Code-First Approach In MVC 5

Here, we are going to learn about the Code First approach in MVC. Basically, I am used to the Database-First approach. It is obvious that both the Code-First approach and DB First approach are useful in their own context but I would like to summarize the basic usage of these as per my working experience.

When to use DB-First Approach?

* When you have to work in a project where DB is already created.
* You don’t have to alter DB columns, stored procedures, and views.
* When your project is small and doesn’t have to be very modular.
* When you don’t want to get a headache writing simple classes because DB-First approach in Entity will write all the code for class.

When to use Code- First?

* When you have to make your application from scratch.
* When you don’t have a readymade database.
* You have to play with database frequently, like creating and deleting tables, views, and stored procedure.
* When you have too many tables, views, and stored procedures in database.

**What we will learn?**

1. Creating a blank database.
2. Creating a MVC Project (UI)
3. Creating a Class Library Project (DAL)
4. Code First Approach implementation.
5. Migration in MVC
6. Creating CRUD Operation.

Creating Blank Database

**Step 1**   
  
Open SQL Server.

**Step 2**    
  
Connect to Database Server.

Graphical user interface, text, application, email

Description automatically generated

**Step 3**

Right click on Database to "Create New Database".

Graphical user interface, application

Description automatically generated

You will be able to see your newly created database on Server.

Creating a MVC Project

**Step 1**  
Open Visual Studio.

**Step 2**  
Go to File => New Project.

1. Select Visual C# => Web => ASP.NET Web Application.
2. Name your Solution (eg. EasyCRM) and Project (EasyCRM.UI) and then click OK.
3. Select MVC.
4. Change Authentication to "Individual User Accounts".  
     
   Graphical user interface, application

   Description automatically generated

Now, you will see the following screen.

Text

Description automatically generated with low confidence

Creating a class Library Project (DAL)

**Step 1**  
Right click on Solution Explorer to add new project.

Graphical user interface, application

Description automatically generated  
**Step 2**  
Select Visual C# => Windows => Class Library.

Graphical user interface, application

Description automatically generated

**Adding Entity Framework to DAL Project.**

**Step 1**  
Right click on DAL Project and select "Manage NuGet Packages".

Graphical user interface, text, application

Description automatically generated

**Step 2**   
  
Browse Entity Framework and Install.

Graphical user interface, text, application, email

Description automatically generated

Code First Approach

Now, add 3 folders to EasyCRM.DAL Project named as

1. DataConnection
2. Entity
3. Identity

**Creating Entity for Customer**

* Add a new folder inside Entity, named as Customer.
* Add New Class called Customer\_PersonalDetail.cs.
* Add the following code inside the class.

**public** **string** CP { **get**; **set**;}

[Required(ErrorMessage = "City required")]

**public** **string** City {**get**;**set**;}

[Required(ErrorMessage = "Contact person required")]

[DisplayName("Contact person")]

**public** **string** ContactPerson {**get**;**set**;}

[Required(ErrorMessage = "Telephone required")]

[DisplayName("Telephone")]

[Phone]

**public** **string** Phone {**get**;**set**;}

[Required(ErrorMessage = "Email required")]

[RegularExpression(@ "\w+([-+.']\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\*", ErrorMessage = "Wrong email format")]

[EmailAddress]

**public** **string** Email {**get**;**set**;}

**public** **string** Notes {**get**;**set**;}

* Add new class called "Users" under Entity folder.
* The new class file should look like it -
  1. **using** System;
  2. **using** System.Collections.Generic;
  3. **using** System.Linq;
  4. **using** System.Text;
  5. **using** System.Threading.Tasks;
  6. **using** Microsoft.AspNet.Identity.EntityFramework;
  7. **using** System.ComponentModel.DataAnnotations;
  8. **namespace** EasyCRM.DAL.Entity {

**public** **class** Users : IdentityUser {

        [Required]

**public** **string** Name {**get**;**set**;}

        [Required]

**public** **string** Address **get**;**set**;}

        [Required]

**public** **string** Gender {**get**; **set**; }

    }

* 1. }

**Step 3**   
Add new class in DataConnection folder, named DataContext.cs. Don’t forget to make this class public

**Add Following References**

**using** EasyCRM.DAL.Entity;

**using** EasyCRM.DAL.Entity.Customer;

**using** EasyCRM.DAL.DataConnection;

**using** Microsoft.AspNet.Identity.EntityFramework;

**using** System.Data.Entity;

Now, write the codes in a similar way.

