Creating the Foundation for the Application

Gill Cleeren
ARCHITECT

@gillcleeren www.snowball.be

Agenda

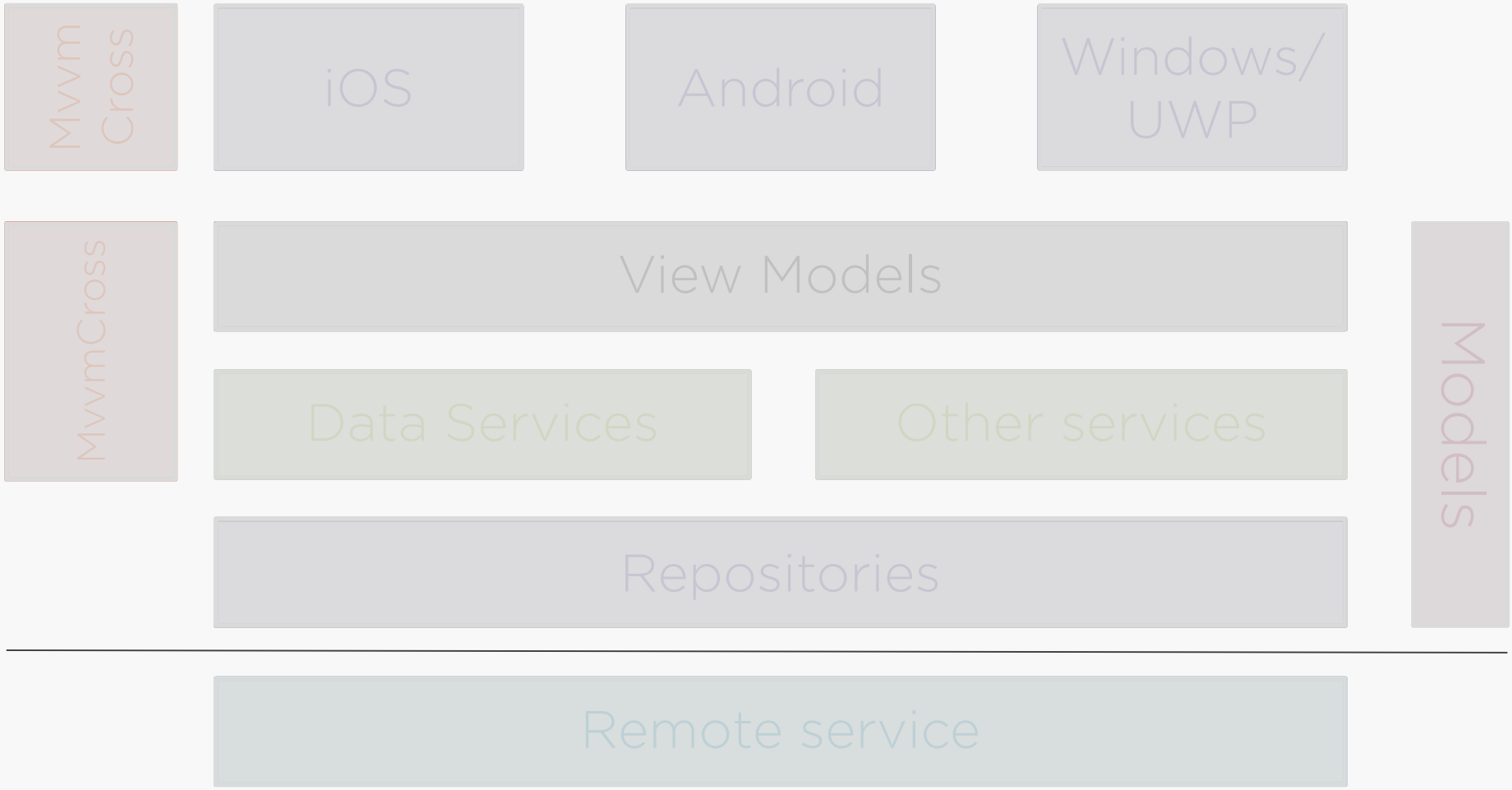
Application architecture overview

MvvmCross core functionality

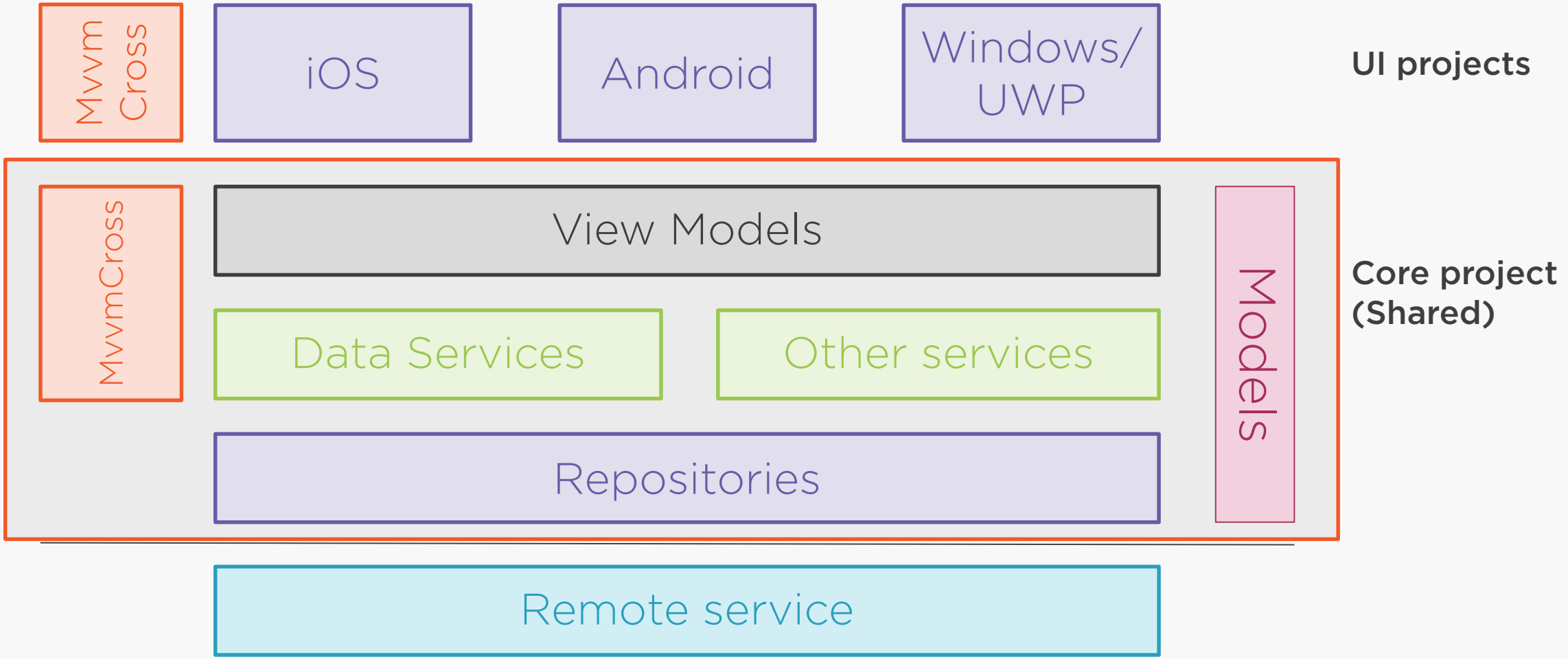
Exploring the Core project

