In-Class Assessment 1 - OMST - WEIGHT 10 %

> LINQ Query and EntityFrameWork CRUD - **Read the entire instructions before beginning your work**

Database

The physical database can be installed from either the `.bak` file or the `.bakpac` file included in the starter kit.

About OMST

> **Off Main-Street Theatre** is a privately owned theatre located near Down-Town.

Requirements

In this assessment, you will be demonstrating your understanding of:

- Reverse-engineering entities from a database
- Writing basic CRUD-related methods for working with a database
- Creating a UI for CRUD behaviour
- Performing a simple LINQ query of a database
- Displaying query results in a UI

You have been supplied a starting solution for this assessment. It contains a database and a Visual Studio solution with certain portions of the assessment pre-coded. **The supplied code works and should not be altered.** Only *modify/extend* the portions identified in these assessment instructions.

Use Frequent Commits

Commit your work at the end of each and every **Activity**. Mark(s) may be deducted based on the commit activity in your repository. Ensure you sync your local clone to GitHub before the end of class. The classroom access ends at the end of class. **It is your responsibility that your work is properly submitted. Failure to submit your work may result in deductions up to and including a mark of 0 (zero) for this assessment.**

Setup

Restore the supplied SQL database backup. The database name is **OMST_2018**. The database contains data for testing your solution.

Run the solution in the starter kit. It should run without compiler errors and you should be able to open the incomplete assessment form pages.

Activity 1 - Entities

Reverse engineer the entities for this assessment from the database. Use **OMSTContext** as the model name. Use **OMSTDB** as the connection name. Alter the default messages for the `Location` entity's string properties. All entity classes and the DAL class must be declared as `private class`. Move the generated context class to

the application library project folder "DAL". Change the namespace and add any necessary additional `using` statements to import the namespace for referencing the entities.

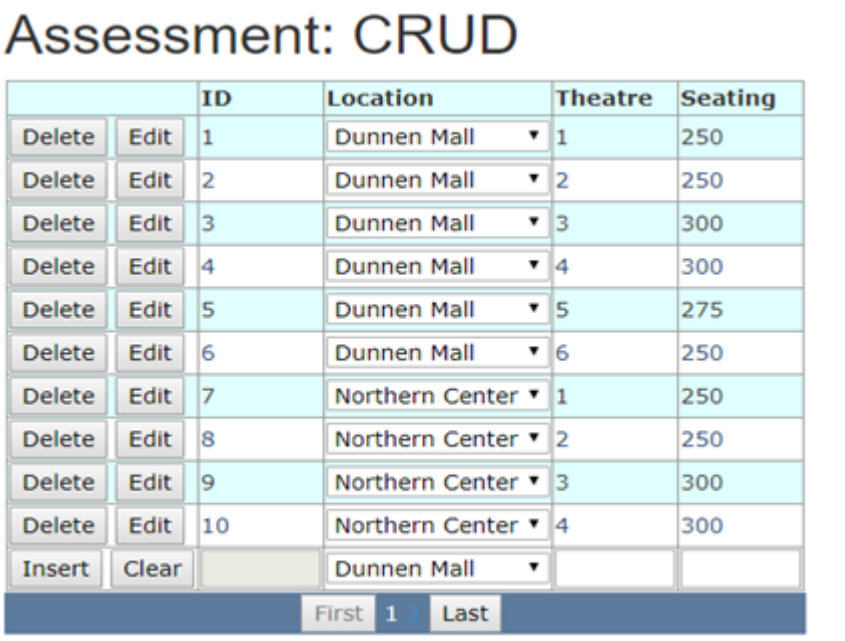
Activity 2 - Complete the `TheatreContoller`, `LocationController` and `LocationInfo` Classes

Complete the code in the **`TheatreController`** class for the following methods: `Theatre_Add`, `Theatre_Update`, `Theatre_Delete`. These methods must use the `TheatreItem` view model for the CRUD behaviour, mapping it accordingly to the `Theatre` entity. The `TheatreItem` view model class has already been coded for you. Use the appropriate `[DataObject]` and `[DataObjectMethod]` attributes for the class and its methods. The following two methods have been coded for you and simply need to be uncommented in order to work: `Theatres_List` and `Theatres_Get`.

For the **`LocationController`** class, create a method to return the `LocationID` and `Description` information for all the locations in the database. Create your own `LocationInfo` view model class to represent this location information, and use it in your method when querying the database for the `Location` entities.

Activity 3 - WebForm: `AssessmentPages/AssessmentCRUD.aspx`

Use a **ListView** to implement CRUD ODS processing for the Theatre entity. Add an ObjectDataSource control to access the CRUD DataObjectMethods within your Theatre controller. The ListView must support adding, updating and deleting theatre records. Customize the ListView's column titles and disable editing of the primary key column. Use a DropDownList (via an ObjectDataSource) for the Location foreign key. Use the `MessageUserControl.ascx` in the `UserControls` folder to handle all exceptions from your ObjectDataSource control.



