C# Summery By Mahmoud Badawy

-----------------------------------------------------

ASPNET Core and Angular

الفيديوهات المستخدمه فى الشرح   
Udemy - Build an app with ASPNET Core and Angular from scratch

MB

##################################################################

اول حاجه كده هنروح نحمل .Net 5 عشان هنشتغل عليه و نعمل بيه مشروع الكور  
<https://dotnet.microsoft.com/en-us/download/dotnet/5.0>

و لو هتحدثه ل 6 بس خلى بالك الدوت نت 6 مش شغال الا على VS 2022   
<https://dotnet.microsoft.com/en-us/download/dotnet/6.0>

فعموما انا هشتغل على 6 عشان فيه فيتشارز جميله جدا و بالتالى لازم يكون عندى VS 2022

دى بقه الاختلافات الى حصلت فى الملفات الرئيسه فى .net 6 عن الفيرشنز الى قبلها ضرورى تبص عليهم

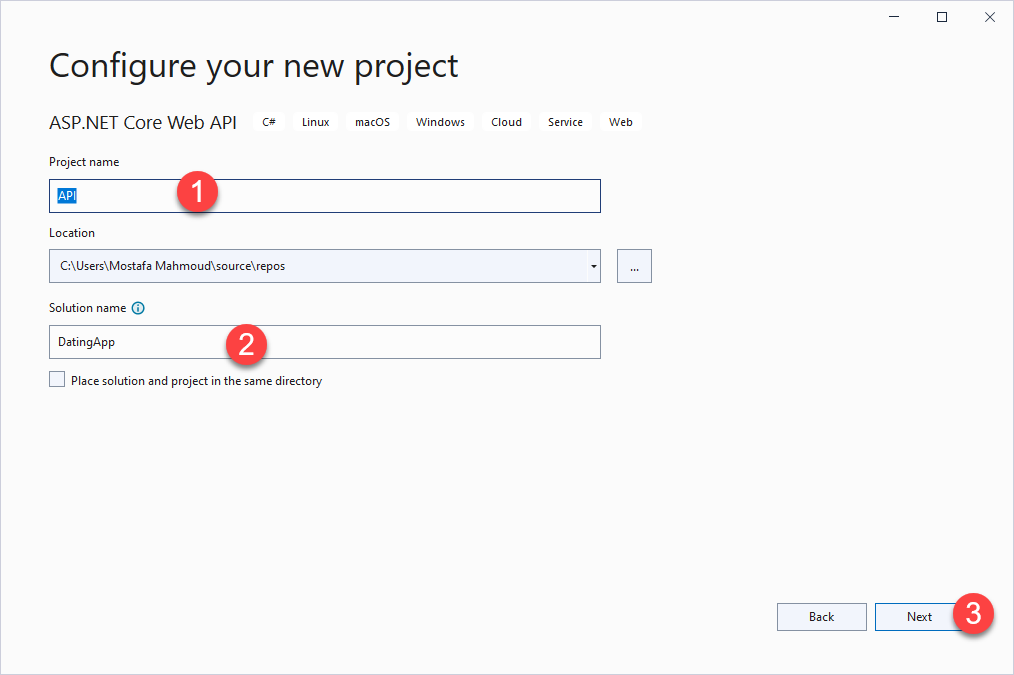
<https://docs.microsoft.com/en-us/aspnet/core/migration/50-to-60-samples?view=aspnetcore-6.0>

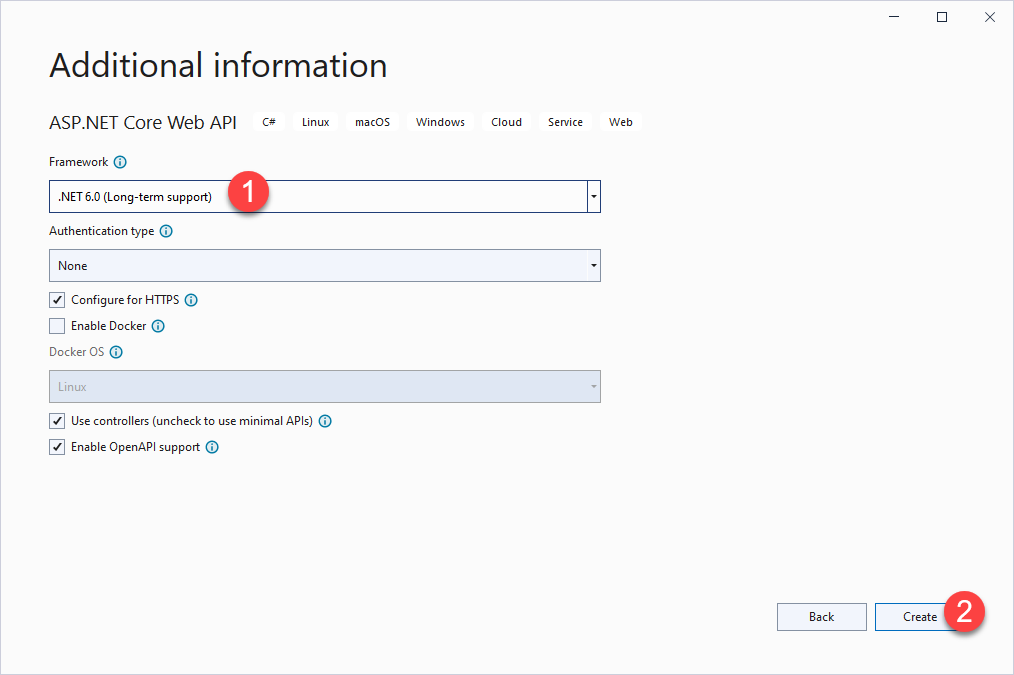
اللينك ده جميل جدا هيرويك استخدام سيرفيسييس و كونفيجريشنز كتير جدا و انت بتقرأ هيتفحلك ابواب كتير