Application Architecture

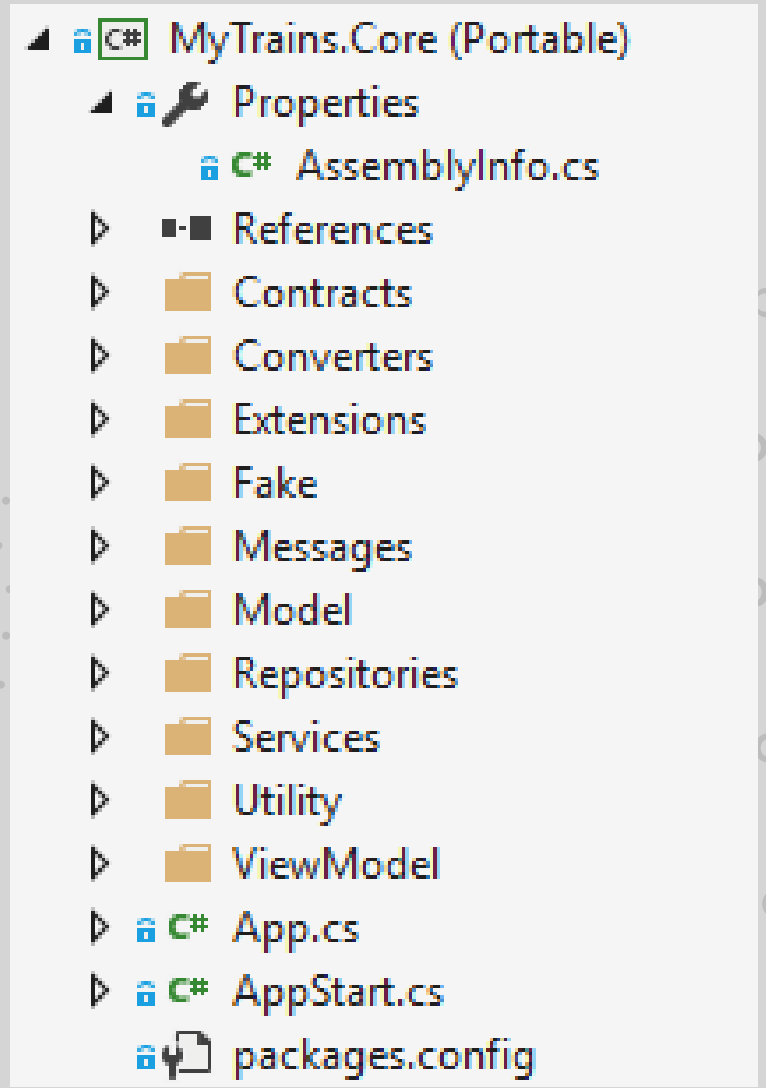
Architecture of the Solution



Topic of This Module



Solution and Projects



Code Sharing in the Core Project

Portable Library

Shared Project

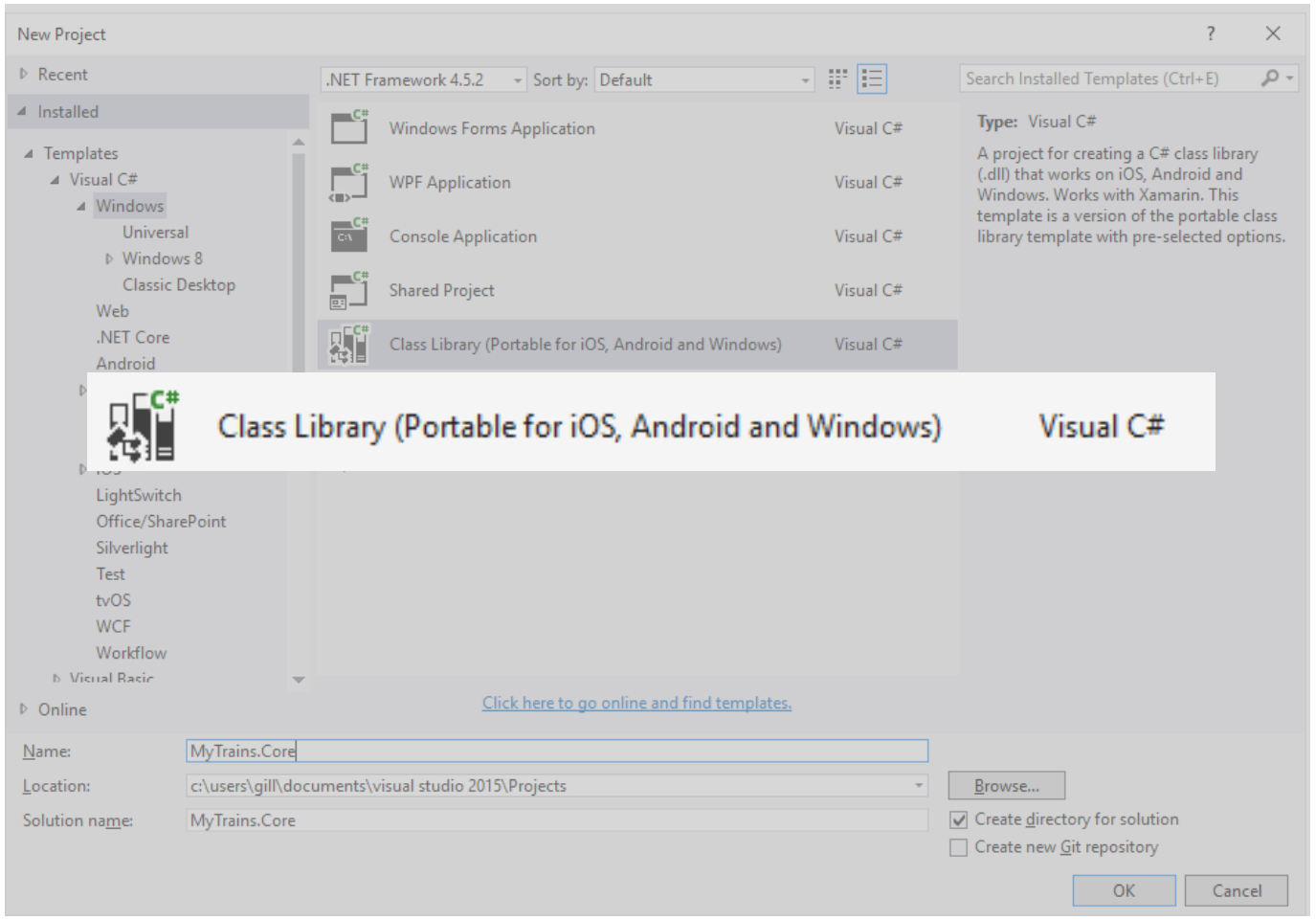
PCL (Portable Class Library)

