Goals

The primary goal will be to create a simple app in Xamarin Forms that displays a list. The user of the app can add an item to that list by typing in a word in the entry bar and pressing the + button. The app can be used as a small todo list.

Required assets

We are going to use the "xamarin-workshop" Git repository (https://github.com/Xablu/xamarin-workshop). This repository already provides a starter as well as a completed solution for the "ToDoApp". Please make sure you have downloaded or cloned this repository before you begin.

Challenge

You'll start with an existing application that has an empty page. Currently it's showing a label. You can already run this app and see what it does. You are going to create a ToDo app using the MVVM pattern. The project already has a Utillities folder containing a NotifiyPropertyChanged class. This class will be used to notify the view when data has changed. When adding new code, make sure to also add the imports for the needed objects.

1. Inspect and run the starter solution in the folder on an Android or iOS device/emulator, you will see that it's not very visually appealing.
2. Create a Models folder and add a TodoItem class.
3. Create a ViewModels folder and add a TodoViewModel class.
4. Update the TodoAppPage.xaml so it displays an Entry, a Button and a Listview.
5. Add the Completed event to the TodoAppPage.xaml.cs.
6. Set the ViewModel as BindingContext inside the TodoAppPage.xaml.cs.
7. Run the app and try to add a ToDo item.

Steps

Below are the step-by-step instructions to implement the exercise.

Inspect the starter solution

Open the starter solution in the folder and run the application. It works but it's not very visually appealing.

Create a Models folder and a TodoItem class

We need an object that can be used as our model. This object will have a Title property that will be displayed in the list later.

1. Add a new folder to the **ToDoApp** project and name it **Models**.
2. Add a new class to that folder and name it **TodoItem**
3. Add two properties to the class called **IsDone** (bool) and **Title** (string).

```
1. public class TodoItem
2. {
3.     public bool IsDone { get; set; }
4.     public string Title { get; set; }
5.
6.     public override string ToString()
7.     {
8.         return Title;
9.     }
10.}
```

Create a ViewModels folder and a ViewModel class

To handle the logic between the **Model** and the **View** we'll need a **ViewModel**. This class will add a newly added item to the list and notify the view that the data has changed.

1. Add a new folder to the **ToDoApp** project and name it **ViewModels**.
2. Add a new class to that folder and name it **TodoViewModel**. Enherit the class from **NotifyPropertyChanged**.
3. Add a list and a command. The list will hold the added items and the command is to add the filled in item. The list is of type **ObservableCollection**, this is because it will notify the view automatically when the list changes.

```
1. public class TodoViewModel : NotifyPropertyChanged
2. {
3.     private readonly ObservableCollection<TodoItem> _todos = new Observ
   ableCollection<TodoItem>();
4.     private string _newTodoItemTitle;
5.
6.     public TodoViewModel()
7.     {
8.         NewTodoItemCommand = new Command<string>(HandleNewTodoItemComma
   nd);
9.     }
10.
11.    public ICommand NewTodoItemCommand { get; }
12.    public ObservableCollection<TodoItem> Todos => _todos;
```

```
13.     public string NewTodoItemTitle
14.     {
15.         get => _newTodoItemTitle;
16.         set => SetProperty(ref _newTodoItemTitle, value);
17.     }
18.
19.     private void HandleNewTodoItemCommand(string title)
20.     {
21.         var newTodoItem = new TodoItem { Title = NewTodoItemTitle };
22.         _todos.Add(newTodoItem);
23.
24.         NewTodoItemTitle = null;
25.     }
26. }
```

Update the TodoAppPage.xaml so it displays components

We need to add components to the page so the user can add a new ToDo item.

1. Open the **TodoAppPage.xaml** file and delete the **Label**.
2. Add padding to the page according to the platform that you use. This is because for example in iOS the app starts on the left top corner of the screen. This means that without the padding the top bar will overlap your app.

```
1. <ContentPage.Padding>
2.     <OnPlatform
3.         x:TypeArguments="Thickness"
4.         Android="10, 0, 10, 0"
5.         iOS="10, 40, 10, 0"
6.         WinPhone="30,20,30,20" />
7. </ContentPage.Padding>
```

3. At an **Entry** and a **Button** so it is displayed at the top of the page.
4. The **Entry** should have a **Completed** handler and **Text** property that binds to the **TodoViewModel**.
5. The **Button** should have a **Command** that binds to the **TodoViewModel**.
6. At a **ListView** under the **Entry** and **Button**.
7. The **ListView** should have an **ItemsSource** property that binds to the list that is inside the **TodoViewModel**.
8. The **ListView** should have an **ItemTemplate** containing a **Label** that binds to the **Title** of the **TodoItem**.

Completed **ToDoAppPage.xaml** on the next page

```xaml
<ContentPage
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:local="clr-namespace:TodoApp"
    x:Class="TodoApp.TodoAppPage">

    <ContentPage.Padding>
        <OnPlatform
            x:TypeArguments="Thickness"
            Android="10, 0, 10, 0"
            iOS="10, 40, 10, 0"
            WinPhone="30,20,30,20" />
    </ContentPage.Padding>

    <StackLayout
        Orientation="Vertical">

        <StackLayout
            Orientation="Horizontal">

            <Entry
                HorizontalOptions="FillAndExpand"
                Text="{Binding NewTodoItemTitle}"
                Completed="Handle_Completed"
                Placeholder="Add a todo..." />

            <Button
                Text="+"
                HorizontalOptions="End"
                Command="{Binding NewTodoItemCommand}" />

        </StackLayout>

        <ListView
            ItemsSource="{Binding Todos}">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <ViewCell>
                        <StackLayout
                            Orientation="Horizontal">

                            <Label Text="{Binding Title}"
                                HorizontalOptions="StartAndExpand" />

                        </StackLayout>
                    </ViewCell>
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>

    </StackLayout>
</ContentPage>
```

Add the Completed event to the TodoAppPage.xaml.cs

Some platforms use a keyboard where the user can click on done or continue. By handling this event we want to add a new **TodoItem** to the list.

1. Open the **TodoAppPage.xaml.cs** file.
2. Add a method that is called **Handle_Completed** to the class that handles when the user completed entering text to the **Entry**.

```
void Handle_Completed(object sender, System.EventArgs e)
{
    if (_todoViewModel.NewTodoItemCommand.CanExecute(null))
        _todoViewModel.NewTodoItemCommand.Execute(null);
}
```

Set the ViewModel as BindingContext

We need to let the view know what to use as **BindingContext**. BindingContext will be the ViewModel. The view will get notified through the **ViewModel** when data has changed.

1. Open the **TodoAppPage.xaml.cs** file.
2. Add a new object of **TodoViewModel** to the BindingContext inside the constructor.

Total **TodoAppPage.xaml.cs** file

```
public partial class TodoAppPage : ContentPage
{
    private TodoViewModel _todoViewModel;

    public TodoAppPage()
    {
        InitializeComponent();

        _todoViewModel = new TodoViewModel();
        BindingContext = _todoViewModel;
    }

    void Handle_Completed(object sender, System.EventArgs e)
    {
        if (_todoViewModel.NewTodoItemCommand.CanExecute(null))
            _todoViewModel.NewTodoItemCommand.Execute(null);
    }
}
```

Summary

Congratulations! You have successfully created and modified app that displays a list of Todo items. You can also add new items to the list using the created entry. Feel free to add extra functionality to the app.