IT 518 - Final Exam:

Developing Web Applications Using ASP.NET MVC Framework

Begin Date : 09:00 Monday, January 25

Due Date : 23:59 Friday, January 29

Part 1: Developing Single Page Application with AngularJS [50 Points]

1. Create an ASP.NET application named "**SensorsWebApp**" in Visual Studio.

> **Note**
> Many of the views in this project will be constructed using HTML files, and the client mostly requests JSON from the server (except for the initial request). The best fit for this scenario is the Web API project template. ?

2. Install Angular within your project. In this ASP.NET MVC project, the home page is one such page, so you can modify the **Index.cshtml** view for the **HomeController** to include Angular.
3. Add a "**Sensor**" class to the Models folder, and this class holds the information you want to store in the database. You'll also need a "**DbContext**" derived class (**SensorDb**) with a "**DbSet**" typed property (**Sensors**) to add, delete, and query sensor objects. Create a local database using Entity Framework Code First workflow option. Enable Entity Framework migrations and use migrations to seed the database with at least five initial data. Also enable automatic migrations to make the process of adding new features easier.

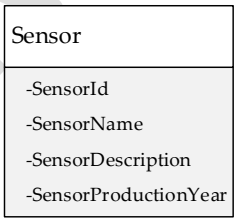


Figure 1. Sensor Model Class Diagram

4. Build a Web API named "**SensorController**". The Web API is simple to build because you only need basic create, read, update, and delete functionality, and the scaffolding provided by Visual Studio can generate the code you need.
5. Building out the features required for managing sensor requires a proper application with modules. A module in Angular is an abstraction that allows you to group various components to keep them isolated from other components and pieces of code in an application. One of the benefits of isolation is to make Angular code easier to unit test, and easy testability is one of the goals the framework creators set out to achieve. Various features of the Angular framework are organized into different modules that you'll need for the

application. But, before you use these modules you need a custom module for your application itself. To start, follow these steps:

- Create a new folder in the project named **Client**. The idea is to organize the scripts for the home page application into a dedicated folder instead of using the existing Scripts folder to hold all JavaScript.
- Inside the **Client** folder, create a subfolder named **Scripts**. Use this folder to hold your Angular scripts.

6. Create the Angular controllers that provides to manage CRUD operations for the sensors.
   - ListController.js
   - EditController.js
   - DetailsController.js
   - SensorService.js
   - Sensors.js
7. Create a new folder named "**Views**" inside the "**Client**" folder. Create the views into this new folder.
   - details.html
   - edit.html
   - list.html

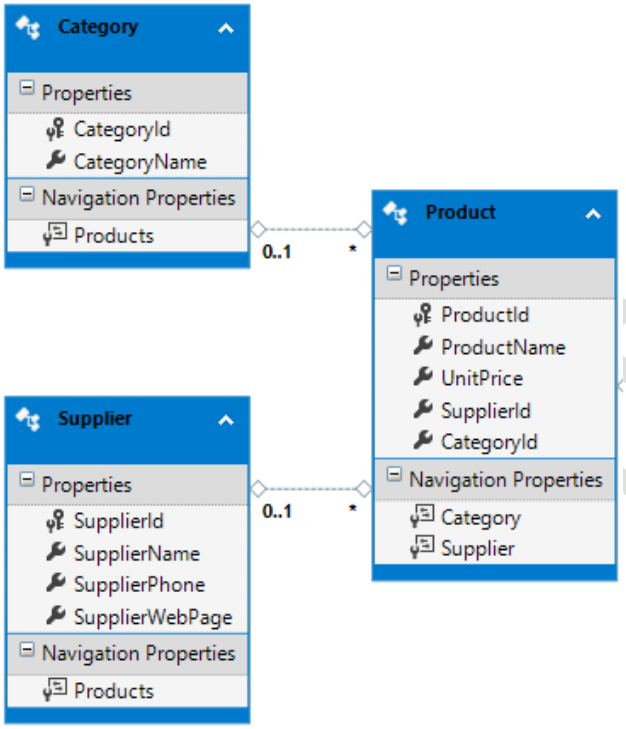
> **Note**
> Routing in Angular is conceptually similar to routing in ASP.NET. Given some URL, such as /home/index/#details/4, you want the application to respond by loading a specific controller, view, and model, and also give you controller information about parameters encoded into the URL. Angular can take care of the preceding requirements, but you do need to download some additional modules and apply some configuration to the application. Install the Angular routing module using NuGet. Include the routing module in the scripts section of Index.cshtml.

