Chapter 2 Local settings: Preferences and Secure Storage

Storing user preferences and app settings locally is a very common requirement with mobile applications. There can be tons of examples, which might include, and that are not limited to, storing the last access date/time, the consent to use a cellular network when Wi-Fi is not available, color scheme preferences and so on. These options do not generally require to be secured, but sometimes you might need to store an encrypted password or an information that identifies the user. In this case, you do need to secure preferences and settings. The Xamarin.Essentials library provides types that make it extremely easy to store local settings in both ways. This chapter describes both scenarios and provides details on the classes, and their methods, you can use for these purposes.

Managing local preferences

Local preferences in Xamarin.Forms can be easily managed via the **`Preferences`** class from the **`Xamarin.Essentials`** namespace. The **`Preferences`** class stores preferences into the native local storages, more specifically:

- For Android, into the Shared Preferences.
- For iOS, into the NSUserDefault.
- For UWP, into the application data container.

On each platform, preferences are key/value pairs. The value can be of one of the primitive .NET types. Before understanding how it works, it is important to list the methods exposed by the **`Preferences`** class, summarized in Table 3.

Table 3: Methods exposed by the Preferences class

Methods exposed by the Preferences class	
`Set`	Adds a preference to the local settings.
`Get`	Gets a preference value from the local settings
`Remove`	Remove the specified preference
`Clear`	Deletes all the preferences for the current app
`ContainsKey`	Detects if the specified preference exists in the storage given its key

Tip: Uninstalling the app will also remove all local preferences.

Let's now start with an example. Suppose you want to store the last date/time the app has been used. You can do this in the **`OnSleep`** method of the App.xaml.cs file as follows:

```
protected override void OnSleep()

{

    Preferences.Set("TimeOfLastUsage", DateTime.Now);

}
```

`TimeOfLastUsage` is the key, and the value is a **`DateTime`** object. Keys are always of type **`string`**. Retrieving a value from the local settings is still easy. In the **`OnStart`** method, you can write something like the following:

```
internal static DateTime timeOfLastUsage;

protected override void OnStart()

{

    timeOfLastUsage = Preferences.Get("TimeOfLastUsage",

        DateTime.Now);

}
```

The generic **`Get`** method returns the value for the specified key and stores the result into a variable. **`Get`** also requires you to specify a default value in case the key does not exist in the storage, and this is true for all the supported data types. In this example, local preferences are used to store an information that can be useful for the app logic at runtime, so it's not really a user preference. Let's make another example at this purpose, checking if the phone is connected to a cellular network only, asking the user permission for using it and storing the result. Code Listing 8 contains the code that you want to add to the MainPage.xaml.cs file.

Code Listing 8

```
        private bool CheckCellularConnection()
        {
            var profiles = Connectivity.ConnectionProfiles;
            return profiles.Count() == 1 &&
                   profiles.Contains(ConnectionProfile.Cellular);
        }
        private bool useCellularNetwork;
        protected override async void OnAppearing()
        {
            if(CheckCellularConnection())
            {
                if (!Preferences.ContainsKey("UseCellularNetwork"))
                {
                    bool result = await DisplayAlert("Warning",
                        "Do you agree on using cellular data when Wi-
Fi is not available?",
                        "Yes", "No");
                    Preferences.Set("UseCellularNetwork", result);
                    useCellularNetwork = result;
                }
                else
                    useCellularNetwork =
                         Preferences.Get("UseCellularNetwork", false);
            }
        }
```

The **`CheckCellularConnection`** method checks if only one connection is available and if this is a cellular network. The remaining code is in the **`OnAppearing`** method for convenience so that the code is executed at the app startup, but you will likely create a separate method that will be invoked when the **`ConnectivityChanged`** event of the **`Connectivity`** class is raised. In this second example, the code invokes the **`Preferences.ContainsKey`** method to detect if the preference was already saved, then it still invokes the **`Set`** and **`Get`** method to store and retrieve the preference value respectively. The **`Preferences`** class is intended to store small pieces of information represented by primitive .NET types. If you need to locally save and retrieve more complex and structured data, your choice will be a SQLite database, as discussed in Chapter 4.