Table of Contents

[App Basics 2](#_Toc106926789)

[.Net 6 Install 2](#_Toc106926790)

[VisualStudio Code Install 2](#_Toc106926791)

[nodeJs Install 2](#_Toc106926792)

[Install Postman 2](#_Toc106926793)

[Angular Install 2](#_Toc106926794)

[Making VS Code IDE Better 3](#_Toc106926795)

[WebApi Project 3](#_Toc106926796)

[Setting up .Net API Project 4](#_Toc106926797)

[Create WebAPI Method 1 4](#_Toc106926798)

[Create WebAPI Method 2 4](#_Toc106926799)

[Tidy up the Solution 4](#_Toc106926800)

[Actual Commands used for creating the api 4](#_Toc106926801)

[Folder Structure 4](#_Toc106926802)

[Running .Net API Project 4](#_Toc106926803)

[Switching to Classic hosting model 4](#_Toc106926804)

[Implicit Using Statement 5](#_Toc106926805)

[Program.cs 5](#_Toc106926806)

[WeatherForecast.cs 5](#_Toc106926807)

[/Controller/WeatherForecastController 5](#_Toc106926808)

[Nullable Enabled 5](#_Toc106926809)

[WebApi Folder Structure 6](#_Toc106926810)

[Angular Project 6](#_Toc106926811)

[Setting up Angular App 6](#_Toc106926812)

[Create the Anuglar app 6](#_Toc106926813)

[Add resources bootstrap and font-awsome 6](#_Toc106926814)

[Adding c# related extensions & packages in VS Code 6](#_Toc106926815)

[Angular Extensions 7](#_Toc106926816)

[Adding Packages 7](#_Toc106926817)

[App Features 7](#_Toc106926818)

[What we will be using? 7](#_Toc106926819)

[Entity Framework 7](#_Toc106926820)

[Important Documents 8](#_Toc106926821)

# Cloning Project

The cloned project will not run. For that first follow the “[0004 EntityFrameWork Setup Code First - DBContext - Sqlite.docx](0004%20EntityFrameWork%20Setup%20Code%20First%20-%20DBContext%20-%20Sqlite.docx)”, section “Migrations and Database Update” and issue the following two commands. In command prompt you must be in MSC.Api folder. This is where the project is.

* dotnet ef migrations add InitialCreate -o Core/DB/Migrations
* dotnet ef database update

# App Basics

|  |  |
| --- | --- |
| .Net 6 Install  1. Got to dotnet.microsoft.com/download 2. Click download 3. Select your OS Windows/Linux/macOS/Docker 4. Install SDK x64/x86  VisualStudio Code Install We’ll use VS Code as the code editor: <https://code.visualstudio.com/> | nodeJs Install  1. Go to <https://nodejs.org/en/> 2. Then other downloads 3. Select your OS 4. Either select the current version or select Node.js 16.13.0 from previous releases 5. Look into installing different versions of nodejs with NVM: <https://joachim8675309.medium.com/installing-node-js-with-nvm-4dc469c977d9> |
| Install Postman <https://www.postman.com/downloads/>  No need to create an account to use Postman. Towards the bottom there is skip link to skip login | Angular Install Installing it globally. Nodejs must be installed first   * npm uninstall -g @angular/cli * npm cache clean * npm install -g @angular/cli@13.0.2 |

|  |  |
| --- | --- |
| > dotnet --list-sdks | >dotnet --info |
| >node --version  v16.13.0 |
| >npm --version  8.1.3  **Alternate use nvm to install different versions of node**  <https://joachim8675309.medium.com/installing-node-js-with-nvm-4dc469c977d9> |
| >ng –version |

# Making VS Code IDE Better

1. AutoSave: Go to File and select “AutoSave” to automatically save our changes
2. Go to File > Preferences > Settings
   1. Type Font and change
      1. Main font size
      2. Scroll down and change for Console and Terminal as well
   2. Type exclude and add following to hide BIN and OBJ folders
      1. \*\*/bin
      2. \*\*/obj
   3. Type folders, go to Explorer:Compat Folder and unselect