8. Complete your project which enables users to create a new sensor, update an existing sensor delete an existing sensor, and list stored sensors.

Part 2: Developing Web API and Desktop Client Application [50 Points]

1. Create a Web API project.
2. Create three models named **Product**, **Category**, and **Supplier**, respectively.

UnitPrice:
decimal(16,2) ?



Google:

- connectionString for LocalDB

- LocalDB creation with EF 6 using Code First

Figure 2. Entity Relationship Model

3. Create a local database named "**StocksDb**" using Entity Framework Code First Workflow option. Enable Entity Framework migrations and add the needed code to the Seed method to populate the database with sample data. Furthermore, enable automatic migrations to make the process of adding new features easier.

> Notice that as shown in Figure 2, there are relationships between the entities. Provide the relationships using Entity Framework Code First.

4. Create the Web API controllers using the "Web API 2 Controller with read/write actions, using Entity Framework" option. The Web API is simple to build because you only need basic create, read, update, and delete functionality, and the scaffolding provided by Visual Studio generates the code you need.
   - ProductController
   - CategoryController
   - SupplierController
5. Create a Windows Forms Client Application to consume the Web API. This client application is responsible to manage CRUD operations for the product using Product API. You need to display the suppliers and the categories into **ComboBox** controls using Category and Supplier APIs.

- ***MvcMusicStore*** and ***Midterm*** Projects -

Multiple (!) Tables, SIMILAR to FINAL!
Notice the ***NAVIGATION*** Properties in these TWO projects!
Code First

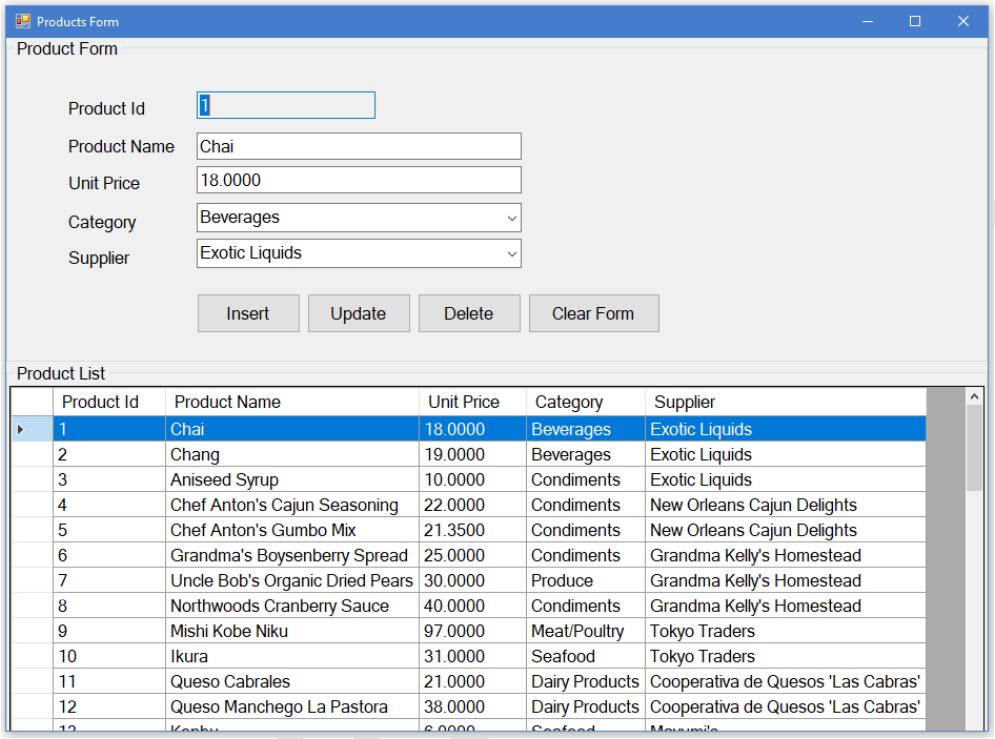
MusicStore = WebStore (Midterm)
Category = Genre
Supplier = Artist
Product = Album