Library shared across platforms

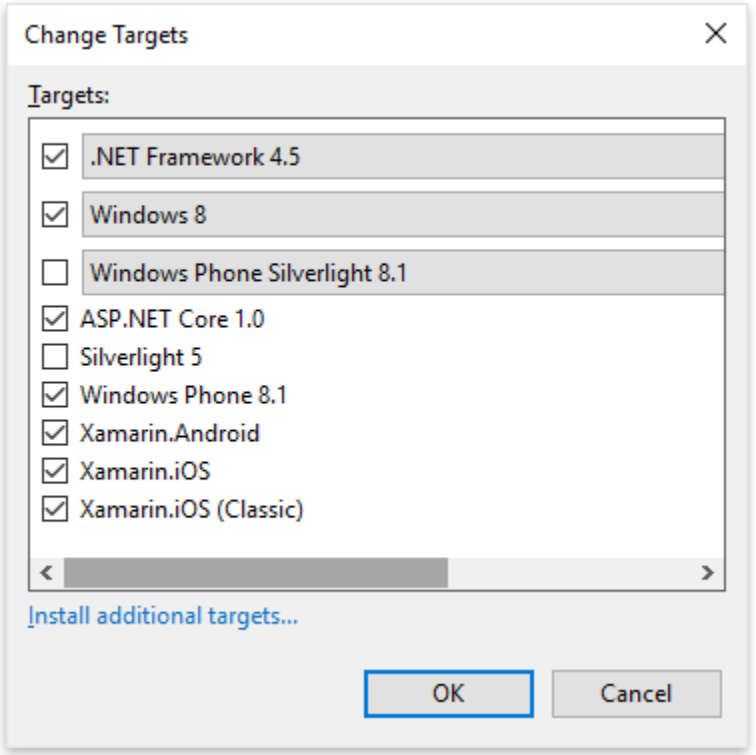
Lowest common denominator

Extendable to other platforms

Creating a PCL



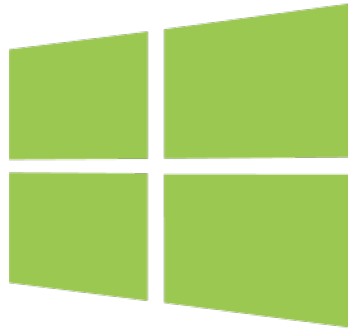
Configuring a PCL



Shared Project

Code is copied to referencing project

Precompiler statements



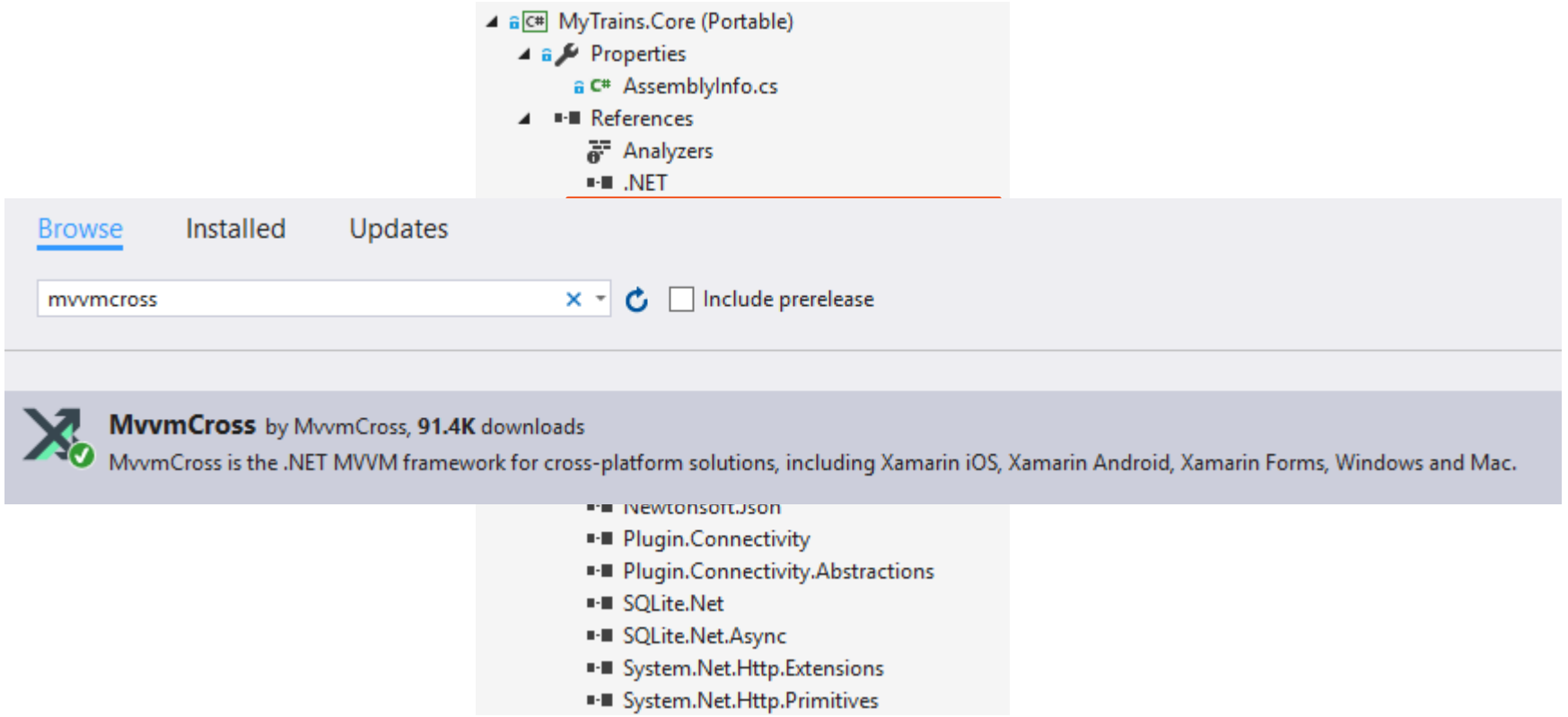
Similar to “old” Universal Projects

We Are Going to Use...

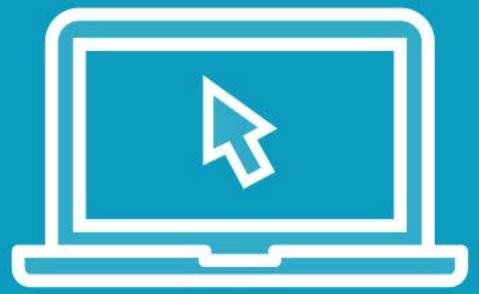
Portable Library

Shared Project

Required NuGet Packages



Demo



Setting up the solution and projects

Adding MvvmCross packages

MvvmCross Core Functionality

Core Building Blocks

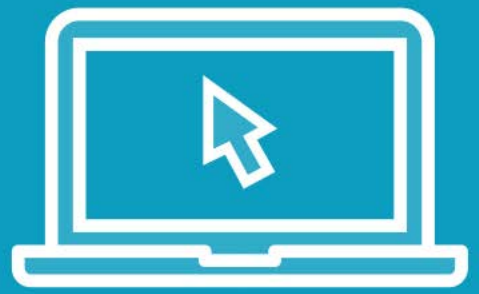
The App Class

```
public class App: MvxApplication
{
    public override void Initialize()
    {
        base.Initialize();

        CreatableTypes()
            .EndingWith("Service")
            .AsInterfaces()
            .RegisterAsLazySingleton();

        RegisterAppStart(new AppStart());
    }
}
```

AppStart.cs

```
public class AppStart: MvxNavigatingObject, IMvxAppStart
{
    public void Start(object hint = null)
    {
        //Application setup code goes here

        ShowViewModel<MainViewModel>();
    }
}
```