Activity 4 - Complete the `MovieController` and `MovieInfo` Classes

Create a view model class called `MovieInfo` which represents the data to be displayed in the query page of this assessment. The class must have properties for the **Title**, **Year**, **Rating**, **Genre**, **ScreenType**, and **Length** of the movie.

Edit the **MovieController** class by adding a `[DataObjectMethod]` called **Movies_ListByYear**. Use this method to retrieve the movie information from the database, returning a list of `MovieInfo` data.

Activity 5 - WebForm: `AssessmentPages/AssessmentQuery.aspx`

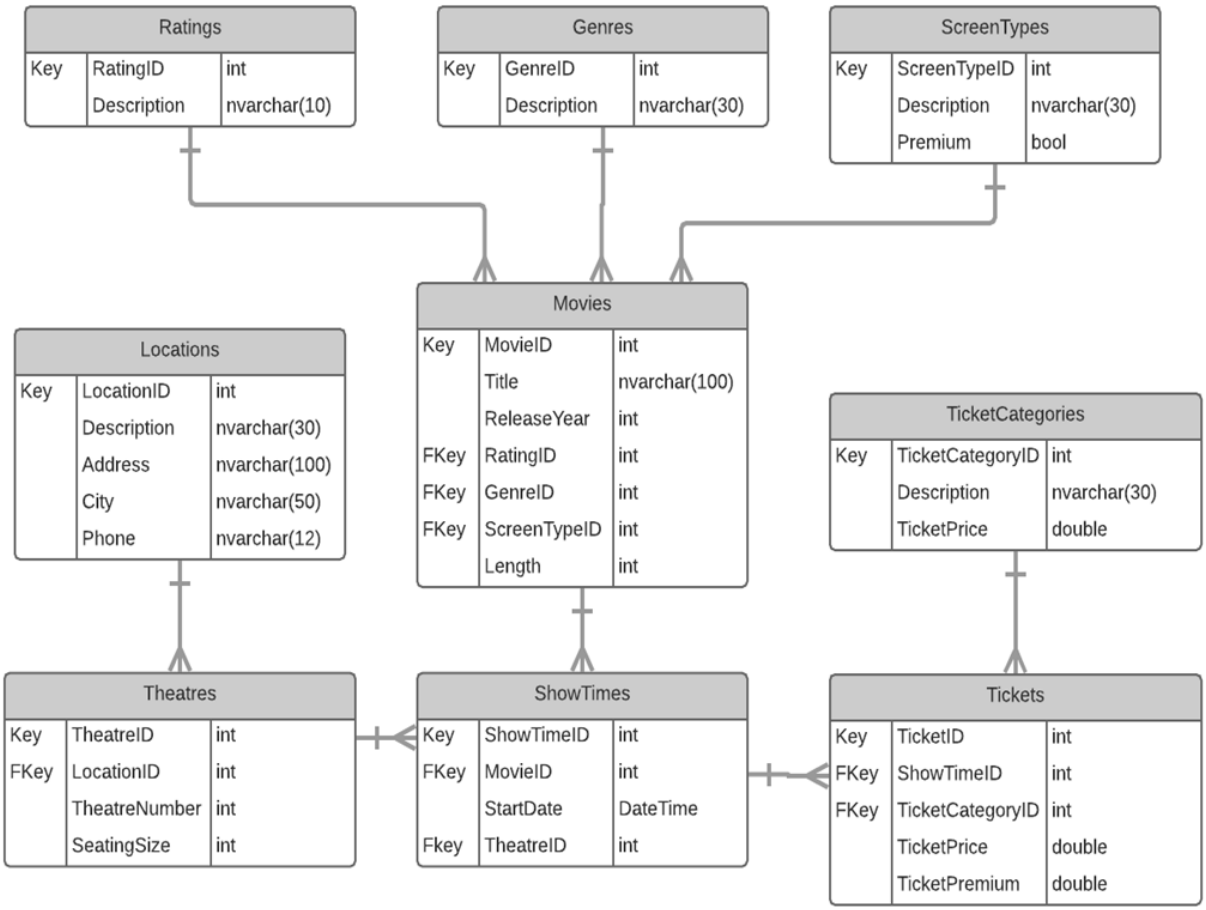
Edit the `AssessmentQuery.aspx` page to display the Movie Info from the database. Use an ObjectDataSource to access the DataObjectMethod within your controller class. Customize GridView control's column headings and enable paging. Use the Bootstrap `table` and `table-striped` classes on your GridView.

Assessment: Query

Title	Year	Rating	Genre	ScreenType	Length
It's a Wonderful Life	1946	G	Romatic	Normal	133
Blazing Saddles	1974	14+	Comedy	Normal	95
Star Wars 3D	1977	14+	Action	3D	125
Blade Runner	1982	18+	Action	Ultra	117
Jaws 3D	1983	G	Adventure	3D	99
The Princess Bride	1987	G	Adventure	Normal	98
Back to the Future	1995	PG	Adventure	Normal	116
Independence Day	1996	14+	Adventure	3D	153
Finding Nemo	2003	G	Animation	Normal	100
Ratatouille	2007	G	Animation	Normal	111

1 2

OMST ERD



OMST ERD

All fields are required