كمان هنحمل ال NodeJs عشان هنستخدمه فى الانجيولار اكيد لل CLI

<https://nodejs.org/en/download/>

1. اول حاجه احنا هنعمل بروجيكت من نوع .net core web api و تعمل السوليوشن DatingApp و البروجيكت API
2. هنختار dot net 6



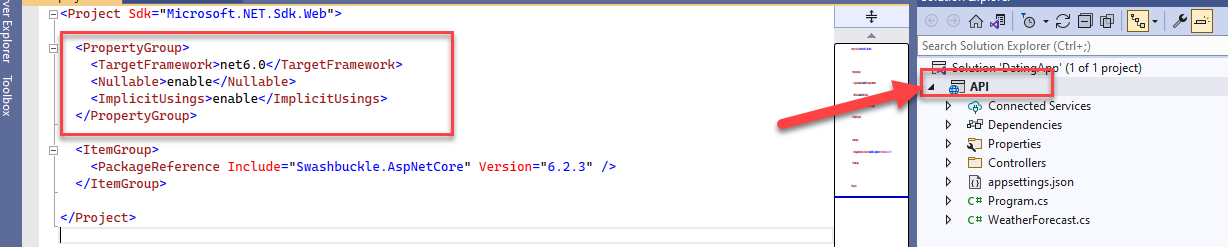


هيتفح البروجيكت يا سيدى و طبعا فى تغيرات رهيبه فى الفيرشن الحديده للدوت نت و اهمها الاتنين دول

<Nullable>enable</Nullable>

<ImplicitUsings>enable</ImplicitUsings>

و دول هتلافيهم موجودين فى ملف ال CsProj و هيبان لما تدوس على اسم البروجيكت الى انت بتشتغل عليه دلوقتى ذى ما فى الصوره بالظبط



<ImplicitUsings>enable</ImplicitUsings>

دى بقه ايه لازمتها ؟؟

دى يا سيدى بتخلى البروجيكت يعمل فايل Shared يبقى كل المشروع شايفه و بيبقى فيه كل المكتبات الى محتاجها فى كل المشروع بشكل اكيد و بالتالى بدل ما اقعد اكتبها كل مره فى الكلاسات الجديده فلا هما كلهم موجودين فى مكان واحده اسمه

DatingApp.GlobalUsings.g.cs

و بالتالى وفر علينا نعمل ايمورتات كتير جدا فى البروجيكت

و طبعا بما ان الفايل ده Autogenerated فمش هينفع نعدل فيه بس ميكروسوفت اتاحت اننا نضيف النيم سبيسيس الى عاوزينها برحتنا عن طريق اننا فى ال Csproj برضوا نقوله استخدم الايتم دى او امسح الايتم دى و ادى مثال

<ItemGroup>

<Using Remove="System.Threading" />

<Using Include="Microsoft.Extensions.Logging" />

</ItemGroup>

ده لينك مفيد جدا جدا فى الجزئيه دى لو عاوز تفهم بتعمق شويه

<https://dotnetcoretutorials.com/2021/08/31/implicit-using-statements-in-net-6/>

Nullable دى لو عملتها Enabled فكده انت مجبر انك و انت بتعمل البروبيرتس فى الكلاسيس انك تهندل ال null مع الاسترينج فكده هيبقى فى تحذير دايم ان الاسترينج ممكن تيجى ب null و بالتالى هتعمل حسابك عليها و مش هيحصلك مشاكل فى الكود بسبب استرينج رايح يتسجل فاضى او جاى فاضى

MB

##################################################################

Controller

-----------------------------------------------------

اول حاجه فى البروجيكت بتبقى موجوده على التجربه هى ال Controller و ده هنفصصه بقه عشان تفهمه

using Microsoft.AspNetCore.Mvc;

namespace API.Controllers

{

[ApiController]

[Route("[controller]")]

public class WeatherForecastController : ControllerBase

{

private static readonly string[] Summaries = new[]

{

"Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering", "Scorching"