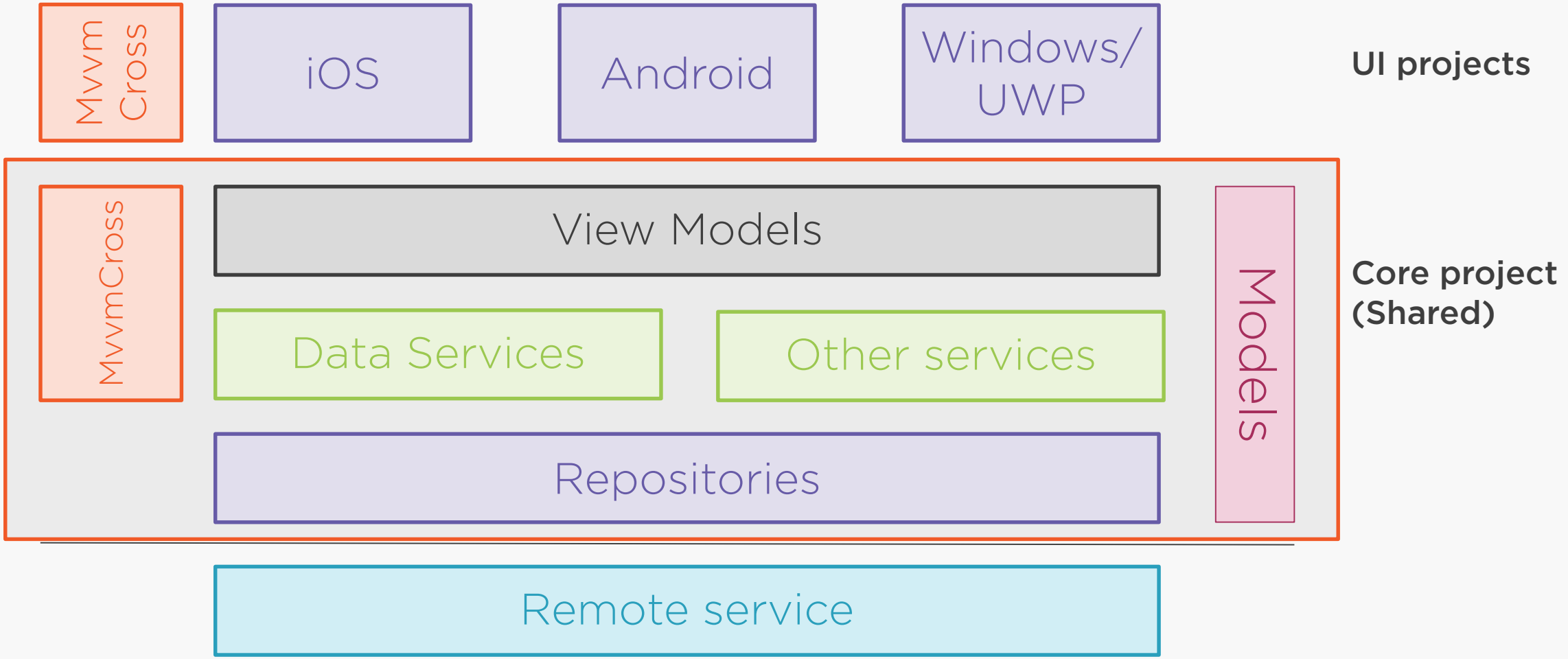
Demo



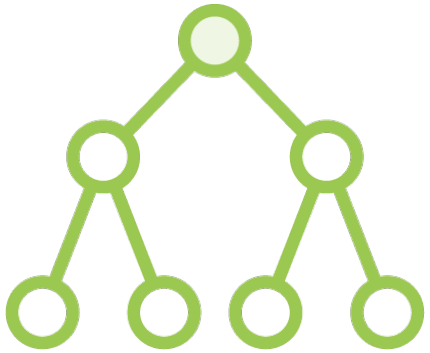
App and AppStart classes

Exploring the Core Project

Topic of This Module

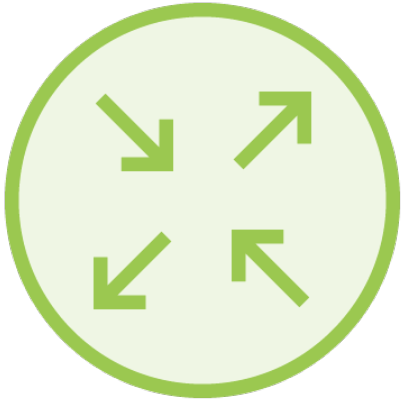


Model Classes



POCO

Client-side model



Mapping may be needed

```
public class User
{
    public int UserId { get; set; }
    public string UserName { get; set; }
    public string Email { get; set; }
    public string Password { get; set; }
}
```

User Model Class

Repositories

Actual data retrieval	Layer between data source and app	Mapping to model classes
Single model	CRUD	Async

Repository

```
public class SavedJourneyRepository
{
    public async Task<IEnumerable<SavedJourney>>
        GetSavedJourneyForUser(int userId)
    {
        ...
    }

    public async Task AddSavedJourney
        (int userId, int journeyId, int numberOfTravellers)
    {
        ...
    }
}
```

Data Services

Mediate between repository and view models

Per unit of functionality

Business rules

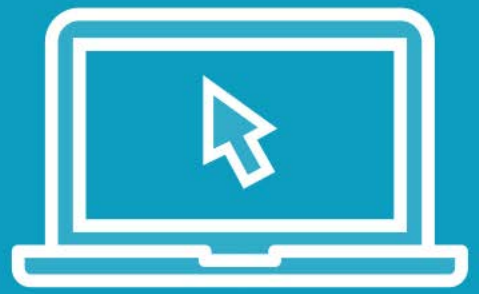
Data Service

```
public class CityDataService
{
    public async Task<List<City>> GetAllCities()
    {
        ...
    }

    public async Task<City> GetCityById(int cityId)
    {
        ...
    }
}
```

Adding Interfaces

```
public interface ICityDataService
{
    Task<List<City>> GetAllCities();
}

public class CityDataService: ICityDataService
{
    public async Task<List<City>> GetAllCities()
    {
       return await _cityRepository.GetAllCities();
    }
}
```

