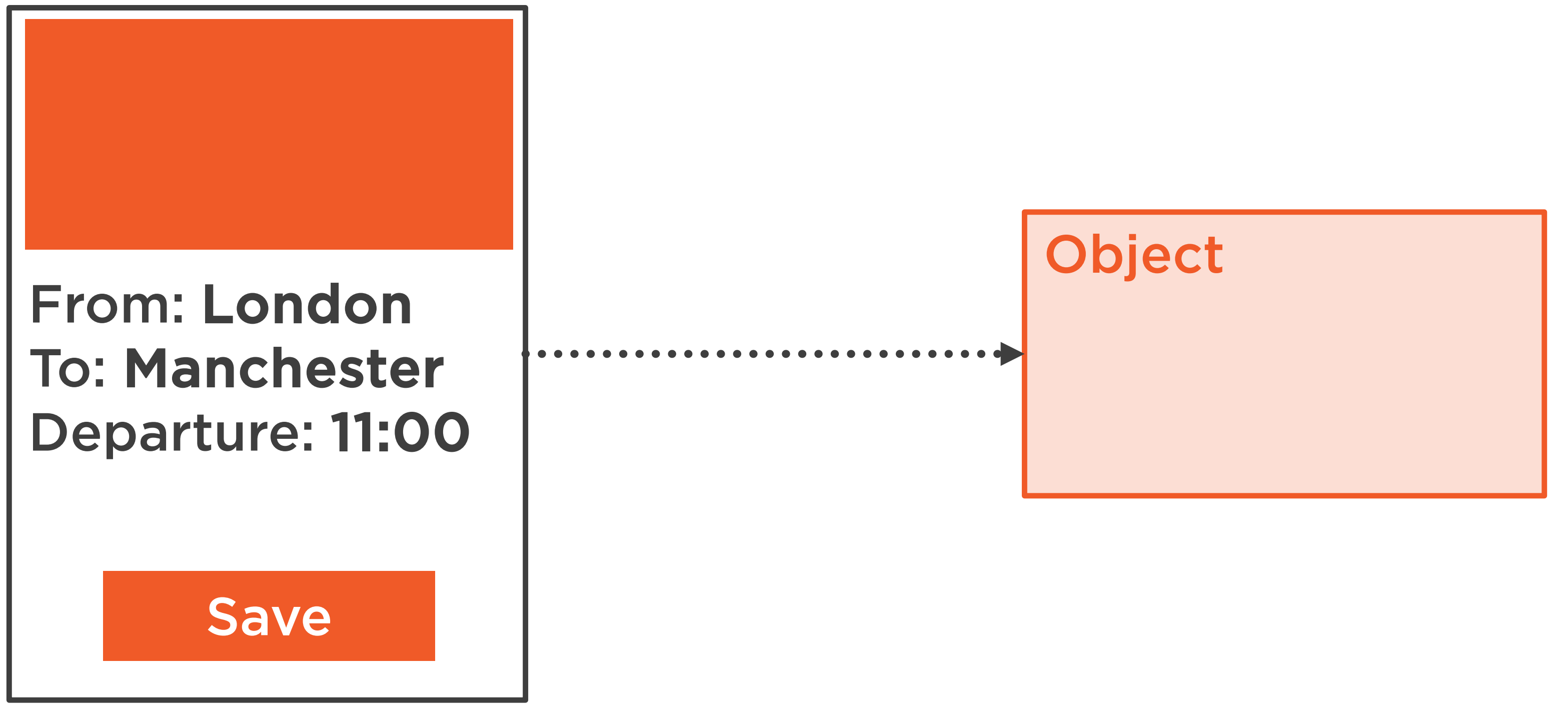
Binding target



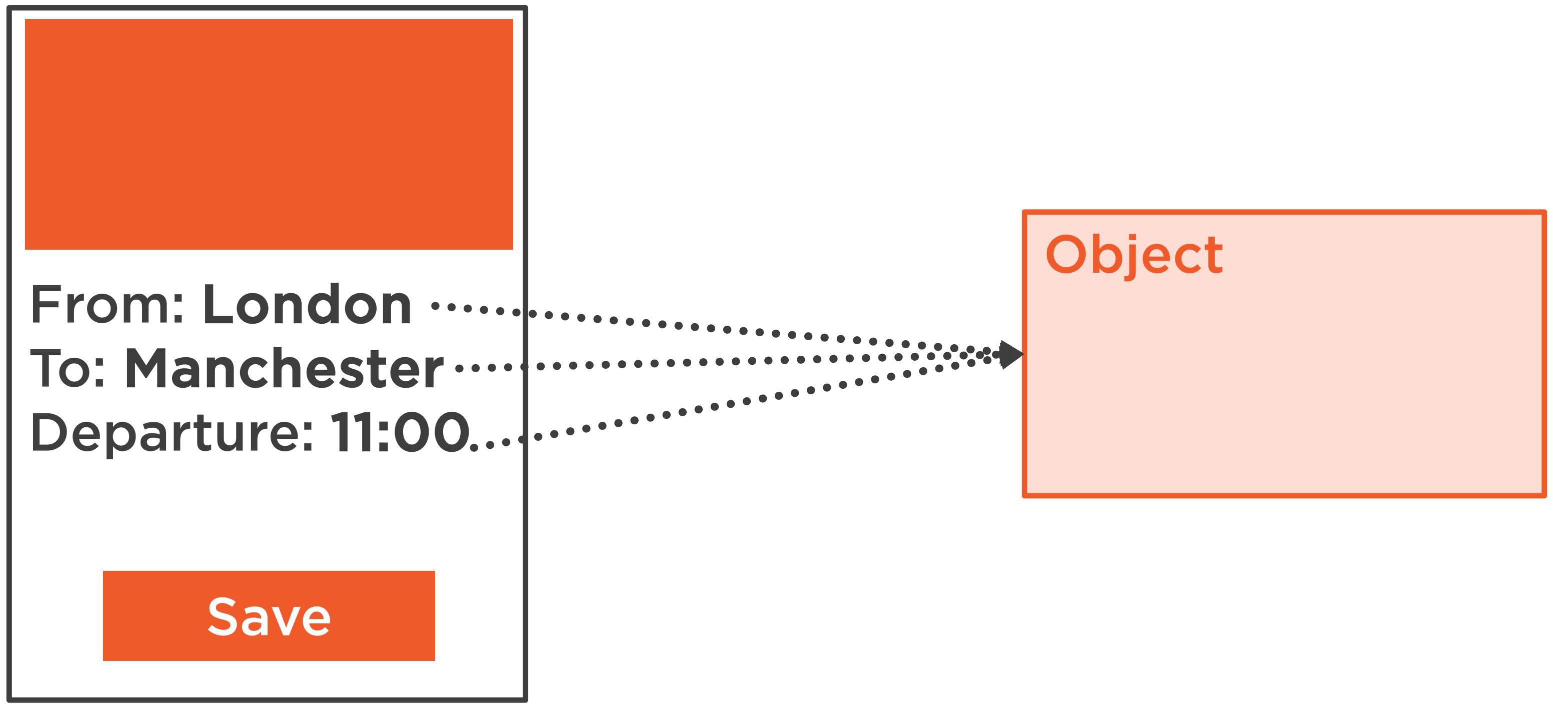
UI Element

Part of the Android & iOS apps

Data Context



Binding to a Single Object



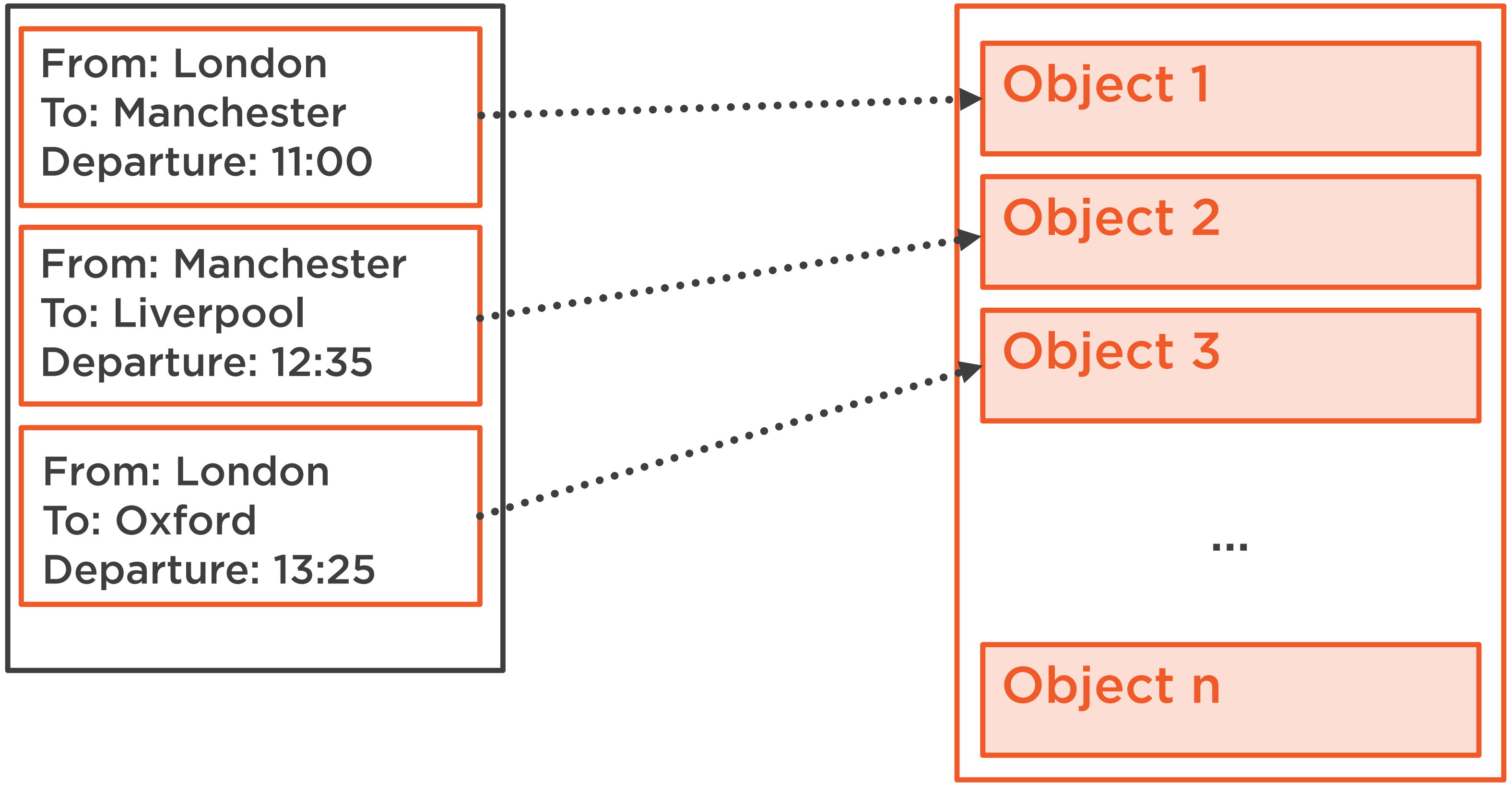
Data Binding in Android

```
<TextView
    android:text="Departure Date“
    android:layout_width="wrap_content“
    android:layout_height="wrap_content“
    android:id="@+id/DepartureDateTextViewValue“
    android:textColor="#00d8cc“
    android:textSize="18dp“
    local:MvxBind="Text SelectedJourney.JourneyDate" />
```