};

private readonly ILogger<WeatherForecastController> \_logger;

public WeatherForecastController(ILogger<WeatherForecastController> logger)

{

\_logger = logger;

}

[HttpGet(Name = "GetWeatherForecast")]

public IEnumerable<WeatherForecast> Get()

{

return Enumerable.Range(1, 5).Select(index => new WeatherForecast

{

Date = DateTime.Now.AddDays(index),

TemperatureC = Random.Shared.Next(-20, 55),

Summary = Summaries[Random.Shared.Next(Summaries.Length)]

})

.ToArray();

}

}

}

using Microsoft.AspNetCore.Mvc;

دى بقه النيم سبيس الى بيشتغل بيها اى حاجه تخص ال controller ذى ال

[ApiController]

و ده معناه ان الكلاس الى الاتريبيوت ده محطوط فوقه هو عباره عن كنترولار هيتحط فيه Api

[Route("[controller]")]

و ده يا سيدى الروت الى هيبقى واخده و طبعا [controller]

دى يا سيدى معناها ان اسم الكنترولار هو الى هيبقى نفس اسم الروت فهيبقى عشان اكلم اى api فى الكنترولار ده هيبقى اللينك الاساسى بتاعه كده

<https://localhost:7243/WeatherForecast>

و ذى ما انت شايف

WeatherForecast

ده اسم الكنترولار و برضوا هو الاسم الاساسى فى الروت الى بيكلم الكنترولار ده

private static readonly string[] Summaries = new[]

{

"Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering", "Scorching"

};

اما الجزئيه دى فهنا عامل اراى حاطط فيه كل توقعات الجو عشان اكيد هستخدمها تحت

private readonly ILogger<WeatherForecastController> \_logger;

public WeatherForecastController(ILogger<WeatherForecastController> logger)

{

\_logger = logger;

}

اما دى فياسيدى هنا اتعمل Dependency Injection للوجر الى بديله اسم الكنترولار الى هيشتغل عليه كجينيرك و اللوجر طبعا بيتعامل مع البروجيكت انه يطلعلك المشاكل و يحفظها فى فايل او داتا بيز او كونسول او اى حاجه انت عاوزه بعد كده يسجل عليها و بيسجل ال errors , info , Warnning اى حاجه ده بيبقى ذى الى بيراقب الى بيحصل فى البروجيكت عشان دايما يبقى معاك لوج تقدر ترجعله يسهل عليك انك توصل للمشاكل بسرعه

[HttpGet(Name = "GetWeatherForecast")]

دى بقه معناها ان الميثود الى هيتحط عليها الاتربيوت ده هتبقى من نوع Get و انا ببعت الريكويست ليها فى ال Api

و الروت بتعها فاضى يعنى مش هزود حاجه فى اللينك و مديها اسم مستعار هنا GetWeatherForecast

**curl -X 'GET' \**

**'https://localhost:7243/WeatherForecast' \**

**-H 'accept: text/plain'**

اما لو حبيت اديلها روت و ساعتها هحتاج ابعته فى اللينك هتبقى كده

[HttpGet]

[Route("GetWeatherForecast")]

**curl -X 'GET' \**

**'https://localhost:7243/WeatherForecast/GetWeatherForecast' \**

**-H 'accept: text/plain'**

اوعى تعملهم كده هيطلع فى وشك ايرور جميل

[HttpGet(Name = "GetWeatherForecast")]

[Route("GetWeatherForecast")]

لازم طالما هتحط اتربيوت الروت يبقى تشيل ال Name من ال HttpGet

MB

##################################################################

Entity

------------------------

بص يا سيدى ال Entity ده يا سيدى هو عباره عن الكلاسات الى بتعمل mapping مع الداتا بيز عشان اكلم بيها الداتا بيز لما اكون عاوز ابعت او استقبل حاجه من الداتا بيز بشكل مباشر

غالبا بنعمل بروجيكت لوحده فيه الكلاسات دى كنوع من التنظيم بس احنا هنا بنبى حاجه بسيطه دلوقتى فهنحطها فى نفس المشروع بس فى فولدر لوحده

namespace API.Entities

{

public class AppUser

{

//prop then Tab or PropFull then Tab

public int Id { get; set; }

public string UserName { get; set; }

}

}

و ادى اول كلاس باسم AppUser و ده لان اول table فى الداتا بيز هيبقى هو جدول مستخدمى الابلكيشن عشان كده تم تسميته AppUser و اتسمى فردى عشان الانتى تى بيعبر عن Row فى الجدول مش كوليكشن فاهمنى يا صديقى انما فى الداتا بيز بيتكتب Users عشان الجدول بيبقى شايل ملايين الروز دى حاجه استاندارد مش اكتر