Demo

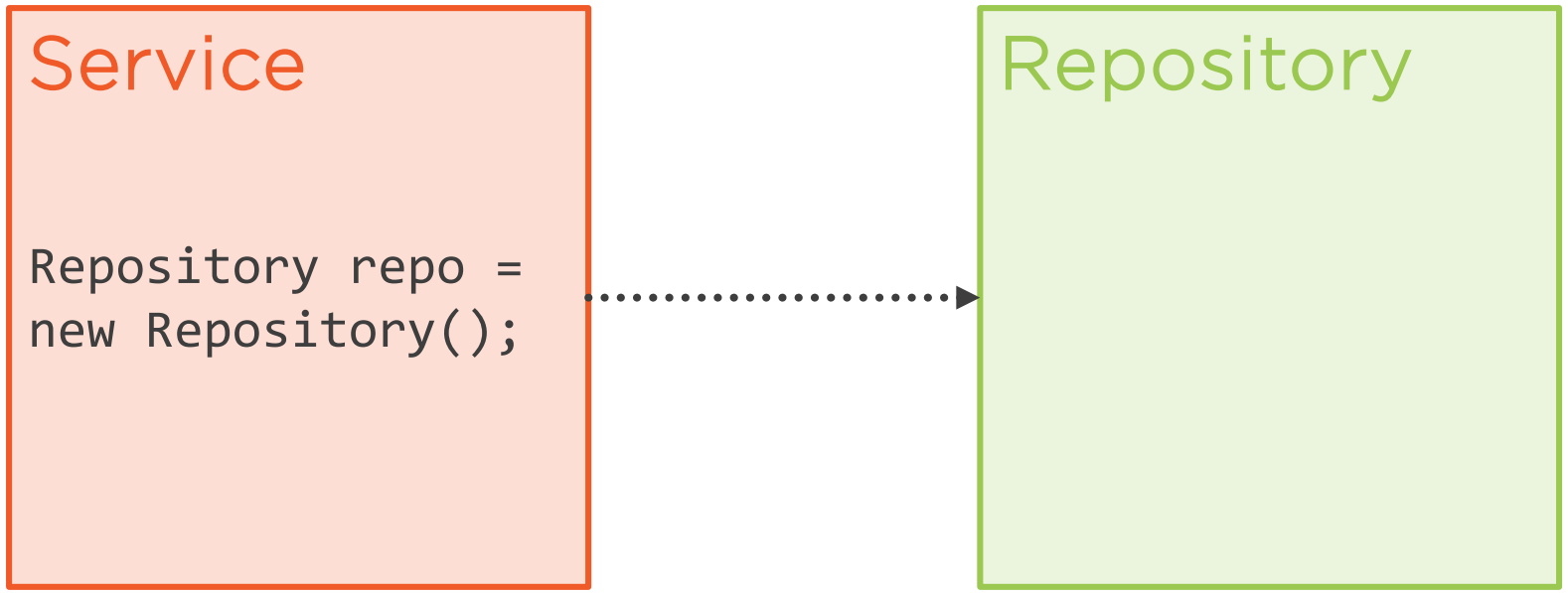


Model classes

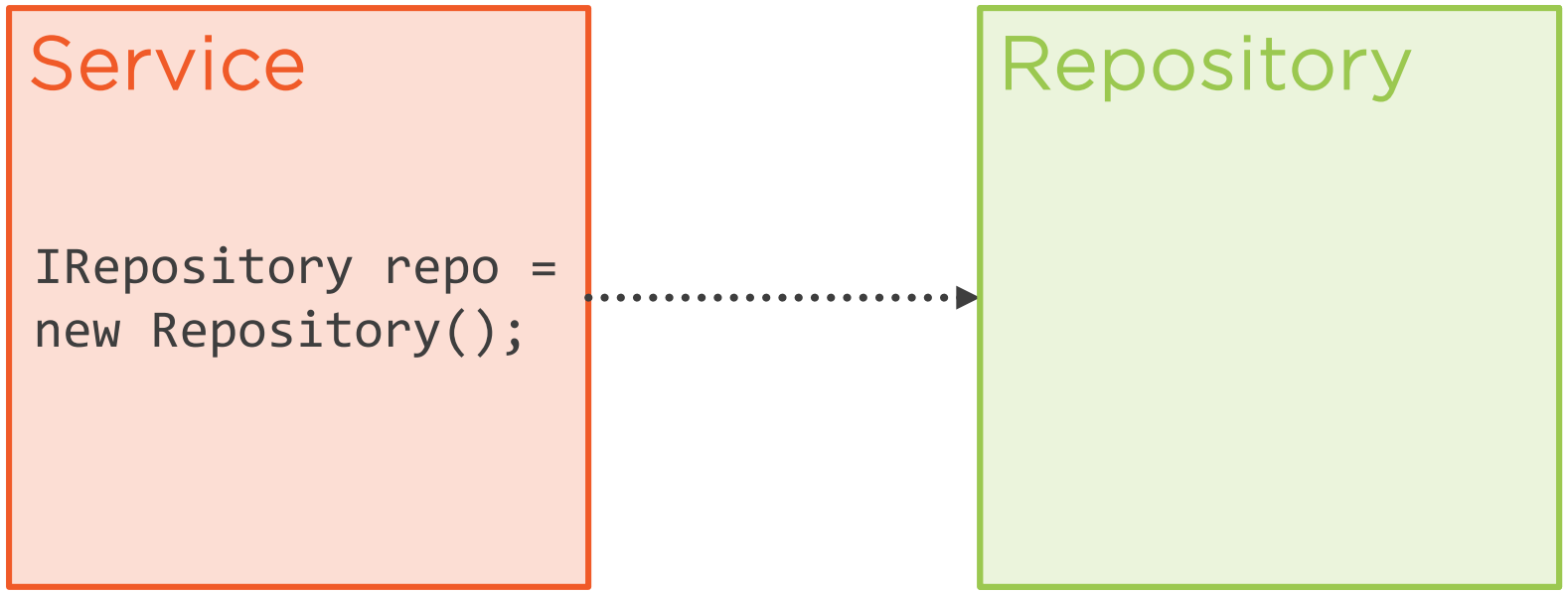
Repositories

Data services

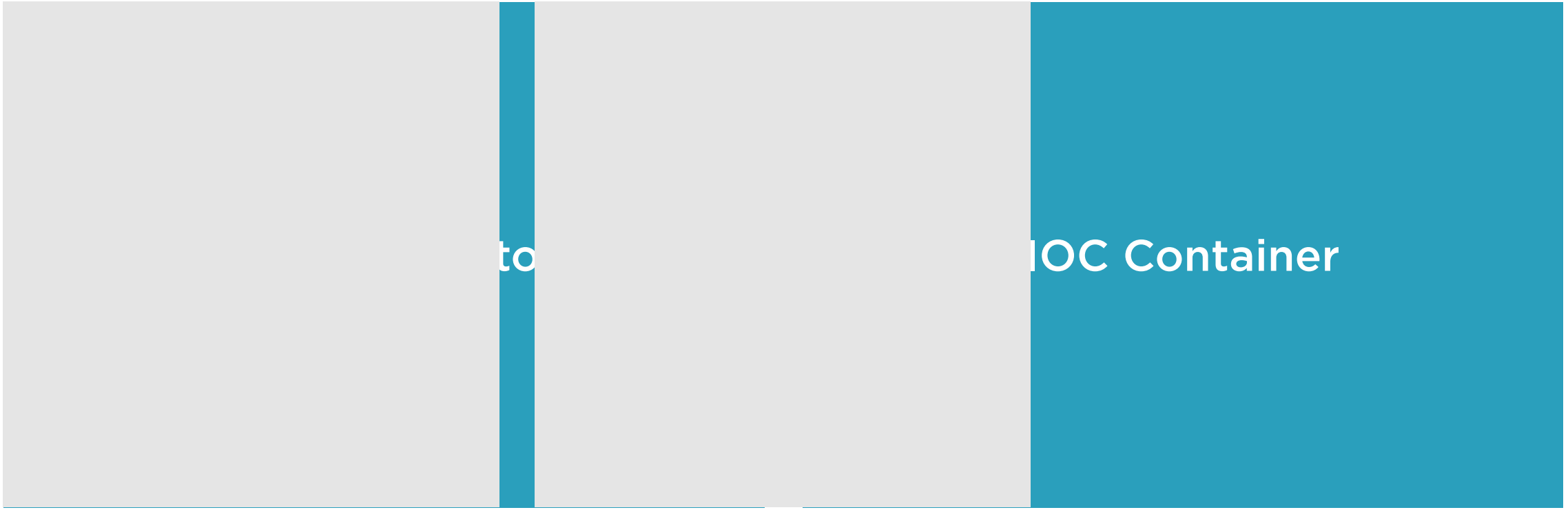
