

Workshop 4: Introduction to Databases and Domain Entities [25]

Objectives: to work with the Database layer and extend the application to include a Local DB

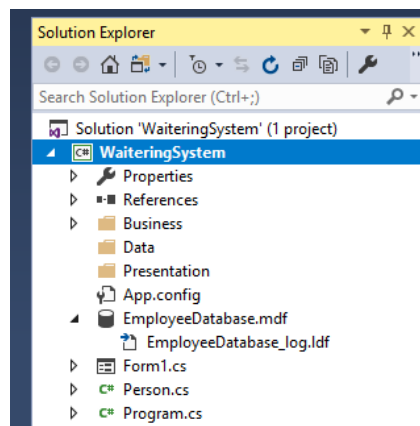
Instructions:

- Download the *startingFile* from vula and answer the questions that follow.
- Upload your file back to vula once completed

Section A: Creating a LocalDB Database

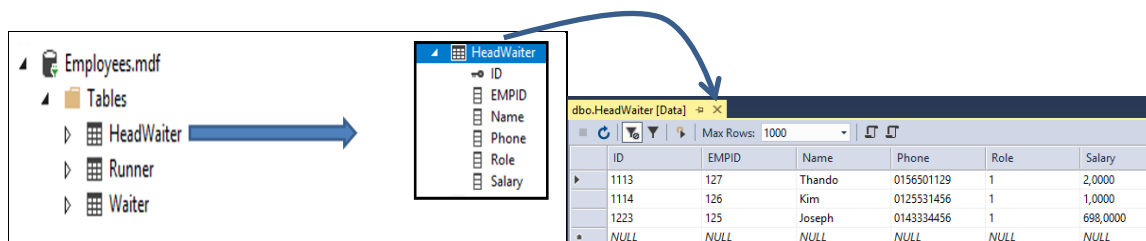
We will now create a database file and include it in your project. When you create a service-based database in Visual Studio, the SQL Server Express LocalDB engine is used to access a SQL Server database file (.mdf).

1. On the menu bar, choose Project, Add New Item. In the list of templates, scroll down until *Service-based Database* appears, and then choose it. Name the database ***EmployeeDatabase*** and then choose the Add button. You will now have a database file in your solution – see below



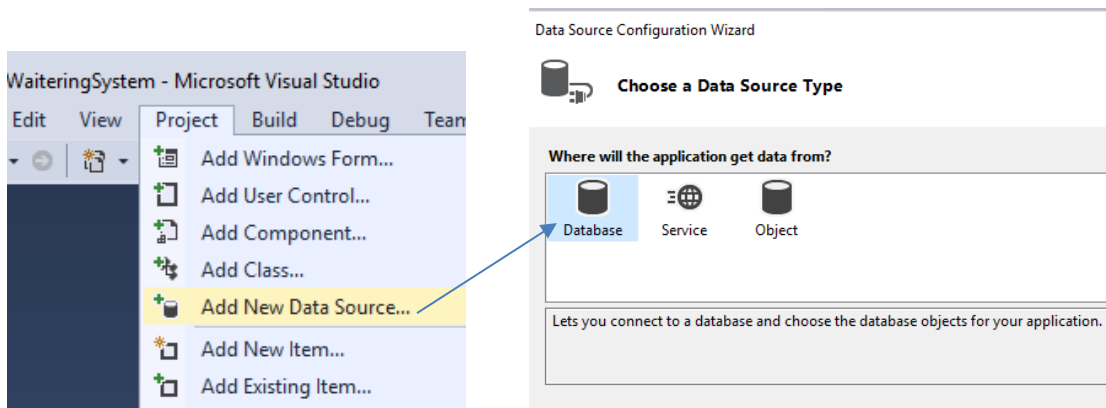
The properties window for the database shows the connection string and the location of the primary .mdf file for the database. To display this window, choose the Server Explorer tab, expand the Data Connections node, open the shortcut menu for the database, and then choose Properties.

