**Building an Automobile Management Web Application using ASP.NET Core MVC and Entity Framework Core**

# Introduction

Imagine you're an employee of an online car retailer named **Automobile eStore**. Your manager has asked you to develop a Web application for automobile management (CarID, CarName, Manufacturer, Price, and ReleasedYear). The application has to support adding, viewing, modifying, and removing products—a standardized usage action verbs better known as Create, Read, Update, Delete (CRUD).

This lab explores creating an application using ASP.NET MVC with .NET Core, and C#. An **SQL Server** **Database** will be created to persist the car's data that will be used for reading and managing automobile data by **Entity Framework Core (EF Core)**

# Lab Objectives

In this lab, you will:

* Use the Visual Studio.NET to create ASP.NET Core MVC and Class Library (.dll) project.
* Create a SQL Server database named MyStock that has a Cars table.
* Develop a DataProvider class to perform CRUD actions using EF core.
* Apply passing data by ViewBag in ASP.NET Core MVC application.
* Apply Repository pattern and Singleton pattern in a project.
* Add CRUD action methods to ASP.NET Core MVC application.
* Run the project and test the ASP.NET Core MVC actions.

MyStock Database

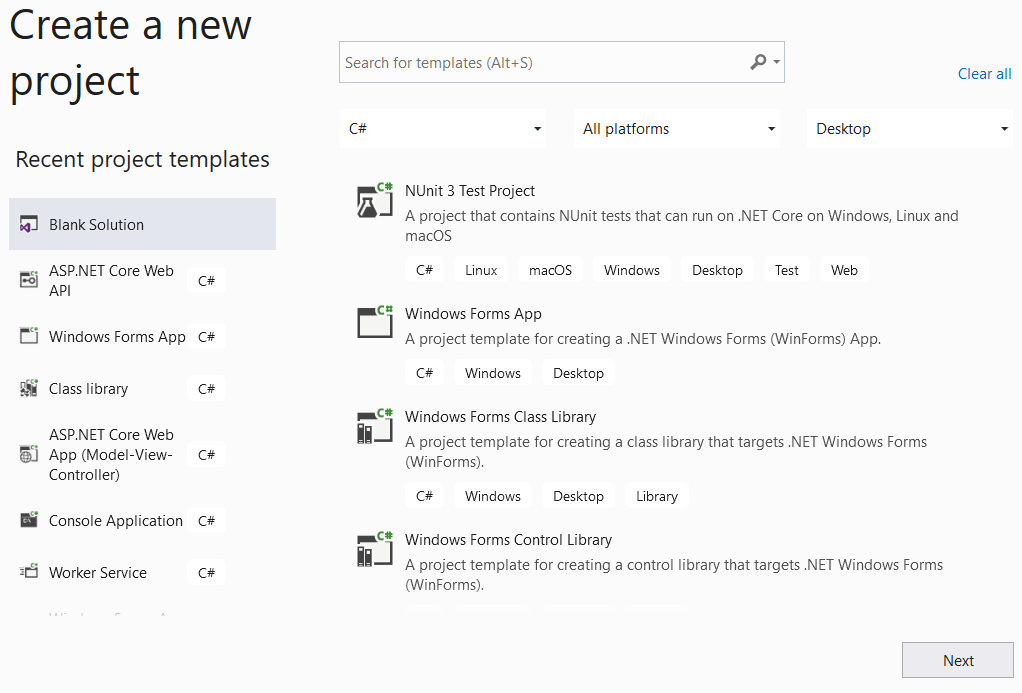
# 

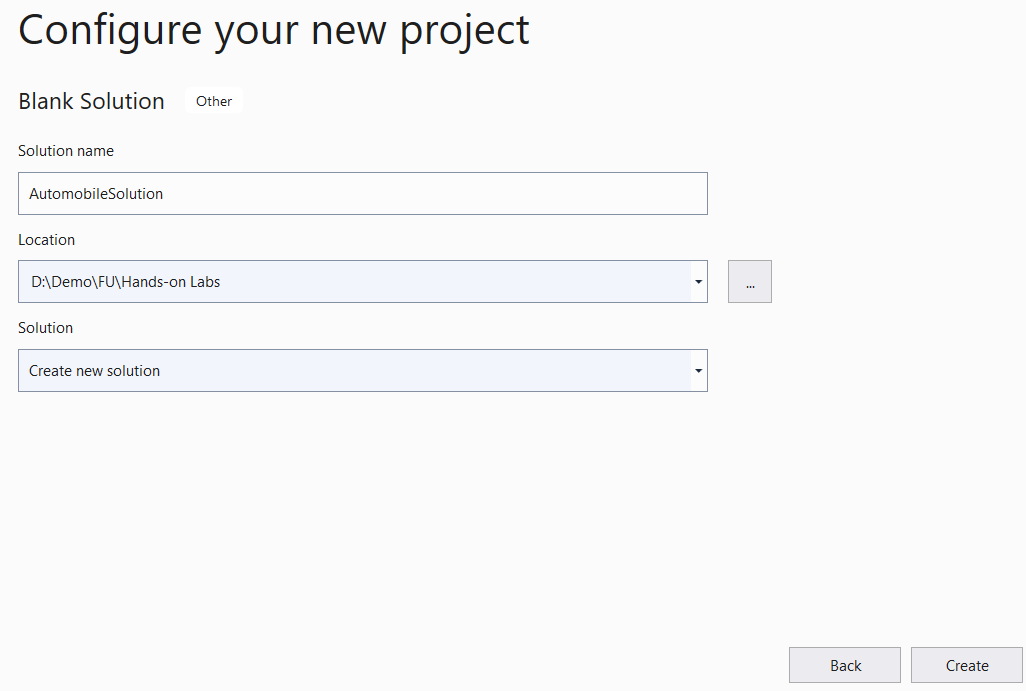
# Activity 01: Build a solution by Visual Studio.NET

Create a Blank Solution named **AutomobileSolution** then add new a Class Library Project named **AutomobileLibrary** and a Windows Forms project named **AutomobileWinApp**

**Step 01**. Create a Blank solution.

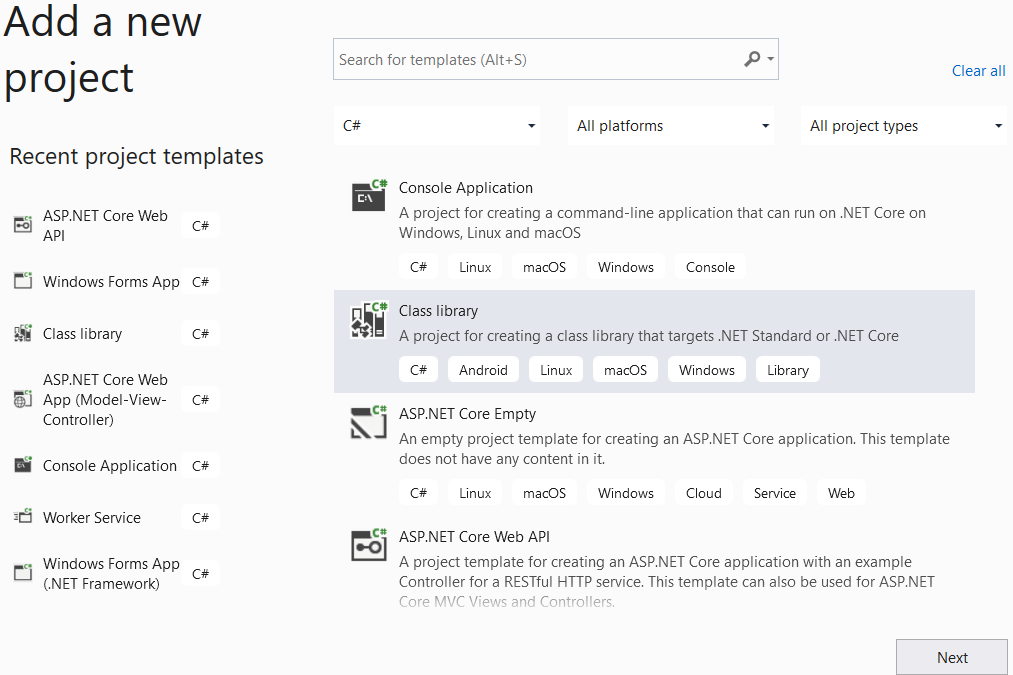
* Open the Visual Studio .NET application and performs steps as follows:

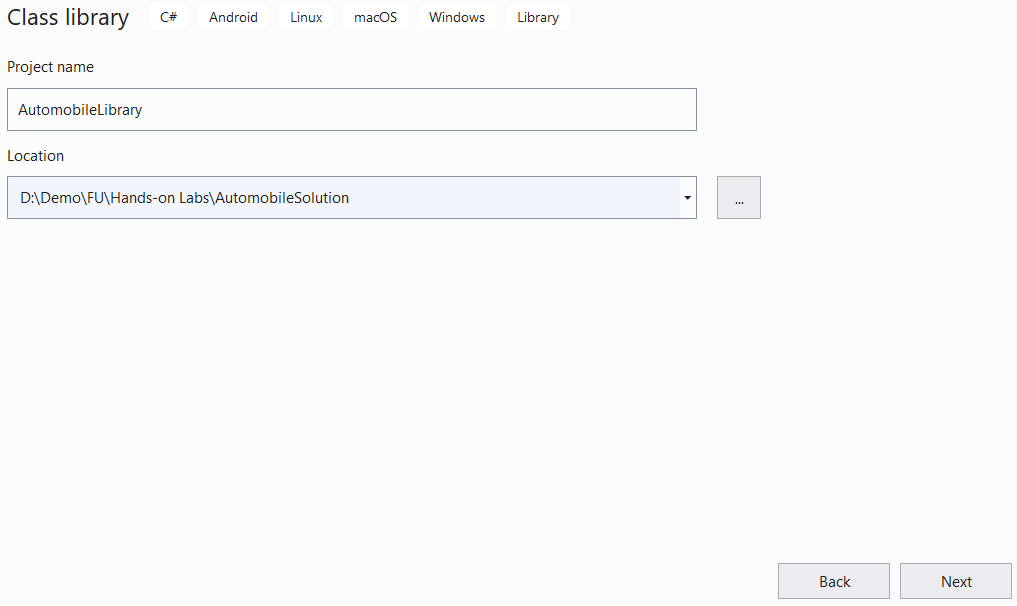




**Step 02.** Create a Class Library project.

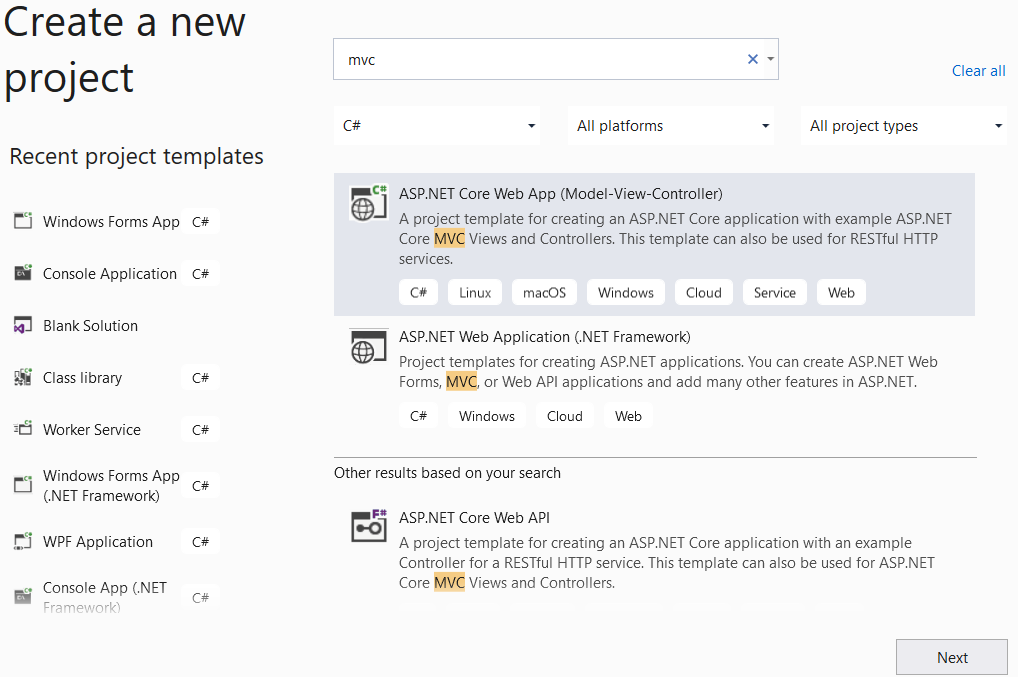
* From the File menu | Add | New Project, on the Add New Project dialog, select “Class Library” and performs steps as follows:

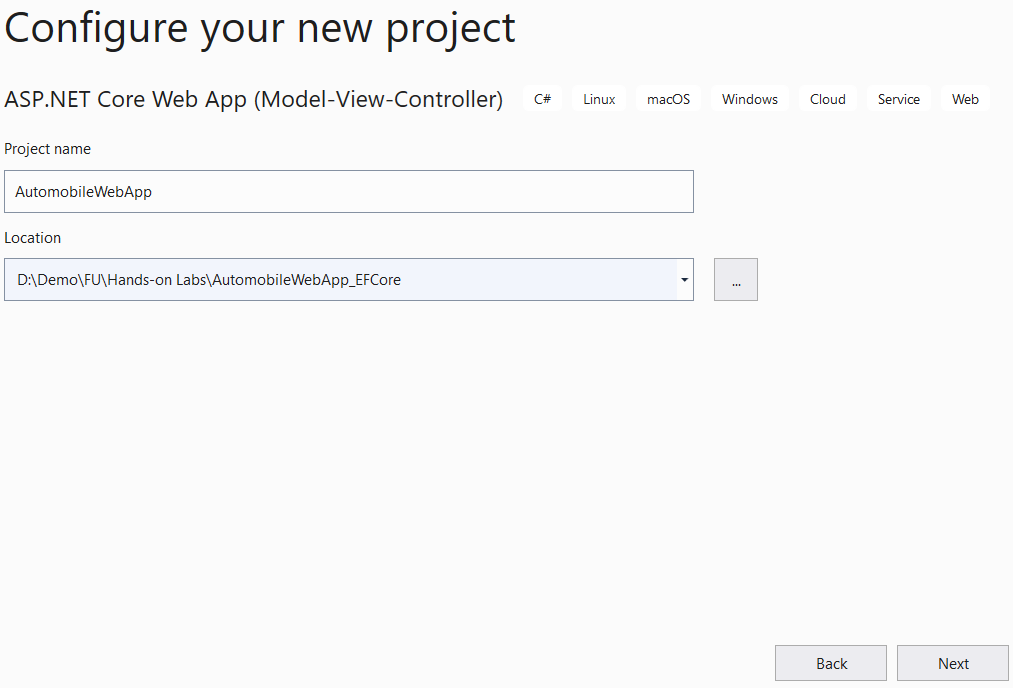


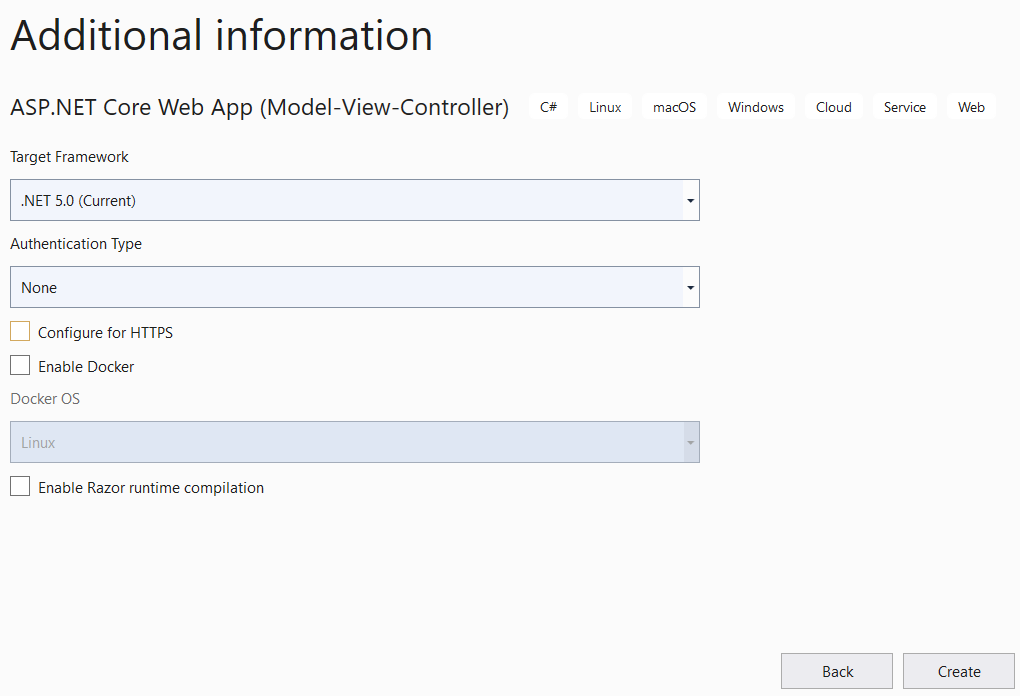


****

**Step 03.** Create an ASP.NET Core MVC project.

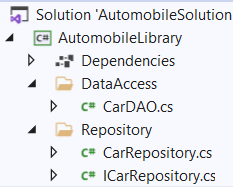




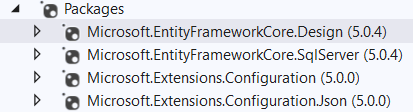


# Activity 02: Write codes for the AutomobileLibrary project

**Step 01**. Create folders and add classes to the project as follows:

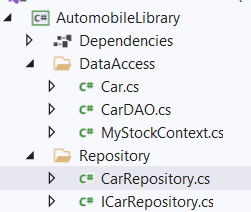