The Problem



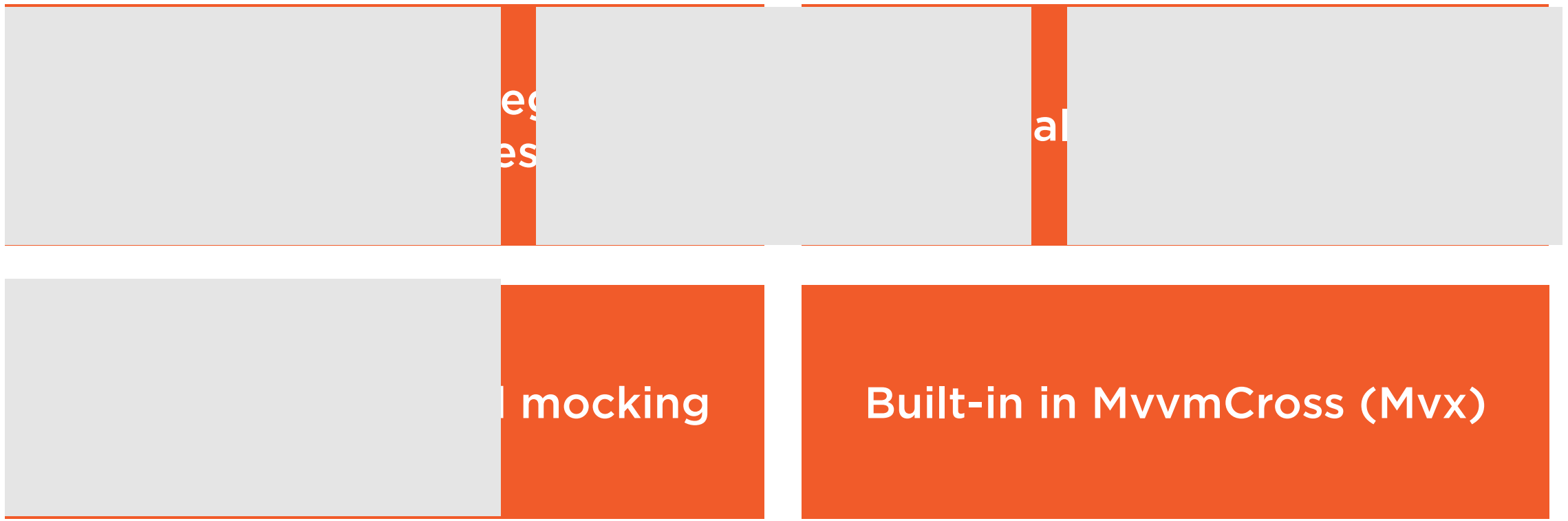
Abstraction



Possible Solutions



Service Location & IOC



