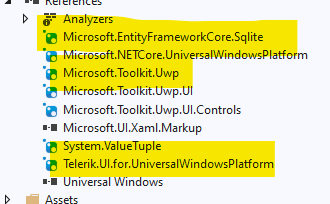
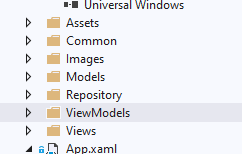
Lottery Example – add a new customer to database, this goes on premise of email as ID, no logon at this point.

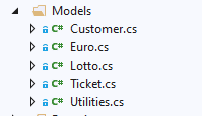
Add the Nuget packages required:



In the Lottery App rename the Pages folder to Views and create the following folders, Models, Repository, ViewModels:



Create new class files for your Business Models and add the code:



In the Repository folder add a file named CustomerContext.cs

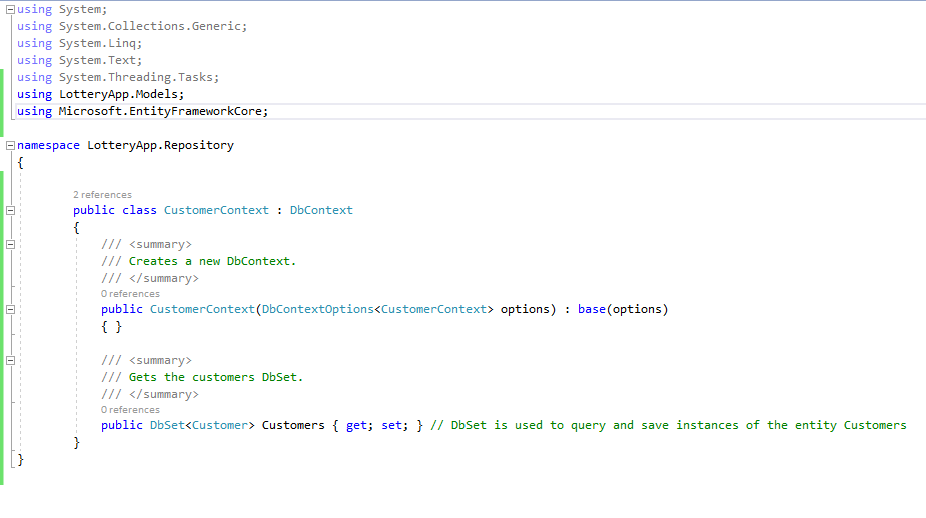


Code CustomerContext.cs as follows:

DBContext:

DbContext instance represents a session with the database and can be used to query and save instances of your entities. DbContext is a combination of the unit of work and repository patterns.

You would typically create a class that derives from DbContext and contains DbSet<TEntity> properties for each entity in the model. If the DbSet<TEntity> have a public setter, the are automatically initialized when an instance of the derived context is created.



using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using LotteryApp.Models;  
using Microsoft.EntityFrameworkCore;

namespace LotteryApp.Repository  
{  
 public class CustomerContext : DbContext  
 {  
 /// <summary>  
 /// Creates a new DbContext.  
 /// </summary>  
  
 public CustomerContext(DbContextOptions<CustomerContext> options) : base(options)  
 { }

/// <summary>  
 /// Gets the Customers DbSet based on the Customer model.  
 /// </summary>

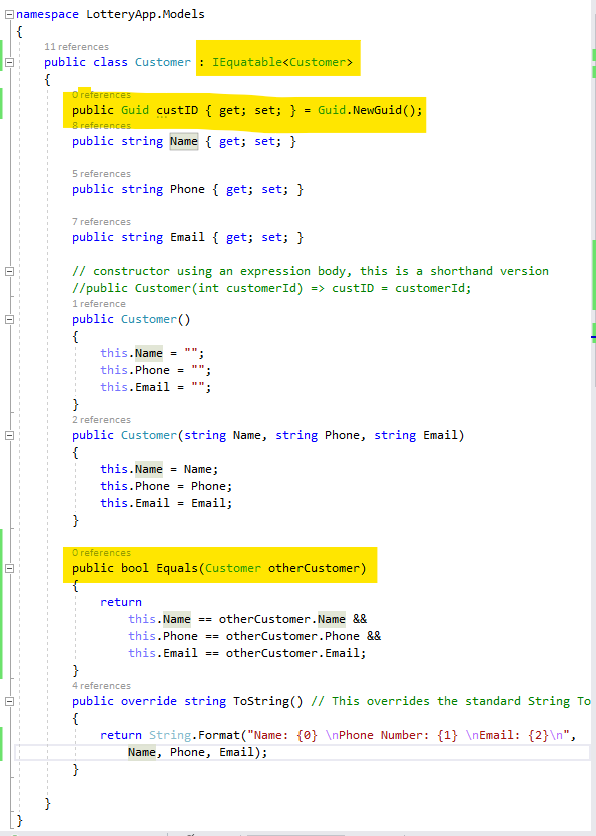
public DbSet<Customer> Customers { get; set; } // DbSet is used to query and save instances of the entity Customers  
 }  
}

Customer.cs

The Customer.cs Model is updated as follows to allow for it to work with the DbSet Customers and the entity in the database Customer.

The IEquatable<Customer> is inherited and the Equals method coded. This automatically checks to see if there is a customer with the same details and if it does it returns the Customer which is already there rather than creating a new instance of the customer.

CustID has been added to match the database Entity, Guid will automatically create an ID in sequence.



App Start

The starting point for the App is App.xaml.cs:

The repository object is set up on launch of the application. The repository creates the Dbsets for the customers to be manipulated. After this the App will launch the first page of the application which is CustomerListPage.xaml.

App.xaml.cs

This creates an instance of the ITutorialRepository model, coded in the model folder, this will be available throughout the app. This is an instance of all of the database entities to be manipulated.

It then runs the method SqliteDatabase() within App.xaml.cs , see next image