خلصنا عملنا اول كلاس فى ال Entity

تعالى بقه نشوف هنكلم الداتا بيز اذاى بقه عن طريق ال EntityFrame Work

بص يا سيدى ال EntityframeWork دى هى عباره عن ORM = Object Relation Mapper

بمعنى ؟

بص يا سيدى احنا عندنا داتا بيز عاوزين نكلمها طب هنكلمها اذاى المفروض هعمل كلاس بيماب الجدول الى هكلمه فى الداتا بيز بالظبط و اروح افتح كونيكشن و اعمل ريد للداتا بيز على حسب البروفايدر و لما اردي اقفل الكونيكشن حوارات كنا بنعمله فى الكلاس الواحد و لو حبينا فى يوم نغيير بروفايدر الداتا بيز يبقى الحياه ساعتها قلبت جحيم

فجه المعلم

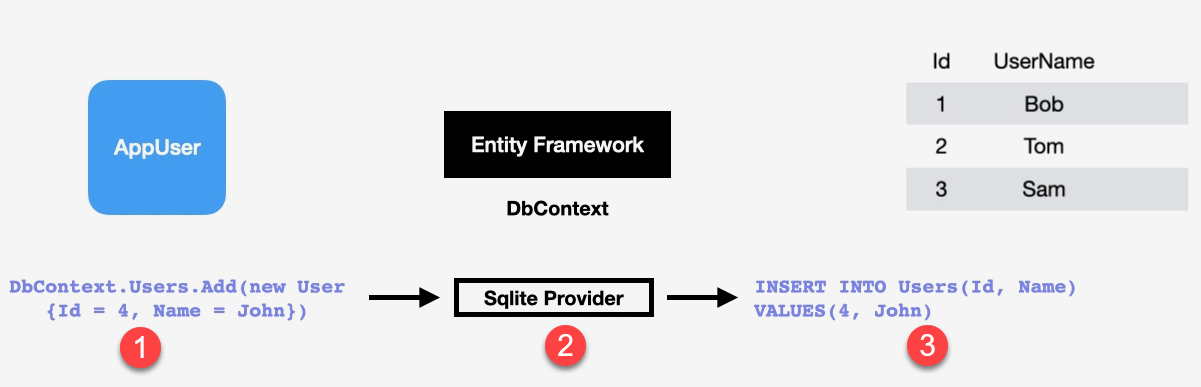
EntityFrameWork

قالك لأ انا هبقى وسيط بين الداتا بيز و بين كلاسات ال Entity الى هتكلمه بحيث تعملوا فيا كلاس اسمه DbContext واحد معفن

فيه بتحطوا الكونيكشن سترينج مره واخده و نوع البروفايدر مره واحده و ال DbSets الى هتعملوها و انا هقدر اوصل بين الداتا بيز و الانتى تى كلاسيس و اودى الداتا و ارجعها بحيث بعد كده لو حبيتوا تغيروا اى حاجه فى الحاجات دى يبقى تغييره فى مكان واحده

و كمان ما يميزه بقه انه بياخد كود Linq يحوله الى Sql كومانديس او Oracle Commands او اى كوماندس على حسب البروفايدر الى انت بعتهوله بحيث لما يكلم الداتا بيز مهما كان نوعها هى تفهمه و متغيرش كل شويه فى الابليكيشن

و دى صوره بتوضح الى بيحصل



صحيح احنا هنستخدم فى المشروع ده SQLite عشان SQL بيشتغل على الويندوز بس بينما Sqlite بيشتغل فى اى حته Cross plateform فهيبقى ملائم اكتر ل Core

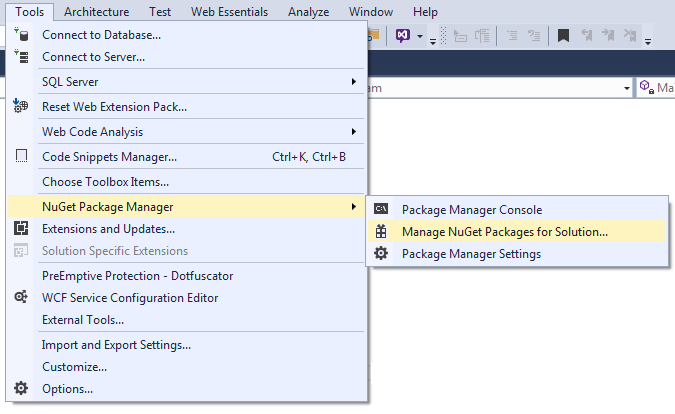
طيب اذاى بقه اسطب EntityFrameWork الخاص بكور على المشروع

هسطب الباكيدج دى الى بتتعامل مع الانتى فريم ورك كور و كذلك التول الخاصه بالديزاين و كذلك التول الخاصه بال Sqlite provider