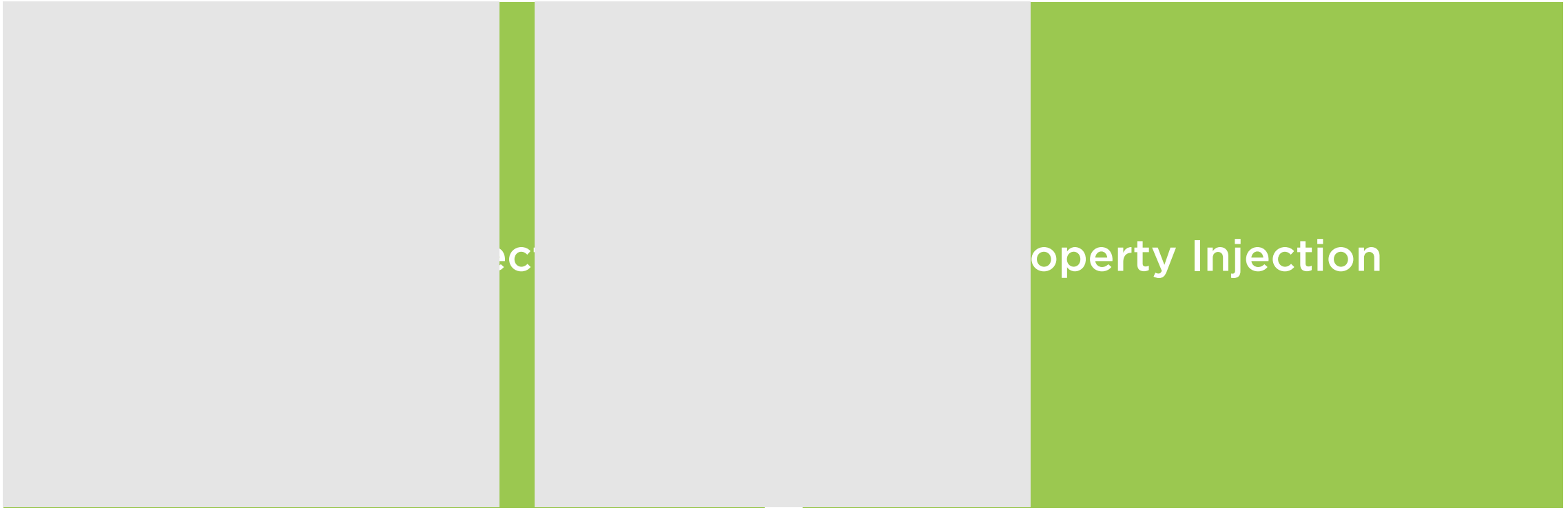
```
Mvx.RegisterSingleton<ICityDataService>
    (new CityDataService());


var cityDataService = Mvx.Resolve<ICityDataService>();
```