1. **using** System;
2. **using** System.Collections.Generic;
3. **using** System.Linq;
4. **using** System.Text;
5. **using** System.Threading.Tasks;
6. **using** System.Data;
7. **using** EasyCRM.DAL.Entity;
8. **using** EasyCRM.DAL.Entity.Customer;
9. **using** EasyCRM.DAL.DataConnection;
10. **using** Microsoft.AspNet.Identity.EntityFramework;
11. **using** System.Data.Entity;

**namespace** EasyCRM.DAL.DataConnection {

**public** **class** DataContext : IdentityDbContext <Users> {

**public** DataContext() **base**("DefaultConnection") {}

**public** DbSet <Customer\_PersonalDetail> Customer\_PersonalDetail { **get**;**set**;}

    }

}

**Referencing DAL Project to UI Project.**

**Step 1**Right click on References of UI Project to add references.

**Graphical user interface, text, application

Description automatically generated**

Enabling Migrations

**Step 1**  
Go to Tools => Package Manager =>Manage NuGet Packages for Solution.

Graphical user interface, text, application

Description automatically generated  
**Step 2**Now run these commands using Console.

Dont forget  to select the DAL Project. (Migrations will be created in DAL Project as in figure)

1. Enable-Migrations
2. add-migration InitialCreate
3. update-database

Once the migration is finished, you will see a Migrations folder with two files, as below.

Graphical user interface, text, application

Description automatically generated

Now, go to Database to browse tables. You will be able to see the following tables.

Graphical user interface, text, application

Description automatically generated

So far, we have created tables using Code First approach. Now, we will apply CRUD operation for *Customer\_PersonalDetail.*

**Adding Controller**

**Step 1**    
  
Right click on Controller of UI Project to add new Controller.

<Graphical user interface, text

Description automatically generated

**Step 2**   
  
Select "MVC 5 Controller with views, using Entity Framework".

Graphical user interface, text, application, email

Description automatically generated

**Step 3**   
  
Now, select Model class, Data context class, and layout page.

Graphical user interface, text, application, email

Description automatically generated

Now, you will see a Controller named "Customer" added in Controllers folder and views created on Views => Customer folder as in the figure.

Graphical user interface, application

Description automatically generated

**Managing Layout Page**

As I have been using AdminLTE Theme for my Admin Panel, I will use the same theme here. Please review "Adding AdminLTE Theme" on MVC 5 to learn working with Themes in MVC. Now, I will add two links - one, to Add New Customer; and next, to List Customer.

**Step 1**  
Open \_LayoutAdminPanel.cshtml as in figure

Graphical user interface

Description automatically generated with medium confidence

**Managing Server Connection String**

Now, we need to modify Server Connection String in web.config of UI Project.

**Step 1**  
Go to View => Server Explorer.

**Step 2**  
Now, connect to database as below.

Graphical user interface

Description automatically generated

**Step 3**   
  
Right click on connected server and select Properties to retrieve Connection String, as in figure.

Graphical user interface, text, application

Description automatically generated

**Step 4**   
  
Open web.config as in figure and replace the old Connection String with new Connection String, as in figure below.

Graphical user interface, text, application

Description automatically generated

**Execute Application**

Now, rebuild the solution (Ctrl+Shift+B) and run your application. You will be able to see the following screen.

Graphical user interface, application, chat or text message

Description automatically generated

Now, click "Add New Customer" to add new customer. Please note that we are adding new customer with Code First approach here.

Once you navigate to "Add New Customer" link, you will see the following form. Add details to the form and save.

Graphical user interface, application

Description automatically generated

You will be able to see the data saved in Database.

Table

Description automatically generated with medium confidence

Conclusion

So, in this way, we can implement Code-First approach in MVC 5 application.