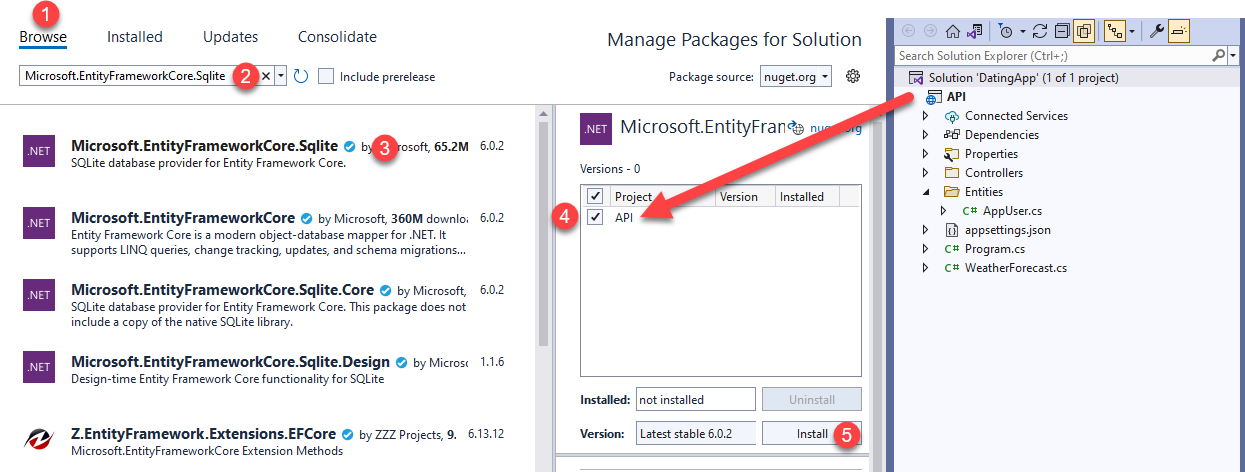
طيب انا ممكن اوفر على نفسى كل ده و اسطب بالسطر ده فى ال nuget Console السكوال لايت انتى تى فريم ورك و هو هيسطب كل الدبندينسيس الى محتاجها

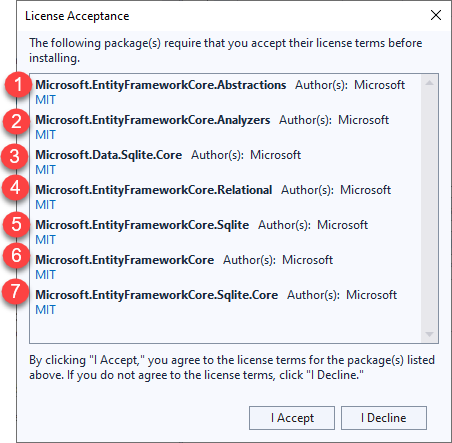
**dotnet add package Microsoft.EntityFrameworkCore.Sqlite --version 6.0.2**

او انى اعمل ذى الى فى الصوره و اسطبه من nuget Manager



ذى ما انت شايف انا عاوز اسطبه فى مشروع ال API فاخترت المشروع الى هسطبه فيه عشان هو ده المكان الى هستخدم فيه EntityFrameWork بشكل مباشر لو هستخدمه طبعا فى اى بروجيكت تانى هروح اسطبه تانى هناك

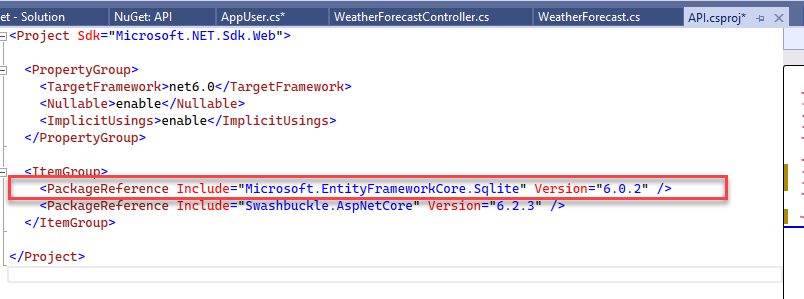


و ذى ما انت شايف اول ما قولت هسطب الباكيدج الخاصه بالبروفايدر SQLite بتاعت ال EntityFrameWork

راح رايح جايب كل الباكيدجات الى محتاجها كلها و هما حوالى 7 ذى ما انت شايف اهه و بالتالى مش هحتاج اسطب حاجه تانى قبله او بعده هههههه

لو حبيت اسطب بتاع السكوال كل التغيير الى هيحصل انك هتكتب SQL بدل Sqlite

لو روحنا دلوقتى ل API.csproj هتلاقى الباكيدج شرفت هناك يا صديقى و دى بتتعامل عشان لما تيجى تلود المشروع لو فى باكيدج مش موجوده المشروع يسطبه من نفسه و طبعا عشان تحفظ و تعرف اصدارات الحاجات الى انت بتسطبها



اهم حاجه بقه اوعى تنسى تسطب ال Designer & tool

**dotnet add package Microsoft.EntityFrameworkCore.Design**

ده يا سيدى الى بيساعد ان الكود يبقى شايف الداتا بيز و يقدر يعمل ماب معاها و يعمل الدنيا

Code first

Database first

اما tool ده بقه الخاص بالميجريشن ذات نفسه

**dotnet add package Microsoft.EntityFrameworkCore.Tools**

و من غيره الميجريشن مش هيرضى يشتغل يا سيدى

Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio.