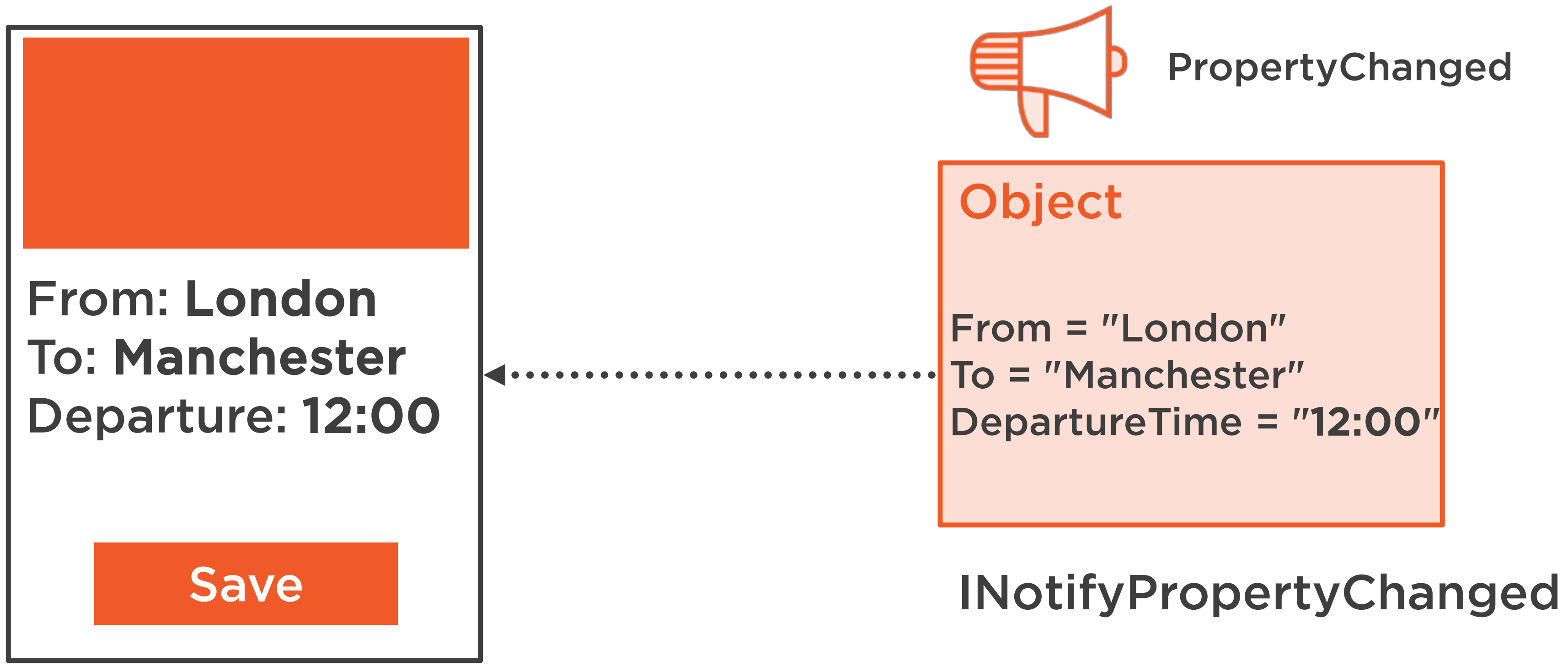
Data Binding in iOS

```
var set = this.CreateBindingSet<JourneyDetailView,
JourneyDetailViewModel>();


set.Bind(DepartureDateLabel)
   .To(vm => vm.SelectedJourney.JourneyDate));
```