2. Now we will need to create the tables (all of them) as shown in the figures below:

[illegible][illegible]

3. Add a datasource to the project as follows:

3.1 Select *Project* from the menu and select *Add New Data Source*



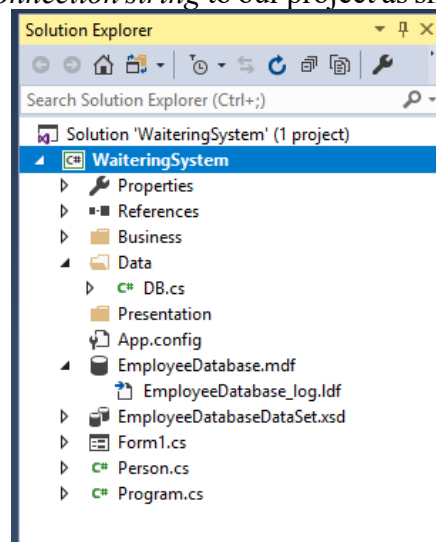
3.2 Select *Database* in the *Data Source Configuration Wizard* dialog, and click *Next*.

3.3 Then, Select *Dataset* as the type of database model you want.

3.4 Click on *New connection* to create the database connection. You will have to browse and locate your database; and save your connection string.

3.5 In the next dialog, choose the *Table* as your database objects, and click *Finish*.

Now we have a *dataset* and a *connection string* to our project as shown below:



SECTION B: ENTITY CLASSES

4. In the Database Layer folder: add a database class to your solution, call it **DB** (we will use this as the base class)

4.1 Import the packages/namespaces that will allow you to manipulate data

4.2 Import the application namespace associated with the properties – this will allow you to have access to the connection string

4.3 In the class header:

- Define a variable *strConn* of type string. This variable should store the string connection below: *Settings.Default.EmployeeDatabaseConnectionString*
- Declare a variable *cnMain* of type *SqlConnection*.
- Declare a variable *dsMain* of type *DataSet*
- Declare a variable *daMain* of type *SqlDataAdapter*

4.4 Add a constructor. Within the constructor of the class, write the code to open a connection and create a new dataset object. Use try catch statements to catch any system exceptions. Return a message to the user of any exception reported.

5. Within your DB class: Create a method *FillDataSet* whose task is to fill the dataset fresh from the database for a specific table and with a specific Query. This method does not return any value to any of its calling procedures. The activities involved in this method include
 - 5.1 The method receives two parameters: the SQL statement *aSQLstring* and the Table *aTable* from which the query will emanate from. Within this method:
 - 5.2 Open a try catch block. Within this block:
 - a) Create an instance of the data adapter (*daMain*). The adapter will need two parameters: the SQL statement *aSQLstring* and the connections (*cnMain*).
 - b) Open the connection
 - c) Clear the dataset
 - d) Fill the data adapter (*daMain*) using the dataset (*dsMain*) and the specific Table
 - e) Close the connection
 - 5.3 Catch any exceptions resulting from the run of the method as follows (implying you should have had incorporated the try catch statement):

```

catch (Exception errObj)
{
    MessageBox.Show(errObj.Message + " " + errObj.StackTrace);
}

```

6. In the DB class: Include a method *UpdateDataSource* which will update the datasource. This method will be used by any of the classes inheriting from this class
 - 6.1 The method has two parameters SQL statement *sqlLocal* to be used to do the update and the table *table* which needs to be updated.
 - 6.2 Declare a Boolean variable *success* that will signal the successful update
 - 6.3 Open a try catch statement block, and within this block:
 - a) Open the connection
 - b) Update the database table via the data adapter. i.e the data adapter, uses the Update Method. You will need to call the Update method which takes two parameters: the dataset *dsMain* and the table *table*
 - c) Close the connection
 - d) Fill the dataset with the SQL statement *sqlLocal* and the specific table *table*.
 - e) The variable success returns true (Assign *true* to the variable success).
 - f) Catch any exceptions resulting from the run of the method as follows (implying you should have had incorporated the try catch statement):

```

catch (Exception errObj)
{
    MessageBox.Show(errObj.Message + " " + errObj.StackTrace);
    success = false;
}
finally
{
}
return success;
}

```

Debug your program and ensure there are no errors.

-----End of the Workshop-----