Enables these commonly used commands:

Add-Migration

Bundle-Migration

Drop-Database

Get-DbContext

Get-Migration

Optimize-DbContext

Remove-Migration

Scaffold-DbContext

Script-Migration

Update-Database

و عشان اتعامل مع الميجريشن بسهوله جدا فى الكوماندس فهنزل الباكيدج دى لو الى فوق مش عجبانى بس خلى بالك عشان حاولت اسطب البتاعه دى من الناجيت مانيجر و طلعت عينى فانت فى الكونسول حط الامر ده

|  |
| --- |
| **dotnet tool install --global dotnet-ef --version** [**6.0.2**](https://www.nuget.org/packages/dotnet-ef/6.0.2)  و لازم تجيب رساله السكيس دى  Tool 'dotnet-ef' (version 6.0.2') was successfully installed.  دى بقه لو عاوز تشيل التسطيب  **dotnet tool uninstall --global dotnet-ef** |

**Top of Form**

**Bottom of Form**

**Top of Form**

**Bottom of Form**

Entity Framework Core Tools for the .NET Command-Line Interface.

Enables these commonly used dotnet-ef commands:

dotnet ef migrations add

dotnet ef migrations list

dotnet ef migrations script

dotnet ef dbcontext info

dotnet ef dbcontext scaffold

dotnet ef database drop

dotnet ef database update

مقال يفهمك ايه لازمه الباكيدجات الى عمالين نسطب فيها دى و زياده كمان

<https://cementanswers.com/what-is-microsoft-entityframeworkcore-design/>

MB

##################################################################

DbContext

------------------------

كلاس مهم جدا فى ال EntityFrameWork مهمته انه يبقى حلقه الوصل بين الداتا بيز و الكود و فى الغالب بنحط فيه DbSets الى بتعبر عن Tables فى الداتا بيز من خلال كلاسات ال entity

اذاى بعمله فى المشروع بقه ؟؟ ده لينك تقرأ فيه عنها جميل جدا

<https://www.learnentityframeworkcore.com/dbcontext>

و عموما الموقع ده يعتبر لامم كل حاجه تخص ال entityFrameWark و اذاى تستخدمها فى مشروعك هيوفر عليك كتير جدا جدا

<https://www.learnentityframeworkcore.com/efcore/how-to-get>

<https://www.entityframeworktutorial.net/efcore/entity-framework-core.aspx>

بس تعالى نشرح برضوا و نفهم مع بعض الاساسيات الخاصه بال DbContext و اذاى نكريتها و كده

بص يا سيدى جوه البروجيكت هنعمل فولدر اسمه Data و جواه هنعمل كلاس نسميه DataContext هيكون بيورث من الكلاس الاساسى dbContext

و جواه بقه هنعمل الكود الى هوريهولك بقه عشان يشتغل و هفصصهولك حاجه حاجه عشان تعرف منين بيودى لفين

فاول حاجه اول ما هكريته هيبقى بالشكل ده

using Microsoft.EntityFrameworkCore;

namespace API.Data

{

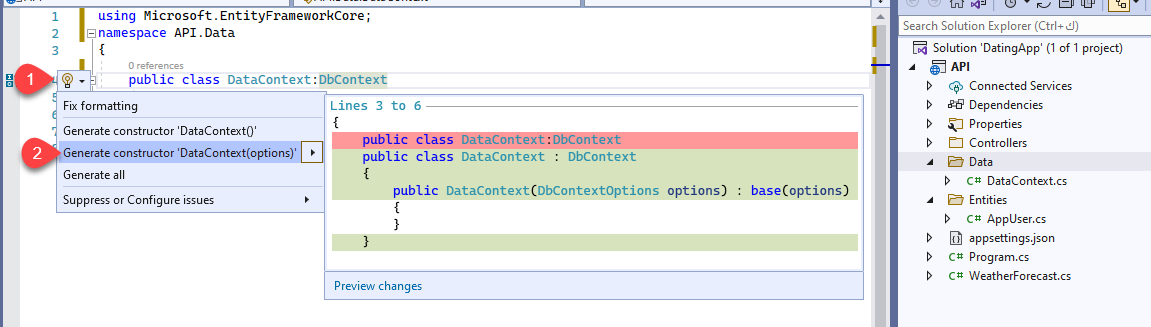
public class DataContext:DbContext

{

}

}

و طبعا عاوزين الكونستراكتور فهنعمله AutoGenerate عشان منتعبش فيه مش فاضيين احنا



فكده عملنا الكونستراكتور و عملنا حقن لل DbContextOptions عشان اقدر اتحكم فى كل اوبشنز الكونتيكست بسهوله لو احتجت اغير حاجه فيهم