Binding to a Collection



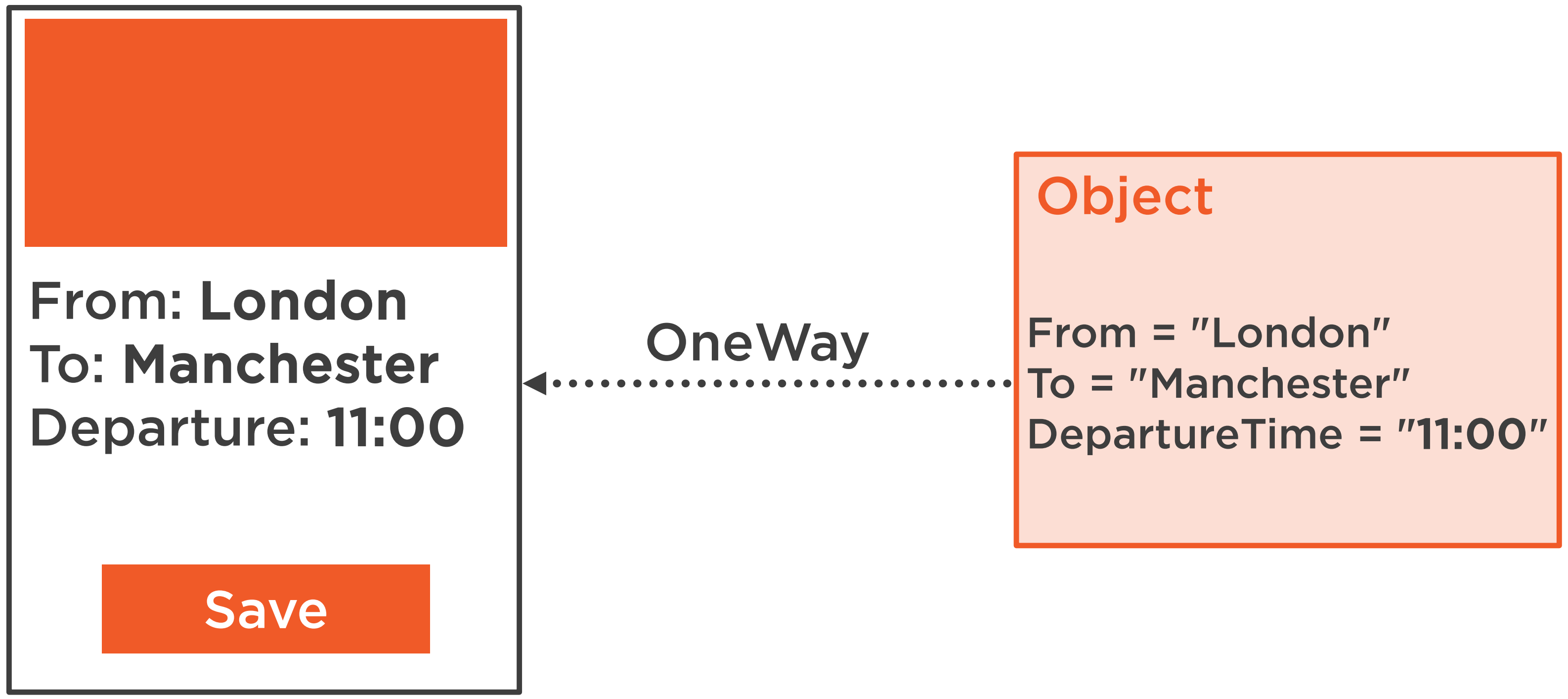
Change Notifications



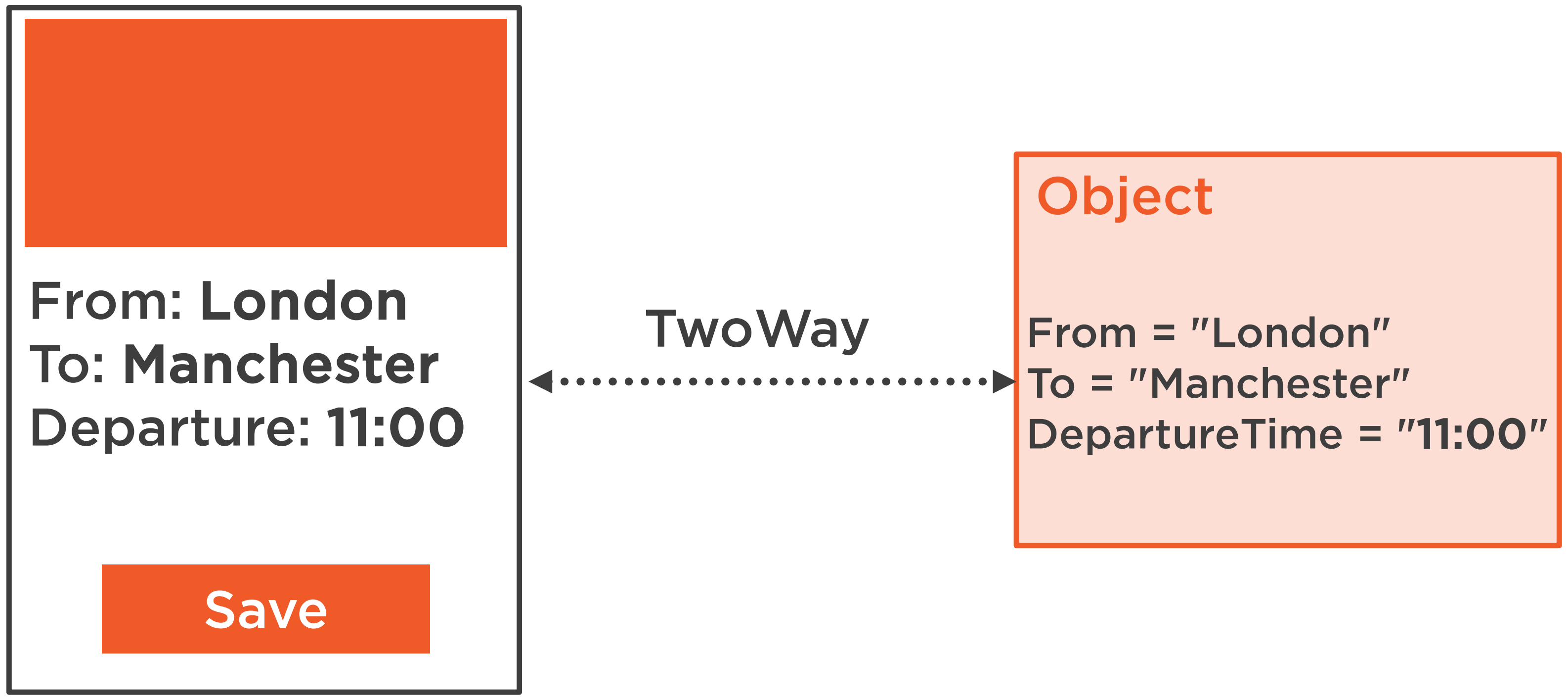
INotifyPropertyChanged Example

```
public class MyModel : INotifyPropertyChanged
{

    private string _myProperty;
    public string MyProperty
    {
        get { return _myProperty; }
        set
        {
            _myProperty = value;
            PropertyChanged
                (this, new PropertyChangedEventArgs("MyProperty");

        }
    }
    public event PropertyChangedEventHandler PropertyChanged;

}
```

