Lab 01: Mobile Data using ADO.Net

Prerequisites

You will need a development environment, either a Mac or Windows PC with the Android SDK and Xamarin tools installed. We will be using the Android emulator to test the code we are building, so make sure to have a virtual device already configured and ready to run. See the **Xamarin.Android** setup documentation if you need help getting your environment setup:

http://docs.xamarin.com/guides/android/getting_started/installation/

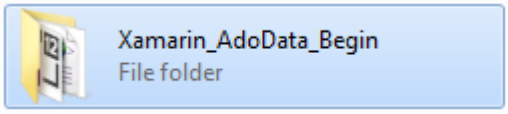
Lab Goals

The goal of this lab will be to introduce integrating local data storage on mobile devices using ADO.Net.

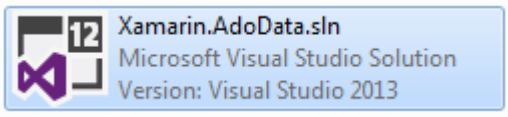
Steps

Open the Starting Solution

1. Launch **Xamarin Studio**
2. Click **Open...** on the Xamarin Studio Welcome and navigate to the **Lab 01 Resources** folder included with this document.
3. Locate the **Xamarin_AdoData_Begin** folder – make sure it's the **begin** and not the **completed** folder.



4. Inside the **Xamarin_Data_Begin** folder, you will find a **Xamarin.Data.sln** file – double click on this file to open the starter solution



5. Go ahead and build and run the application in the emulator to make sure it compiles and your environment is ready. Let the instructor know if you have any trouble.

Setting up ADO.Net for Mobile Data

1. Right-Click on the **Xamarin.AdoData.Droid** project and select **Edit References...**

2. Add **Mono.Data.Sqlite** and **System.Data** as references

3. Open the Ado\StockDatabase.cs file
4. Locate the comment `TODO: Step 1 - Setup database path`

```
public static string DatabaseFilePath
{
    get
    {
        var sqliteFilename = "StockDB.db3";

        // Just use whatever directory SpecialFolder.Personal returns
        string libraryPath = Environment.GetFolderPath(Environment.Spe
cialFolder.Personal);

        var path = Path.Combine(libraryPath, sqliteFilename);

        return path;
    }
}
```

5. Locate the comment `//TODO: Step 2 - Create Database if it does not currently exist`

```
public Boolean CreateDatabaseIfNotExists()
{
    if (File.Exists(DatabaseFilePath))
        return true;

    var creationSuccessful = false;

    lock (locker)
    {

        // Need to create the database and seed it with some data.
        SqliteConnection.CreateFile(DatabaseFilePath);
```

```
        using (var connection = new SqliteConnection("Data Source=" +
DatabaseFilePath))
        {
            connection.Open();
            using (var c = connection.CreateCommand())
            {
                c.CommandText = "CREATE TABLE [Stock] (_id ntext, Symb
ol ntext);";
                creationSuccessful = c.ExecuteNonQuery() > 0;
            }
        }
    }

    return creationSuccessful;
}
```

Tip: If your application has multiple threads accessing your database, make sure to lock access or restrict multiple connections

6. Locate the comment `TODO: Step 3 - Perform an insert`

```
connection.Open();
using (var c = connection.CreateCommand())
{
    c.CommandText = String.Format("INSERT INTO [Stock] ([_id], [Symbol
]) VALUES ('{0}', '{1}')", stock.Id, stock.Symbol);
    insertSuccessful = c.ExecuteNonQuery() > 0;
}
```

7. Locate the comment `TODO: Step 4 - Query database and map to model objects`

```
using (var contents = connection.CreateCommand())
{
    contents.CommandText = "SELECT [_id], [Symbol] from [Stock]";
    var r = contents.ExecuteReader();
    while (r.Read())
        stockInDatabase.Add(
            new Model.Stock()
            {
                Id = Convert.ToInt32(r["_id"]),
                Symbol = r["Symbol"].ToString()
```

```
            });
}
```

Tip: Try to create your database methods so that they receive and return model objects or primitive types. This way, your UI is further separated from any database implementation.

8. Locate the comment `TODO: Step 5 - Integrate the ADO.Net client into your UI`

```
var adoDatabase = new Ado.StockDatabase();


await Task.Run(() =>
{


    adoDatabase.CreateDatabaseIfNotExists();


    for (int i = 0; i < 5; i++)
    {
        adoDatabase.InsertStock(Ado.StockDatabase.GenerateStock()/* Ge
nerates a fake stock */);
    }
});


var selectAllStock = adoDatabase.SelectStock();
this.ListAdapter = new ArrayAdapter<Model.Stock>(this, Android.Resourc
e.Layout.SimpleListItem1, selectAllStock);
```

Summary

In this lab, we learned how to integrate the ADO.Net components into our Mobile applications and perform simple SQL queries to create a database, insert and retrieve data.