Mvx Service Locator Registration

```
CreatableTypes()
    .EndingWith("Service")
    .AsInterfaces()
    .RegisterAsLazySingleton();
```

Bulk Registration

Dependency Injection

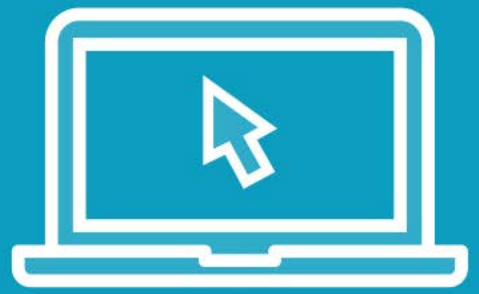


Constructor Injection

```
public class UserDataService: IUserDataService
{
    private readonly IUserRepository _userRepository;

    public UserDataService(IUserRepository userRepository)
    {
        _userRepository = userRepository;
    }
}
```

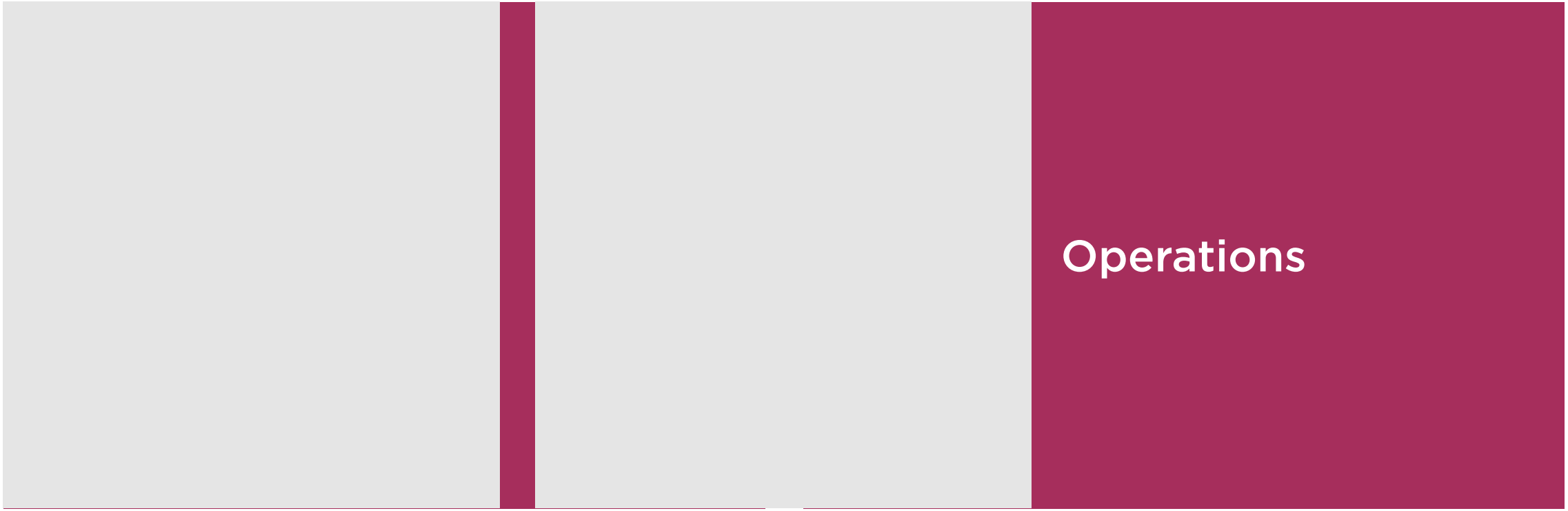
IOC can be used to plug a different implementation in the core per platform.

Demo



Mvx Service Locator & IOC

View Models



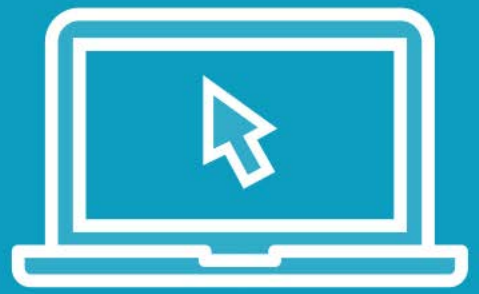
View Model

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

View Model

```
public class JourneyDetailViewModel :
  MvxViewModel, IJourneyDetailViewModel
{
    public MvxCommand CloseCommand { get; set; }

    public JourneyDetailViewModel()
    {
        CloseCommand = new MvxCommand(() =>
        {
            Close(this);
        });
    }

}
```

Demo



Working with view models

```
NavigateToSearchJourneyCommand =
    new MvxCommand(() =>
        ShowViewModel<SearchJourneyViewModel>());
```

Basic View Model Navigation

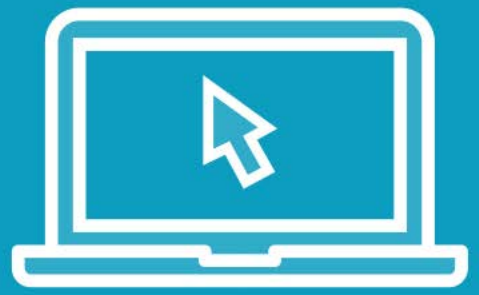
```
ShowJourneyDetailsCommand = new
MvxCommand<Journey>(selectedJourney =>
{
    ShowViewModel<JourneyDetailViewModel>(
        new { journeyId = selectedJourney.JourneyId});

});
```

Passing Parameters

```
public void Init(int journeyId)
{
    _journeyId = journeyId;
}
```

Receiving Parameters

Demo



Navigation between view models

Summary

PCL used for the Core shared code

IOC is heavily used

Code up until View Model layer is shared