Binding Modes



Binding Modes



Binding Modes

One-Way-To-Source

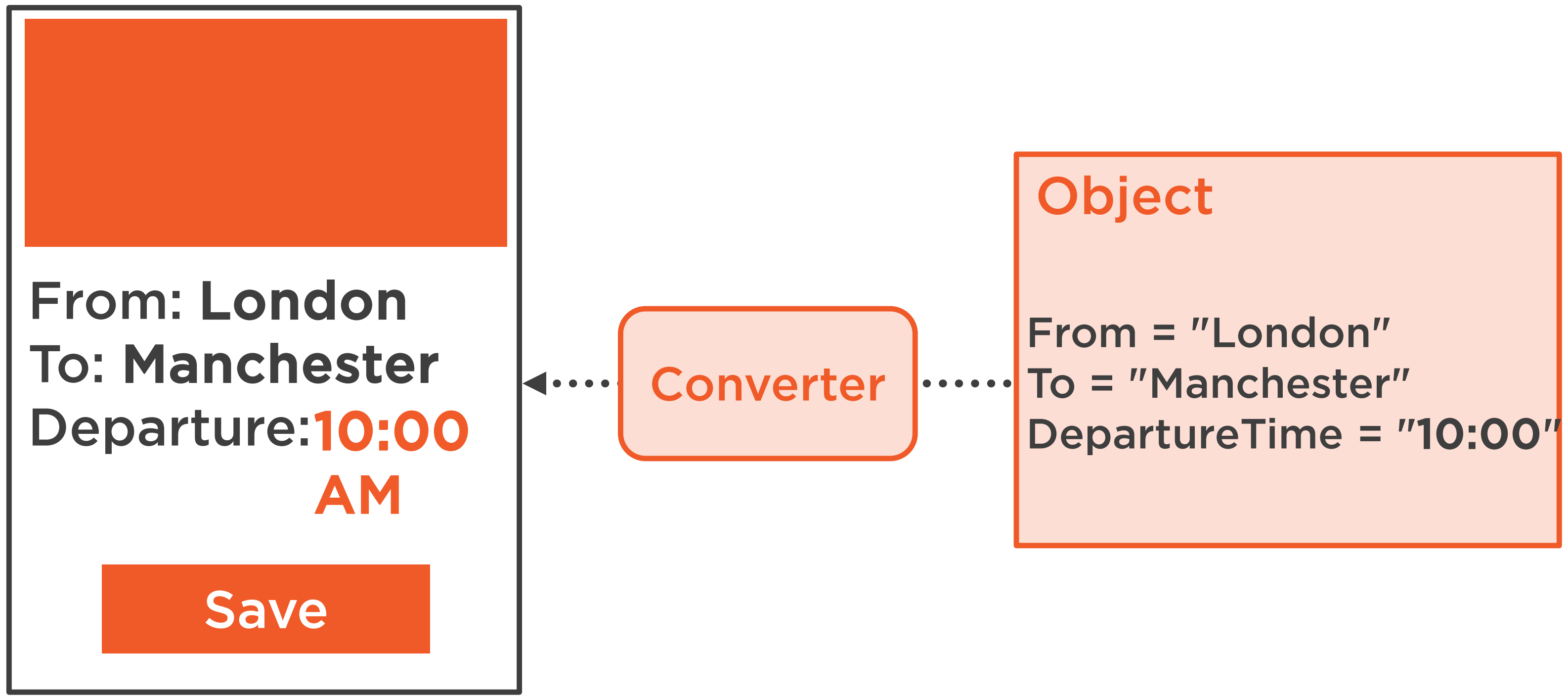
```
<EditText
    local:MvxBind="Text NumberOfTravellers, Mode=TwoWay" />
```

Binding Modes in Android

```
set.Bind(NumberOfTicketsTextField)
    .To(vm => vm.NumberOfTravellers).TwoWay();
```

