C# Review/OOP Practice - Entity Framework Code First Migration

Deadline: Friday, September 18th 2020, 09:00
[GitHub Classroom Repository](#)

Introduction

This practice is designed to help you familiarize yourself with the fundamentals of using Entity Framework to generate a database. This scenario is for when you would like to build your server and database both using C#. We will be building a simple database which will be expanded upon in future practices. Our database will eventually store "Shelves" made of different materials, with each shelf containing items. In this practice, we will only deal with the "Shelves" table.

Requirements

- Create a new console application in Visual Studio called CodeFirstPractice.
  - Add the "Pomelo.EntityFrameworkCore.MySql" package.
- Create **models** and a **context** for a simple database with the following:
  - A table called "Shelves" with:
    - An int primary key called ID.
    - A varchar of length 50 called Name.
- Use Entity Framework to create a database called "code_first_practice" using your models.
  - This must be done in a **migration**.
- Write a program that will take in a shelf name and add it to the database. For example, CreateShelf("Games Shelf") or AddShelf("Movies Shelf").

Challenges

- Research Data Seeding - EF and seed the table with 5 shelves when you push it to MySQL.
- Ensure that the shelves to be inserted by the user do not already exist (case insensitive, trimmed).

Reminders

Ensure you are tracking your time and that your timesheet is in the appropriate folder for viewing and marking.

Ensure you are committing frequently. It is advised to commit once per successful feature implementation at a minimum.

Ensure you are pushing your repository to the GitHub Classroom repository, and not a personal, private repository.

Ensure your repository has a readme with at a minimum your name, the name of the project, the project's purpose and a link to your Trello board.

Ensure you are using Trello appropriately to keep track of outstanding features, and that it is linked in the README.md file.

Ensure you are using the #class channel to request clarifications on assignment specifications.

Ensure you are using the #homework-help channel in slack to request for assistance from instructional staff if needed, and that you include (at a minimum) a specific description of the problem and a list of what you have tried.

Feel free to reach out on #peer-support if the #homework-help queue is lengthy. If someone helps you out, [give them a shoutout using our handy-dandy form](#)!

Citation Guide for Borrowed Code

Whenever you borrow code, the following information must be included:

- Comments to indicate both where the borrowed code begins and ends.
- A source linking to where you found the code (URL, book, example, etc.).
- Your reason for adding the code to your assignment or project instead of writing it out yourself.
- Explain to us how the code is supposed to work, include links to documentation and articles you read to help you understand.
- A small demonstration to prove you understand how the code works.

```javascript
1    const inputArr = [5,1,3,4,2];
2
3    /*Borrowed code for bubbleSort starts*/
4    let bubbleSort = (inputArr) => {
5        let len = inputArr.length;
6        for (let i = 0; i < len; i++) {
7            for (let j = 0; j < len; j++) {
8                if (inputArr[j] > inputArr[j + 1]) {
9                    let tmp = inputArr[j];
10                   inputArr[j] = inputArr[j + 1];
11                   inputArr[j + 1] = tmp;
12               }
13           }
14       }
15       return inputArr;
16   };
17
18   /*Borrowed code from bubbleSort ends*/
19
20   //Source: bubbleSort function obtained from https://medium.com/javascript-algorithms/javascript-algorithms-bubble-sort-3d27f285c3b2
21   //Reason to add: implementing bubble sort can be tedious and bug prone, it would be better to use a proven version than to write my own
22   //How it works: I read the following article to understand how bubble sorts work (http://www.pkirs.utep.edu/CIS3355/Tutorials/chapter9/tutorial9A/bubblesort.htm)
23   //Demonstration of understanding:
24   //Example array: [3,1,2]
25   //Step 1: Compare 3 and 1. Since 1 is smaller, swap places.
26   //Array: [1,3,2]
27   //Step 2: Compare 3 and 2. Since 2 is smaller, swap places.
28   //Array: [1,2,3]
29   //Step 3: Compare 1 and 2. No need to swap.
30   //Array: [1,2,3]
31   //Step 4: Compare 2 and 3. No need to swap.
32   //Array: [1,2,3]
33   //Function complete.
34   console.log(bubbleSort(inputArr));
```