**Step 02**. Install the following packages from Nuget:

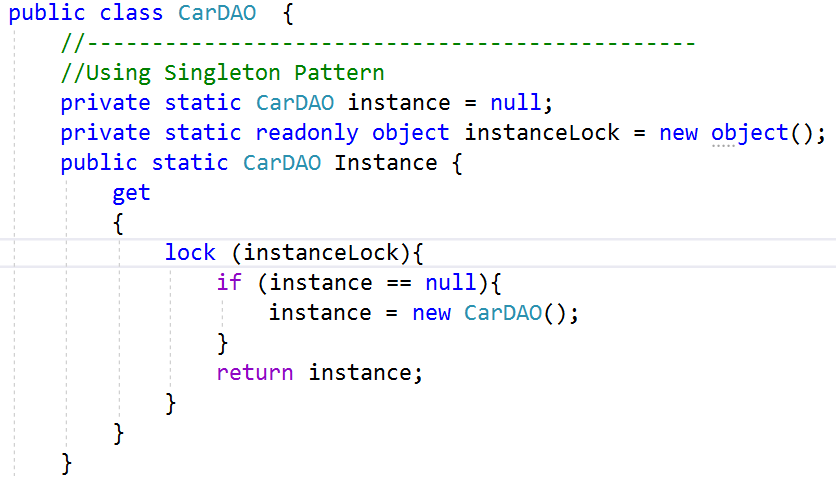


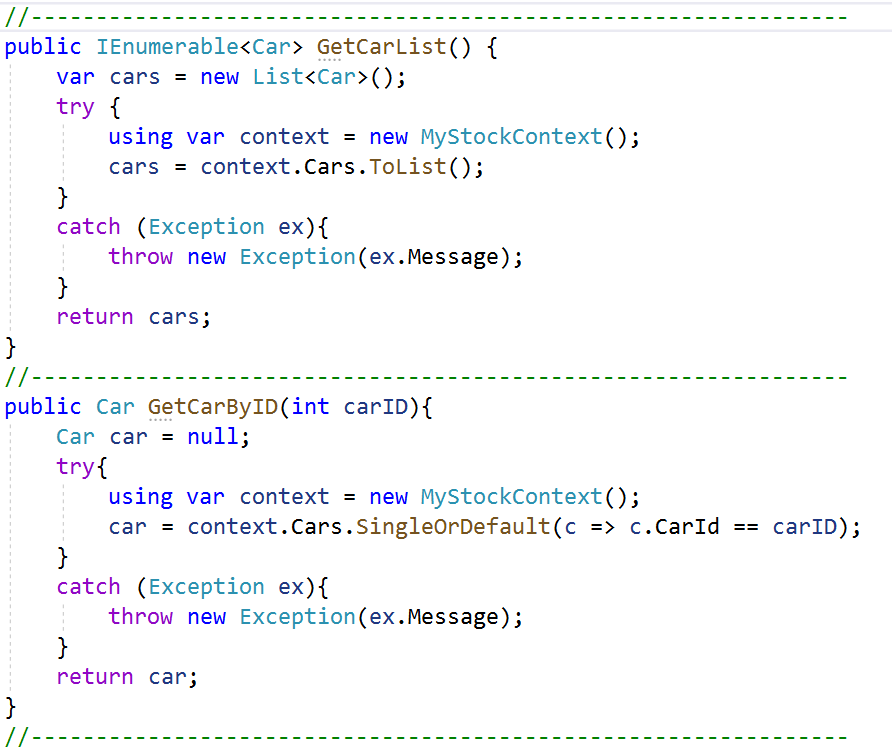
**Step 03**. Right-click on the project, select **Open in Terminal.** On Developer PowerShell dialog, run **dotnet ef** command to generate database context and Car entity model from **MyStock** database as follows:

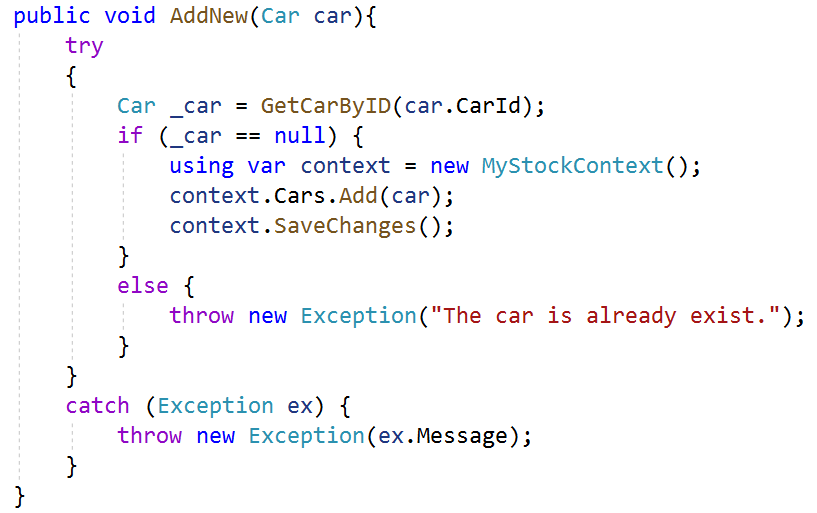
dotnet ef dbcontext scaffold "Server=(local);uid=sa;pwd=123;database=MyStock" Microsoft.EntityFrameworkCore.SqlServer --output-dir DataAccess

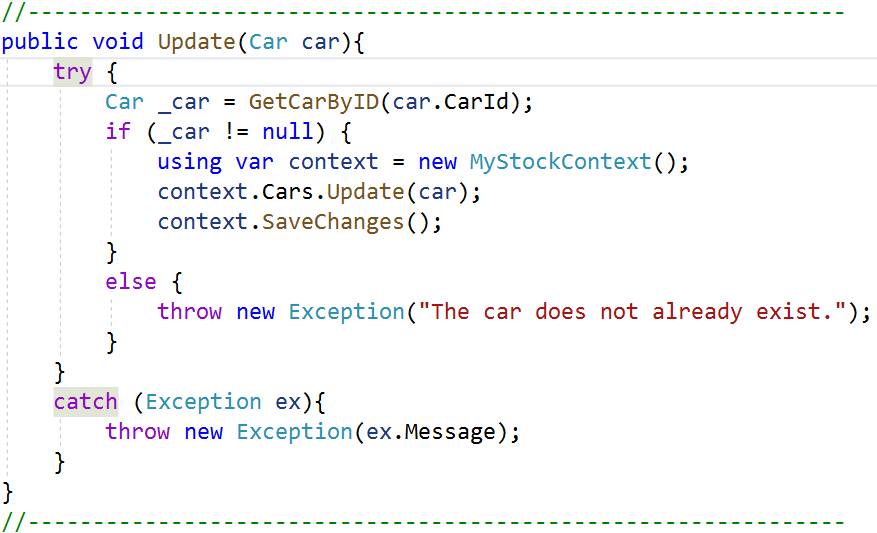


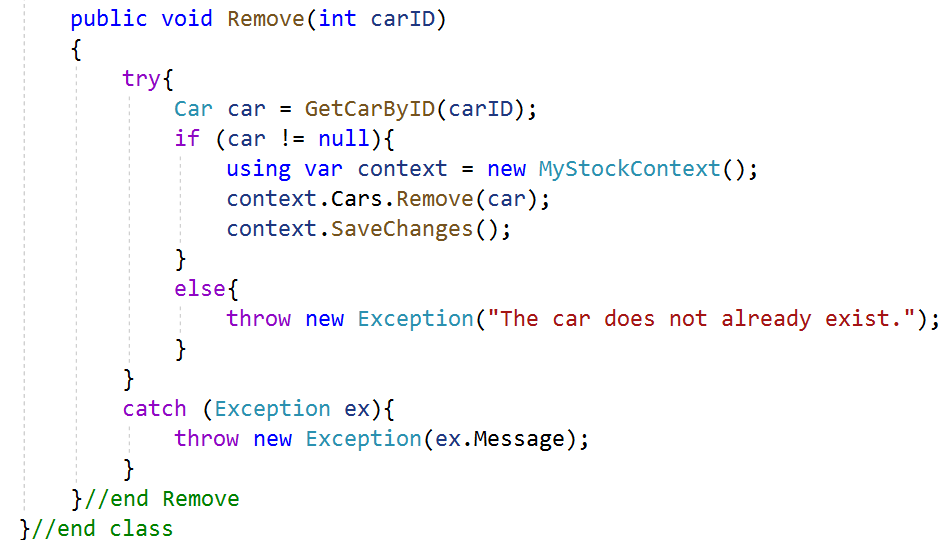
**Step 04**. Write codes for **CarDAO** class of **CarDAO.cs** as follows:



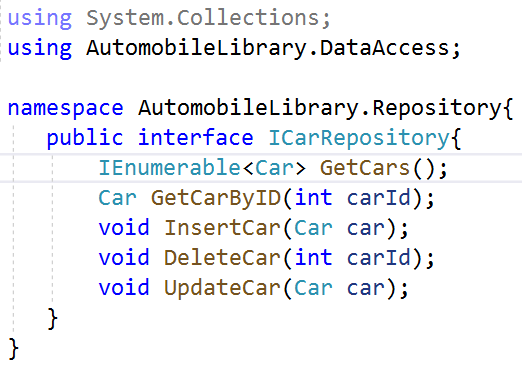




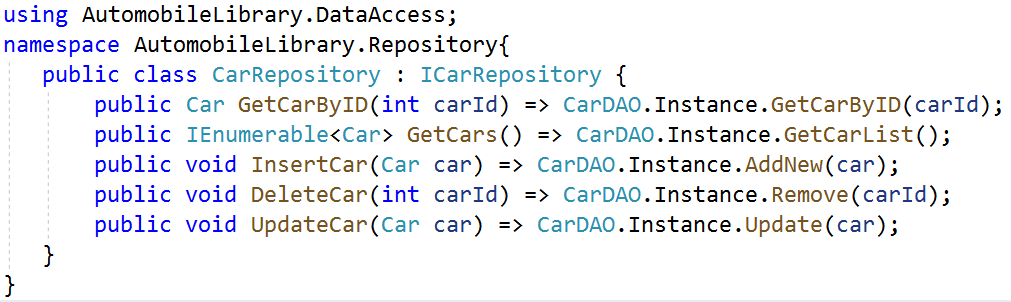




**Step 07**. Write codes for **ICarRepository.cs** as follows:

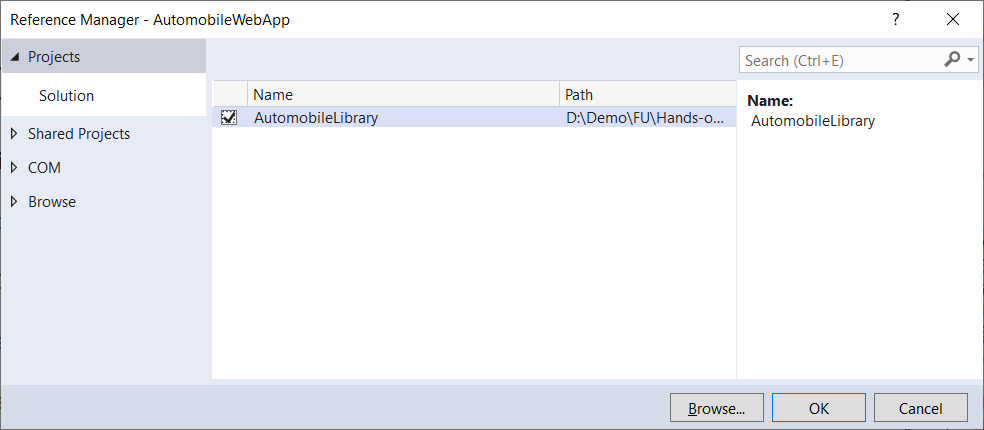


**Step 08**. Write codes for **CarRepository.cs** as follows:



# Activity 03: Reference to AutomobileLibrary project and write codes for ASP.NET Core MVC project

# Step 01. Right-click on AutomobileWebApp project, select Add | Project Reference, and perform as the below figure:



# Step 02. Right-click on the AutomobileWebApp project, select Add | Controller and perform as the below figure then click Add.