# WebApi Project

Create a base folder to house the projects and files

## Setting up .Net API Project

|  |  |
| --- | --- |
| Create WebAPI Method 1 >dotnet new sln  Solution name the same as the container folder  > dotnet new sln --name MySolution  Solution name with custom name  >dotnet new webapi -o MSC.WebApi  Create a new project with name MCS.WebApi  >dotnet sln add MSC.WebApi  Add the project to the solution Create WebAPI Method 2 >dotnet new webapi -o MSC.WebApi -n MCS.WebApi Tidy up the Solution File > Preference > Settings and then type “Exclude”  Exclude Bin and obj folder by following the examples | Actual Commands used for creating the api Create a dir to house web api solution and project  >md MySocialConnect-API  cd into new dir  >cd MySocialConnect-API  Create a new solution  >dotnet new sln --name MSC-API  Create a new WebApi project  >dotnet new webapi -o MSC.Api  Add the project to the solution  >dotnet sln add MSC.Api Folder Structure |

## Running .Net API Project

|  |  |
| --- | --- |
| * Open command prompt and navigate to “[basePath]/MySocialConnect-API/MSC.Api” * Then execute “dotnet run” or “dotnet watch run” commands. * Once running successfully then pick the url from the command prompt | Text  Description automatically generated |

Then go to <https://localhost:7135/swagger> or <http://localhost:5157/swagger>

Swagger will display. Expand the GET method end point under WeatherForecast, click Try it out and then click Execute. You should see result here.

## Switching to Classic hosting model

1. Go to folder Documents/dotnet/ClassicHostingModel
2. There are two files, Program.cs and Startup.cs.
3. Put these in the MSC.Api folder. Startup.cs is a new file and Program.cs will get replace.
4. Also go to MSC.Api/Properties and open launchSettings.json file. Change the launchUrl. For this following this link [Tutorial: Create a web API with ASP.NET Core | Microsoft Docs](https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-6.0&tabs=visual-studio-code)

## Implicit Using Statement

|  |  |
| --- | --- |
| In Program.cs, WeatherForecast.cs and /Controller/WeatherForecastController there are using statement missing. This is due to a flag in MSC.Api.csproj file. Comment out the ImplicitUsings.  Now when you go to Program.cs, WeatherForecast.cs and /Controller/WeatherForecastControoler you’ll see a lot of errors since the using statement is missing. | Text  Description automatically generated |

Put the cursor on each error and then click CTRL+. to use the using statement.

|  |  |  |
| --- | --- | --- |
| Program.cs Following will get added   * using Microsoft.AspNetCore.Builder; * using Microsoft.Extensions.DependencyIn jection; * using Microsoft.Extensions.Hosting; | WeatherForecast.cs Following will get added   * using System; | /Controller/WeatherForecastController Following will get added   * using System; * using System.Collections.Generic; * using System.Linq; * using Microsoft.AspNetCore.Mvc; * using Microsoft.Extensions.Logging; |

## Nullable Enabled

|  |  |
| --- | --- |
| If you look at the WeatherForecast.cs, the stril property Summary is nullable. This will cause some issues for us so we will remove the ? from it and also comment out the nullable flag from MSC.Api.csproj file.  And then from WeatherForecast.cs remove ? |  |

## WebApi Folder Structure

|  |  |
| --- | --- |
| Other than the default structure, create a Core folder inside MSC.Api folder.  We’ll add al of our items, other than controllers to this core folder. However, the content will grouped together inside sub folders. For starter, create following sun folders and we’ll add as we go forward with this project   1. BusinessLogic: here we’ll add our business logic. Controller 🡺 BL 🡺 Repository 2. Constants: add any constant files 3. DB: add dbcontext and the actual DB etc 4. Extensions: house all the extensions here 5. Migrations: add entity framework migrations 6. Modals: add all the modals here 7. Repositories: house all the repository classes here |  |