ثم بعد كده هروح اكتب ال DbSets الخاصه بكل جدول هتعامل معاه جوه الكونتيكست دى

فتعالى نكتب الكود كامل بقه

using API.Entities;

using Microsoft.EntityFrameworkCore;

namespace API.Data

{

public class DataContext : DbContext

{

public DataContext(DbContextOptions options) : base(options)

{

}

public DbSet<AppUser> Users { get; set; }

}

}

كده خلاص عملنا الكونتيكست و عرفنا الحدول عن طريق بروبرتى من نوع DbSet اوف AppUser

كده الى باقى اننا نروح فى الكلاس الابتدائى بتاع البرويجكت و نشغل ال dbContext ضمن ال services الى المفروض المشروع يشغلها فهنروح للكلاس program.cs و هنكتب جوه الكود ده الى هيبقى فيه اسم ال dbContext و ال ConnectionString

// Add services to the container.

**builder.Services.AddDbContext<DataContext>(options =>**

**{**

**options.UseSqlite("Connection String");**

**});**

كده محتاجين بقه اننا نروح نظبط الكونيكشن سترينج ياباشا فى فايل AppSettings.cs الى هو يعتبر ال webConfig فى شكله الجديد و هنروح نكتب جواها بقه الكونيكشن استرينج بالشكل الجميل ده

<https://www.connectionstrings.com/sqlite/>

{

**"ConnectionStrings": {**

**"DefaultConnection": "Data Source=datingapp.db"**

**},**

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

}

طيب بعد كده عاوز اروح اقرأ الكونيكشن سترينج فى السيرفيس الى فى فايل program.cs بالمفروض عشان اقدر اوصل للسيكشن دى فى الكونفيجريشن انه اتعامل مع الكلاس الى بيماب مع ال appsettings و اجيبه بيه

فتبقى كده

string connString = builder.Configuration.GetConnectionString("DefaultConnection");

builder.Services.AddDbContext<DataContext>(options =>

{

options.UseSqlite(connString);

});

ال .Net 6 فى الفايل ده اتغير تماما ههههههههه مبقاش فيه انجيكت للكونفيجريشن ولا بتنجان وفروا علينا وقت جاى معمول جاهز على الاستخدام على طول

على عكس زمان كان بيتكتب كل الهبد ده من اول انك بتعمل دبندينسى انجيكشن و بعد كده تروح تستخدم الحاجه و بتاع كان مللل بصراحه بص القديم عشان تعرف التطور خلاك تكتب سطر واحد كتيان

**In .net 5 Startup.cs**

public class Startup

{

private IConfiguration Configuration { get; }

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public void ConfigureServices(IServiceCollection services)

{

// some stuff

var app = Configuration.GetConfigSection<AppConfigSection>();

var connectionStrings = Configuration.GetConfigSection<ConnectionStringsConfigSection>();

services.AddDbContext<AppDbContext>(options =>

{

if (app.IsLocal)

{

options.UseSqlite(connectionStrings.LocalConnectionString);

}

else

{

options.UseSqlServer(connectionStrings.ServerConnectionString);

}

});

// other stuff

}

}

كده يا سيدى مش باقى غير اننا نعمل الداتا بيز و نعمل المايجريشن بقه

و بما اننا رايحين نستخدم الميجريشن بقه و الهيصه دى فاوعى تنسى ان الكلام ده بيحتاج ال

**dotnet add package Microsoft.EntityFrameworkCore.Design**

**dotnet add package Microsoft.EntityFrameworkCore.Tools**

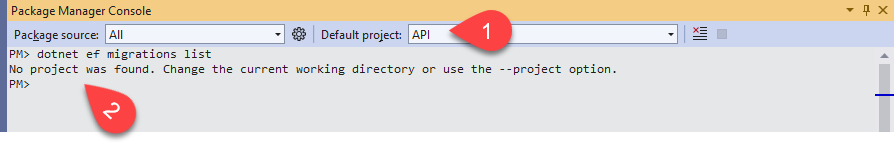
**dotnet tool install --global dotnet-ef --version** [**6.0.2**](https://www.nuget.org/packages/dotnet-ef/6.0.2)

فاحنا دلوقتى هنجهز الميجريشن بتاع ال CodeFirst قبل ما نعمل الداتا بيز يلا بينا

او لحاجه هنتاكد ان الميجريشن شغال فهنقوله الامر ده على البروجيكت الى انت مسطب عليه ال tool الى هى فى حالتنا بروجيكت ال API

dotnet ef migrations list

فهترد عليك و تقولك مفيش ميجريشت لسه اتعملت يا معفور على المشروع ده



ساعتها بقه اعمل migration يا معلم بس الاول عشان تعرف الكوماندس بتتكتب اذاى للميجريشن هتكتب الامر بتاع الهيلب