Binding Modes in iOS

Converters


From: **London**
To: **Manchester**
Departure:**10:00 AM**
Save
Converter
Object
From = "London"
To = "Manchester"
DepartureTime = "10:00"

Creating a Converter

```
public class DateTimeToDayConverter :
MvxValueConverter<DateTime, string>

{

    protected override string Convert(DateTime value, Type
        targetType, object parameter, CultureInfo culture)
    {

        return value.ToString("MMM dd, yyyy");

    }
}
```

```
<TextView
    local:MvxBind="Text SelectedJourney.JourneyDate,
    Converter=DateTimeToDay" />
```

Using a Converter on Your Data Binding

Demo

Data Binding Using MvvmCross

The MVVM Pattern

Hello MVVM

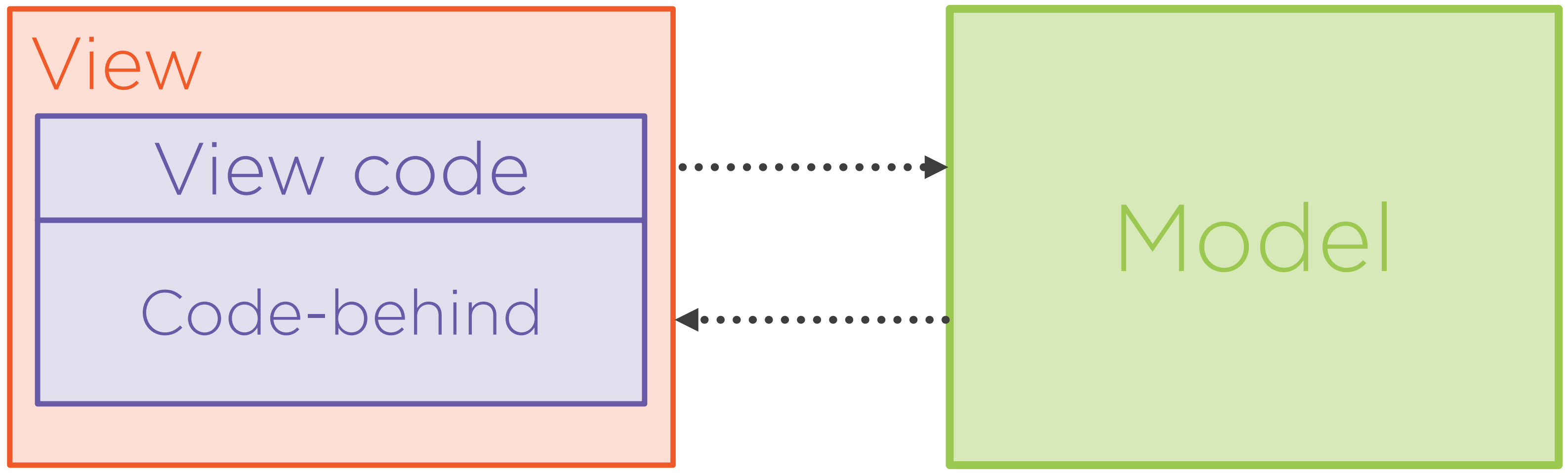
Architectural pattern

Based on data binding and commanding

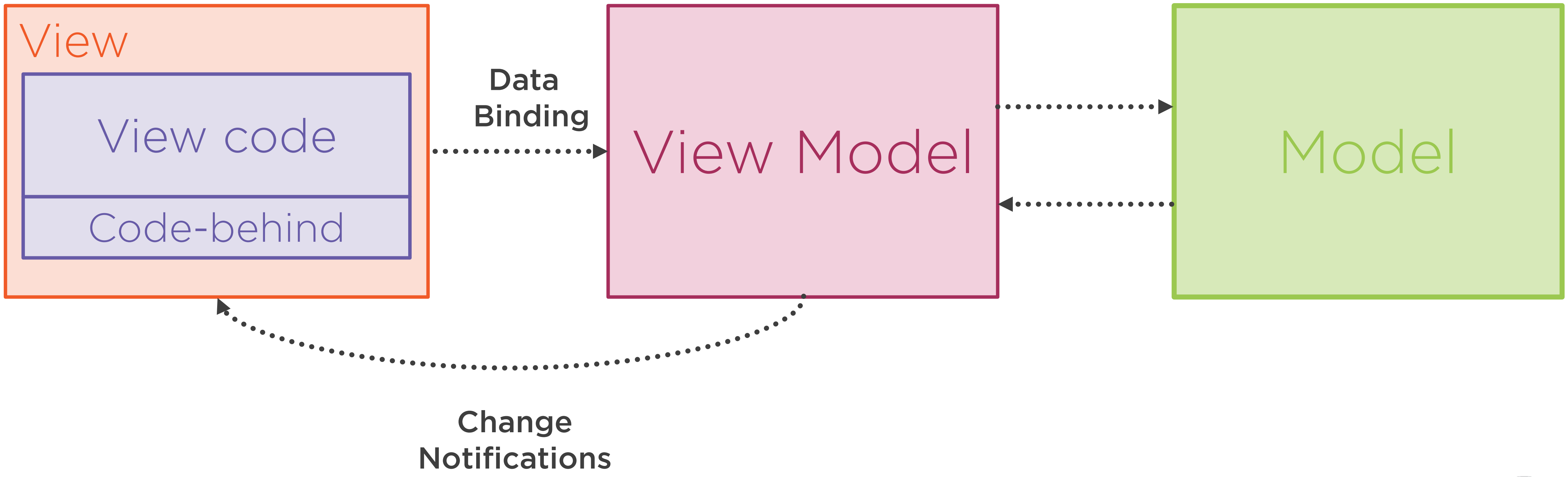
Popular since XAML (WPF, Silverlight…)

Promotes SOC and testability

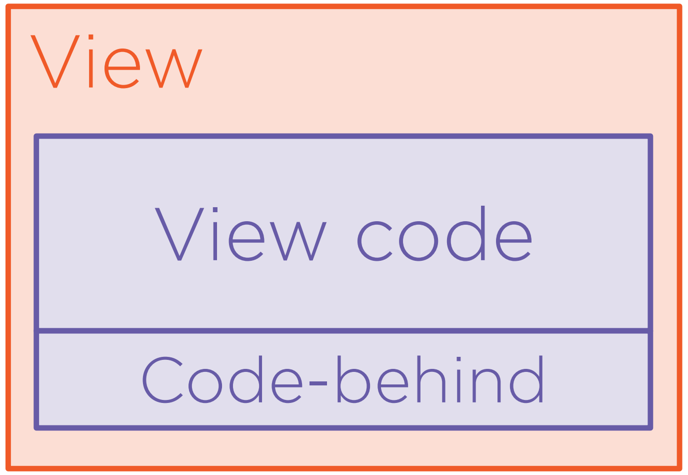
Without MVVM



Architecture of MVVM



Functionalities of the View



User interface

- Activity in Android
- ViewController in iOS
- XAML view in Windows

“Binds” to a view model instance

No business logic

View Code in Android

```
<LinearLayout
    android:orientation="vertical"
    android:id="@+id/mainLinearLayout">
    <TextView
        android:text="Departure Date"
        android:id="@+id/DepartureDateTextViewValue"
        local:MvxBind="Text SelectedJourney.JourneyDate,
            Converter=DateTimeToDay" />
```

Activity or Fragment Code

```
