Table of Contents

[Add Extensions / packages 2](#_Toc113815877)

[appSettings.json & appSettings.Development.json 2](#_Toc113815878)

[Constants 2](#_Toc113815879)

[ConfigKeyConstants.cs 2](#_Toc113815880)

[Extensions 2](#_Toc113815881)

[ConfigExtensions.cs 2](#_Toc113815882)

[ServiceExtensions.cs 2](#_Toc113815883)

[Extension Methods 2](#_Toc113815884)

[Registering the Service Extension Methods in Programs.cs 3](#_Toc113815885)

[Area one: do registration 3](#_Toc113815886)

[Area two: use CORS in the app 3](#_Toc113815887)

[Entity Framework 3](#_Toc113815888)

[Entities 3](#_Toc113815889)

[DBContext Class 4](#_Toc113815890)

[Add /DB/DataContext to Services 4](#_Toc113815891)

[Migrations and Database Update 4](#_Toc113815892)

[Migrations 4](#_Toc113815893)

[Database Update 4](#_Toc113815894)

[Completely remove all migrations and start all over again 4](#_Toc113815895)

[Drop database and recreate 4](#_Toc113815896)

[Seeding the Data 4](#_Toc113815897)

[Viewing the Database 4](#_Toc113815898)

[Add User dummy data using Sqlite explorer 5](#_Toc113815899)

# Add Extensions / packages

Check “0001 Project - Basic Info - Setup.docx”, check sections

1. Adding c# related extensions & packages in VS Code
2. Adding Packages

# appSettings.json & appSettings.Development.json

* AllowSpecificOrigins:
  + Add to both appSettings.json & and appSettings.Development.json. **Note:** do not put the trailing /

"AllowSpecificOrigins": ["https://localhost:4200","http://localhost:4200"]

* ConnectionStrings
  + Add only to appSettings.Development.json

"ConnectionStrings": {

"DefaultConnection": "Data source=Core/DB/MySocialConnect.db"

}

# Constants

Add following to the Constants folder

## ConfigKeyConstants.cs

This file will contain the key names used in app settings. Check the file for details.

# Extensions

Add following to the Extensions folder

## ConfigExtensions.cs

* This file will get extension methods to query items from the app settings.
* There are specific methods as well as generic methods which could be used from anywhere.
* Check file for more detail.

## ServiceExtensions.cs

This extensions will get methods which will help with registering the services in Programs.cs

### Extension Methods

#### RegisterRepos Extension Method – Dependency Injection

* Here we will register Interfaces against their concrete classes to be used with DependencyInjection.
* When you’ll open the class you’ll see registrations. There Interfaces and Concreate classes are created under document “[0005 WebApi Controllers - Repository - Dependency Injection.docx](0005%20WebApi%20Controllers%20-%20Repository%20-%20Dependency%20Injection.docx)”.
* In our case we’ll register Repository and BusinessLogic classes.

#### RegisterCors Extension Method

* Here we’ll allow specific origins that can access the WebApi.
* In our case it is The Angular – SPA.

#### RegisterDBContext Extension Method

* Here DataContext class will get registered.
* [DataContext](#_DBContext_Class) class is generated below.

## Registering the Service Extension Methods in Programs.cs

There are two “CUSTOM” areas

### Area one: do registration

//CUSTOM:Start

ConfigurationManager configuration = builder.Configuration;

builder.Services.RegisterRepos();

builder.Services.RegisterDBContext(configuration);

var myAllowSpecificOrigins = builder.Services.RegisterCors(configuration);

//CUSTOM:End

### Area two: use CORS in the app

//CUSTOM: Start

app.UseCors(myAllowSpecificOrigins);

//CUSTOM: End

# Entity Framework

For development will use Sqlite.

Diagram

Description automatically generated

# Entities

Create entities in MSC.Api/Core/Entities/AppUser.cs

1. AppUser: It will have two properties in the start
   1. int Id
   2. string UserName

# DBContext Class

Create a new /DB/DataContext.cs file. It will inherit from the DbContext.

Add DBSet for following entities

1. AppUser entity which will pull from Users.

## Add /DB/DataContext to Services

Check above [Services Extension](#_ServiceExtensions.cs), [method RegisterDBContext](#_RegisterDBContext_Extension_Method) and then registering the same in [Programs.cs](#_Area_one:_do)

# Migrations and Database Update

## Migrations

* dotnet ef migrations add InitialCreate -o Core/DB/Migrations
  + any time a change a made is to the entities run the above command, just change the InitialCreate part and then run update database command from below.
* dotnet ef migrations remove

## Database Update

After migrations update the database

* dotnet ef database update

## Completely remove all migrations and start all over again

* dotnet ef database update 0
* dotnet ef migrations remove

## Drop database and recreate

* dotnet ef database drop
* dotnet ef database update

## Seeding the Data

This will come later. Check [0013 WebApi - EF Relationships Conventions – Seed Data – Automapper – Automapper Queryable Extensions.docx](0013%20WebApi%20-%20EF%20Relationships%20Conventions%20–%20Seed%20Data%20–%20Automapper%20–%20Automapper%20Queryable%20Extensions.docx) for details.

# Viewing the Database

SQLite extension is already installed. Click CTRL+SHIFT+P to open sqlite explorer. Expand it under the Explorer to view

Text

Description automatically generated

# Add User dummy data using Sqlite explorer

Right click on Users in sqlite explorer and select “new query select”. Add following

INSERT INTO Users (Id, UserName) VALUES (1, "Bob");

INSERT INTO Users (Id, UserName) VALUES (2, "Tom");

INSERT INTO Users (Id, UserName) VALUES (3, "Jane");

Then highlight the line, right click and click Run Selected Query.

Right click Users and click Show Table. You should see the rows inserted.