# Angular Project

## Setting up Angular App

|  |  |
| --- | --- |
| Create the Anuglar app >ng new MySocialConnect-SPA  >ng serve to run Add resources bootstrap and font-awsome >npm install bootstrap font-awesome  go to angular.json, look at the Styles Array. it will be referencing "src/styles.css". Open this css file and import the bootstrap and font-awesome   * @import '../node\_modules/bootstrap/dist/css/bootstrap.min.css';   @import '../node\_modules/font-awesome/css/font-awesome.min.css'; |  |

# Adding c# related extensions & packages in VS Code

|  |  |
| --- | --- |
| 1. C# for Visual Studio Code (powered by OmniSharp) 2. C# Extensions by JosKreativ 3. Material Icon Theme by Philipp Kief 4. SQLite by alexcvzz | * After the reload you'll be shown some file missing popup - click yes. It will create .vscode folder. * If you miss this then do CTRL+SHIFT+P and type assets and click it to adding missing assets |
| **Adding nuget extension**   1. NuGet Gallery by pcislo 2. vscode-nuget-package-manager [use above] |  |
|  |  |

# Angular Extensions

|  |  |
| --- | --- |
| * Angular and then select Angular v7 Snippets by john papa * Angular Files 1.6.2 Alexander Ivanichev * Angular Language Service 0.1.10 by Angular * Angular2-switcher by infinity1207 * Auto Rename Tag 0.0.15 Jun Han | * Bracket Pair Colorizer CoenraadS * Debugger for Chrome Microsoft 4.11.0 * Material Icon Theme Philipp Kief 3.6.0 * Path Intellisense Christian Kohler 1.4.2 * Prettier - Code formatter Esben Petersen 1.6.1 * TSLint egamma 1.4.40 |

# Adding Packages

* For this NuGet extension must be installed
* Do CTRL+SHIFT+P 🡺 type nuget 🡺 Open NuGet Gallery
* Search for the following packages, select package, and tick the check box for the project where it will get installed. In my case it is MSC.Api.csproj
  + Microsoft.EntityFrameworkCore.Sqlite v6.0.6
  + Newtonsoft.Json v13.0.1
  + Microsoft.EntityFrameworkCore.Design v6.0.6 by Microsoft
* Go to nuget.org and install the following from there
  + dotnet -ef : It is a tool, pick the same version is the entity frame work installed above
    - dotnet tool install --global dotnet-ef --version 6.0.6

|  |  |
| --- | --- |
| App Features  1. Registration & Login with ASP.Net Identity 2. View list of members currently online 3. Like members and list 4. View members who liked them 5. Upload photo 6. Update member profile 7. Messaging (live chat) system to message member’s real time 8. Pagination 9. Caching | What we will be using?  * Entity Framework * HTML5 * Bootstrap * CSS * TypeScript * C# * Sqlite (DB) |

# Entity Framework

For development will use Sqlite.

Diagram

Description automatically generated

# Important Documents

The documents are in order in which the project was built

1. [Incorrect+MS+Build+selection+in+Omnisharp.pdf](dotnet/Incorrect+MS+Build+selection+in+Omnisharp.pdf)
2. [0002 VS Code DotNet Angular Commands.docx](0002%20VS%20Code%20DotNet%20Angular%20Commands.docx)
3. [0003 Working with Sample WeatherForecastController.docx](0003%20Working%20with%20Sample%20WeatherForecastController.docx)
4. [0004 EntityFrameWork Setup Code First - DBContext - Sqlite.docx](0004%20EntityFrameWork%20Setup%20Code%20First%20-%20DBContext%20-%20Sqlite.docx)
5. [0005 WebApi Controllers - Repository - Dependency Injection.docx](0005%20WebApi%20Controllers%20-%20Repository%20-%20Dependency%20Injection.docx)