[Activity(Label = "Main Activity")]
public class MainActivity :
    MvxCachingFragmentCompatActivity<MainViewModel>
{
    public MainViewModel ViewModel
    {
        get { return (MainViewModel)base.ViewModel; }
        set { base.ViewModel = value; }
    }

    protected override void OnCreate(Bundle savedInstanceState)
    {
        base.OnCreate(savedInstanceState);
        SetContentView(Resource.Layout.MainView);
        ...

    }

}
```

Functionalities of the View Model

View Model

Glue between view and model

Expose state and operations

Handle flow of the application

Testable

No reference to UI elements

Shared View Model in MvvmCross

```
public class JourneyDetailViewModel : BaseViewModel,
IJourneyDetailViewModel
{
    public Journey SelectedJourney              State
    {
        get { return _selectedJourney; }
        set
        {
            _selectedJourney = value;
            RaisePropertyChanged(() => SelectedJourney);
        }

    }

    public MvxCommand CloseCommand              Operations
    {
        get { return new MvxCommand(() => Close(this)); }
    }
}
```

Functionalities of the Model

Data model and services

Return data

Demo

Looking at a View Model With MvvmCross

```
public interface ICommand
{
    event EventHandler CanExecuteChanged;
    bool CanExecute(object parameter);
    void Execute(object parameter);
}
```

Commanding

Summary

Data binding helps with data-intensive interfaces

MVVM pattern is a great foundation

MvvmCross as starting point