- Notice that the first item of the ComboBox controls should be "**Select an item**". When binding data sources of the ComboBox controls, you need to set the **DataSource**, **DisplayMember**, and **ValueMember** properties.

Example:

```
SuppliersComboBox.DataSource = GetSuppliers();
SuppliersComboBox.DisplayMember = "SupplierName";
SuppliersComboBox.ValueMember = "SupplierId";
```

6. Figure 3 provides the UI design of the product form.



REVIEW ...

Figure 3. Windows Forms Client Application UI Design

- **Midterm** - MVC Web Store - ***Multiple*** (!) Tables, Code First, Seed YOK! yazmamışım
- **Web API hw** - EmployeeAPI - Single Table, Code First, ***Seed,*** Windows Forms,

- **Music Store** - MVC, Code First, ***Seed YOK!!!,... Multiple*** (!) Tables with Navigation Properties, ***LocalDB*** in ***App_Data*** Folder (week4, video2 minute 32) - weeks 2,4,5,6,7
- **Tool Shop** - MVC, Paging, Database First, Repository (Abstract, Concrete), Multiple Tables but NOT DB First, /Content (css, js, image) - week 9,10,11 - Put Entities and Model into 2 different Folders in DB First! - 3 Database ***STORED PROCEDURES*** has been written in this project!
- **Movie** - Web API, Windows Forms Client, ***Seed,*** Code First !?, asynch/await, Newtonsoft.JSON - week 12
- **AtTheMovies** - Angular - Code First, ***Seed, LocalDB*** LocalFolder ***App_Data*** video1 minute~18 - week 13
- **Sports Store** - MVC, Dependency Injection, Domain, WebUI, UnitTests, Ninject, Moq, Paging - week 14

- EF Workflow Types Projects - Windows Forms - week 3
--- Code First
--- DB First
--- Model First

Creating a Connection String and Working with SQL Server ***LocalDB*** (like Final Exam!)
https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/introduction/creating-a-connection-string

Working with SQL Server ***LocalDB (App_Data*** folder!)
https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/introduction/accessing-your-models-data-from-a-controller

GRADING POLICY

Part 1	
Creating Models, View Models, and Database	**5 Points**
Creating Controllers and Actions (MVC)	**5 Points**
Designing HTML Layouts	**10 Points**
Creating Angular Controllers	**20 Points**
Accurate Working of CRUD Operations	**10 Points**

Part 2	
Creating Models, View Models, and Database	**10 Points**
Creating WEB API Controllers and Actions	**10 Points**
Designing Client App	**10 Points**
Accurate Working of CRUD Operations	**20 Points**

CHEATING AND PLAGIARISM POLICY:

Instances of academic dishonesty (cheating, copying, and plagiarism) will not be tolerated. This applies to all exams, and assignments. You are expected to complete your assignments on your own (unless indicated otherwise) and submit your own work. The punishment for cheating at exams or plagiarizing other people's work will be at the instructor's discretion.

The punishment may be:

(a) receiving zero grade in the particular test or assignment;
(b) receiving an FF as the final grade;
(c) official disciplinary action.