# 

# On the next dialog, enter the controller name is CarsController.cs, click Add to finish.

# Step 03. Write codes for action methods of CarsController.cs as follows:

using AutomobileLibrary.Repository;

# using AutomobileLibrary.DataAccess;

# 

# 

# 

# 

# 

# 

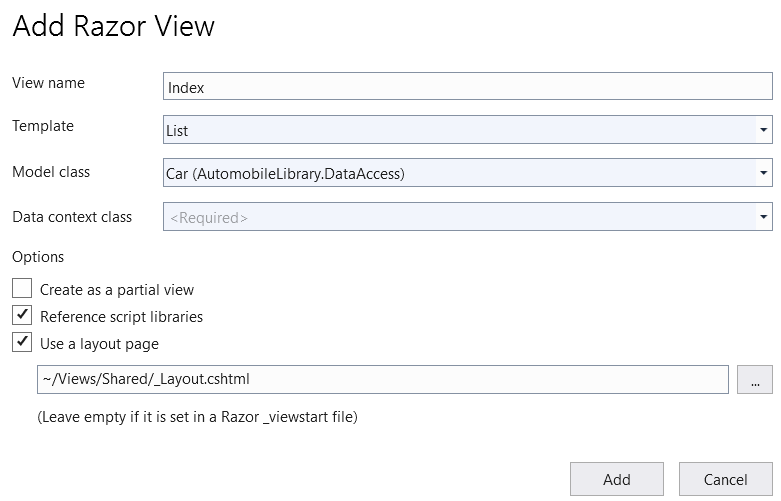
# Activity 04: Create views AutomobileWebApp project

# Step 01. Right-click on View folder | Add | New Folder named Cars then right-click on Cars folder | Add | View named Index as follows:

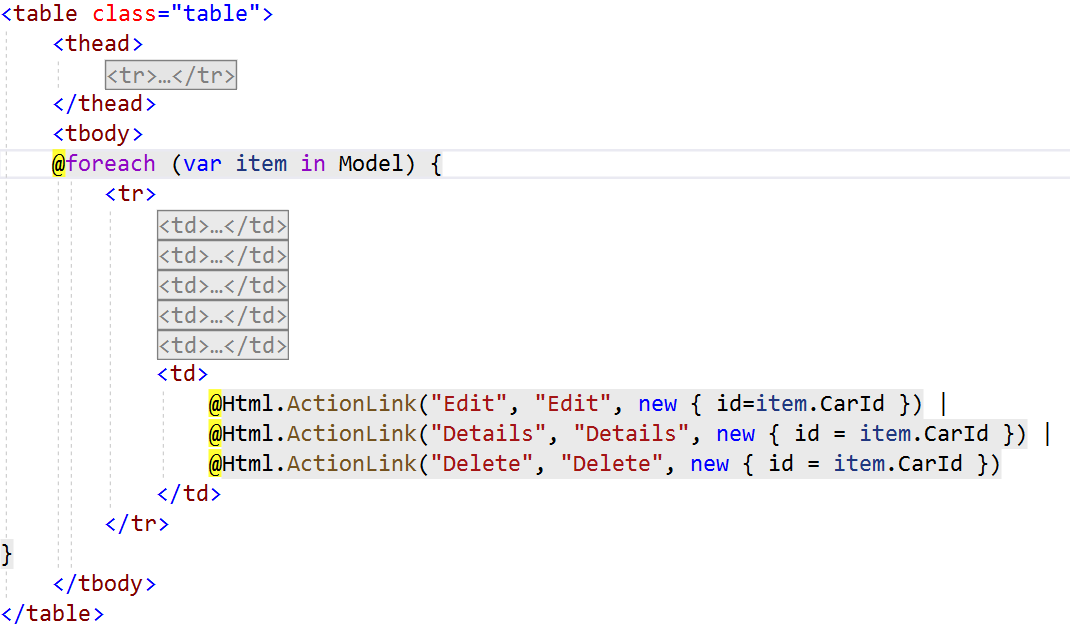
# Step 02. Right-click on Cars folder | Add | View named Index as follows then click Add.

# 

On the next dialog, setup as the below figure then click **Add** to finish.



# Step 03. Open Index.cshtm and update codes for table tag as follows:



# Step 04. Add Details view as the below figure:

# 

# Step 05. Add Create view then open this view and update as follows:

# 

# 

# Step 06. Add Edit view as the below figure:

# 

# Step 07. Add Delete view as the below figure:

# 

# Activity 05: Run AutomobileWebApp project and test all actions

# Step 01. Press Ctrl+F5 to run the project then edit the link on the web browser as follows:

# http://localhost:27136/Cars/Index

# The result will show in the below figure:

# 

# Step 02. Click Edit link and display the result as the below figure, enter the values on text fields then click Save to update the car.

# 

# Step 03. Click Details link to view Car details.

# 

# Step 04. Click Create New link and display the result as the below figure, enter the values on text fields then click Save to add a new car.

# 

# Step 05. Click Delete link and display the result as the below figure. Click Delete button to remove Car

# 