**The laboratory work 8**

**Trip application**

## Creating the initial app

To start off the new TripLog mobile app project, we will need to create the initial solution architecture. We can also create the core shell of our app’s user interface by creating the initial screens based on the basic features we have just defined.

### Setting up the solution

We will start things off by creating a brand new, blank Xamarin.Forms solution within Visual Studio by performing the following steps:

1. In Visual Studio, click on File | New Solution. This will bring up a series of dialog screens that will walk you through creating a new Xamarin.Forms solution. On the first dialog, click on App on the left-hand side, under the Multiplatform section, and then select Blank Forms App, as shown in the following screenshot:



1. On the next dialog screen, enter the name of the app, TripLog, ensure that Use Portable Class Library is selected for the Shared Code option, and that Use XAML for user interface files option is checked, as shown in the following screenshot:



The Xamarin of a Portable Class Library for its core library project. As of the writing of this book, the Visual Studio for Mac templates still use a Portable Class Library.

1. On the final dialog screen, simply click on the Create button, as follows:



After creating the new Xamarin.Forms solution, you will have several projects created within it, as shown in the following screenshot:



There will be a single portable class library project and two platform-specific projects, as follows:

* **TripLog**: This is a portable class library project that will serve as the *core* layer of the solution architecture. This is the layer that will include all our business logic, data objects, Xamarin.Forms pages, and other non-platform-specific code. The code in this project is common and not specific to a platform, and can therefore, be shared across the platform projects.
* **TripLog.** [**iOS**](https://subscribe.packtpub.com/learn-ios/): This is the [iOS](https://subscribe.packtpub.com/learn-ios/) platform-specific project containing all the code and assets required to build and deploy the [iOS](https://subscription.packtpub.com/tech/ios) app from this solution. By default, it will have a reference to the TripLog core project.
* **TripLog.Droid**: This is the [Android](https://subscription.packtpub.com/tech/android) platform-specific project containing all the code and assets required to build and deploy the [Android](https://subscription.packtpub.com/tech/android) app from this solution. By default, it will have a reference to the TripLog core project.

If you are using Visual Studio for Mac, you will only get an [iOS](https://subscription.packtpub.com/tech/ios) and an Android project when you create a new Xamarin.Forms solution.  To include a Windows (UWP) app in your Xamarin.Forms solution, you will need to use Visual Studio for Windows.  Although the screenshots and samples used throughout this book are demonstrated using Visual Studio for Mac, the code and concepts will also work in Visual Studio for Windows. Refer to the Preface of this book for further details on software and hardware requirements that need to be met to follow along with the concepts in this book.

You’ll notice a file in the core library named App.xaml, which includes a code-behind class in App.xaml.cs named App that inherits from Xamarin.Forms.Application. Initially, the App constructor sets the MainPage property to a new instance of a ContentPage named TripLogPage that simply displays some default text.

#### Updating the Xamarin.Forms packages

If you expand the Packages folder within each of the projects in the solution, you will see that Xamarin.Forms is a NuGet package that is automatically included when we select the Xamarin.Forms project template. It is possible that the included NuGet packages need to be updated. Ensure that you update them in each of the projects within the solution so that you are using the latest version of Xamarin.Forms.

**Creating the main page**

The main page of the app will serve as the entry point into the app and will display a list of existing trip log entries. Our trip log entries will be represented by a data model named TripLogEntry. Models are a key pillar in the Model-View-ViewModel (MVVM) pattern and data binding, which we will explore more in our tutorial, ***How to add MVVM pattern and data binding to our Travel app***. For now, we will create a simple class that will represent the TripLogEntry model.

Let us now start creating the main page by performing the following steps:

1. First, add a new Xamarin.Forms XAML  ContentPage to the core project and name it MainPage.
2. Next, update the MainPage property of the App class in App.xaml.cs to a new instance of Xamarin.Forms.NavigationPage whose root is a new instance of TripLog.MainPage that we just created:

|  |
| --- |
| public App() { InitializeComponent(); MainPage = **new NavigationPage(new MainPage());** } |

1. Delete TripLogPage.xaml from the core project as it is no longer needed.
2. Create a new folder in the core project named Models.
3. Create a new empty class file in the Models folder named TripLogEntry.
4. Update the TripLogEntry class with auto-implemented properties representing the attributes of an entry:

|  |
| --- |
| public class TripLogEntry {  public string Title { get; set; } public double Latitude { get; set; } public double Longitude { get; set; } public DateTime Date { get; set; } public int Rating { get; set; } public string Notes { get; set; } } |

Now that we have a model to represent our trip log entries, we can use it to display some trips on the main page using a ListView control. We will use a DataTemplate to describe how the model data should be displayed in each of the rows in the ListView using the following XAML in the ContentPage.Content tag in MainPage.xaml:

In the main page’s code-behind, MainPage.xaml.cs, we will populate the ListView ItemsSource with a hard-coded collection of TripLogEntry objects

|  |
| --- |
| public partial class MainPage : ContentPage { public MainPage() { InitializeComponent();var items = new List{new TripLogEntry{Title = "Washington Monument",Notes = "Amazing!",Rating = 3,Date = new DateTime(2017, 2, 5),Latitude = 38.8895,Longitude = -77.0352},new TripLogEntry{Title = "Statue of Liberty",Notes = "Inspiring!",Rating = 4,Date = new DateTime(2017, 4, 13),Latitude = 40.6892,Longitude = -74.0444},new TripLogEntry{Title = "Golden Gate Bridge",Notes = "Foggy, but beautiful.",Rating = 5,Date = new DateTime(2017, 4, 26),Latitude = 37.8268,Longitude = -122.4798} };trips.ItemsSource = items;}} |

At this point, we have a single page that is displayed as the app’s main page. If we debug the app and run it in a simulator, emulator, or on a physical device, we should see the main page showing the list of log entries we hard-coded into the view, as shown in the following screenshot.