dotnet ef migrations –help

<https://docs.microsoft.com/en-us/ef/core/cli/dotnet>

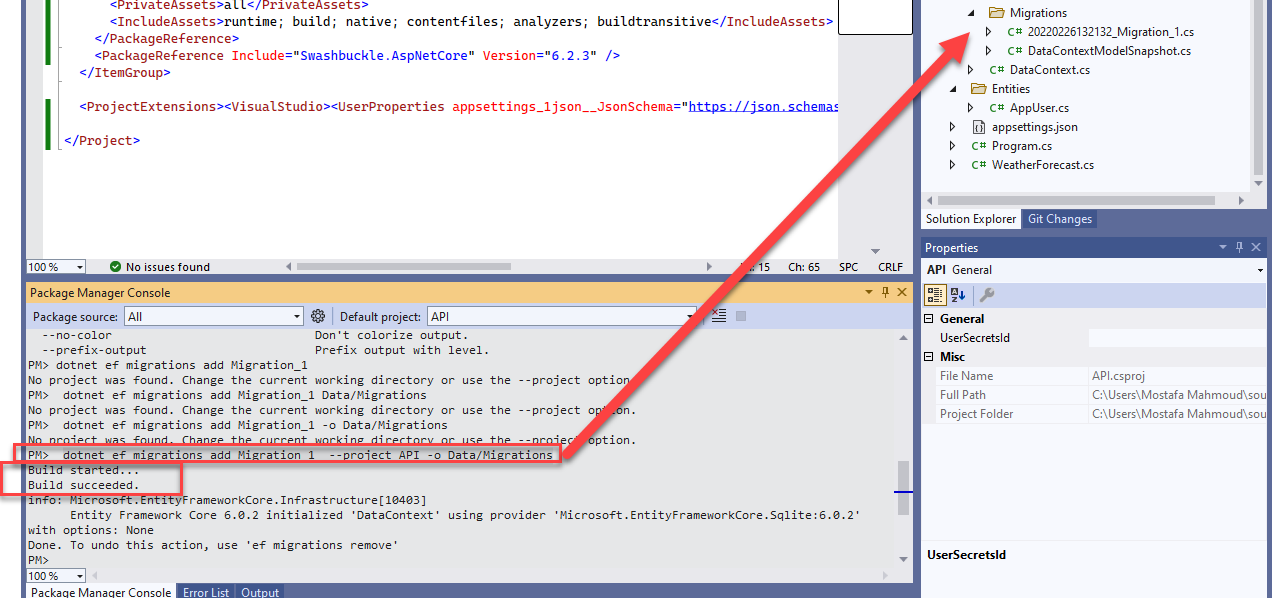
ده لينك تفهم منه كل الاوامر الخاصه بالميجريشن

ثم منه هتفهم تكتب امر اضافه الميجريشن اذاى و بالتالى هتكتبه كده

dotnet ef migrations add Migration\_1 --project API -o Data/Migrations

ده يا سيدى بقوله اعمل ميجريشن جديده فى بروجيكت ال API فى فولدر ال Data/Migrations

و فعلا عمله



تعالى بقه نفهم البتاع ده عمل ايه ؟؟

بص يا سيدى البتاع ده اول ما انت اديته الامر راح رايح للبروجيكت API بص لقى عنده كلاس فيه ال Dbcontext و عرفه و عرف هيكلم داتا بيز من انهى نوع و كل حاجه و كل ده طبعا من الكونفيجريشن الى انت عامله فى ال program.cs فراح رايحله على طول جواه لقى جواه Dbset فعرف ان الكونتيكست ده جواه Entity و بالتالى راح داخل جوه الانتى تى ده و شاف هو مكون من ايه بالظبط و راح عامل كود الميجريشن Up و Down بحيث Up يكون فيه انشاء الجداول الجديده او احدث بعض التغيرات لو كانت دى تانى ميجريشن و بتعدل حاجه و هو بيحطش الا التغييرات الى هو شايفها حاليا بس و بالنسبه بقه لل down فدى فيها reverse كود بمعنى انه بيبقى عامل كود عكسى يمسح الحاجه الى اتعملت دى فى حاله انك رجعت فى كلامك

و ادى الكود عشان تفهم برضوا

using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace API.Data.Migrations

{

public partial class Migration\_1 : Migration

{

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.CreateTable(

name: "Users",

columns: table => new

{

Id = table.Column<int>(type: "INTEGER", nullable: false)

.Annotation("Sqlite:Autoincrement", true),

UserName = table.Column<string>(type: "TEXT", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Users", x => x.Id);

});

}

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropTable(

name: "Users");

}

}

}

خدت بالك ان الميجريشن البرنس عمل ال Id برايمرى كى و اوتو انكريمنتال من غير ما تقوله حتى هههههههههه ده لانه اول ما بيشوفك كاتب Id كده هو بيفهم لوحده من غير ما تقوله اصلا

بس انا بفضل انى اكتب [key] و انا بعمل الكلاس بصراحه فوق ال Id

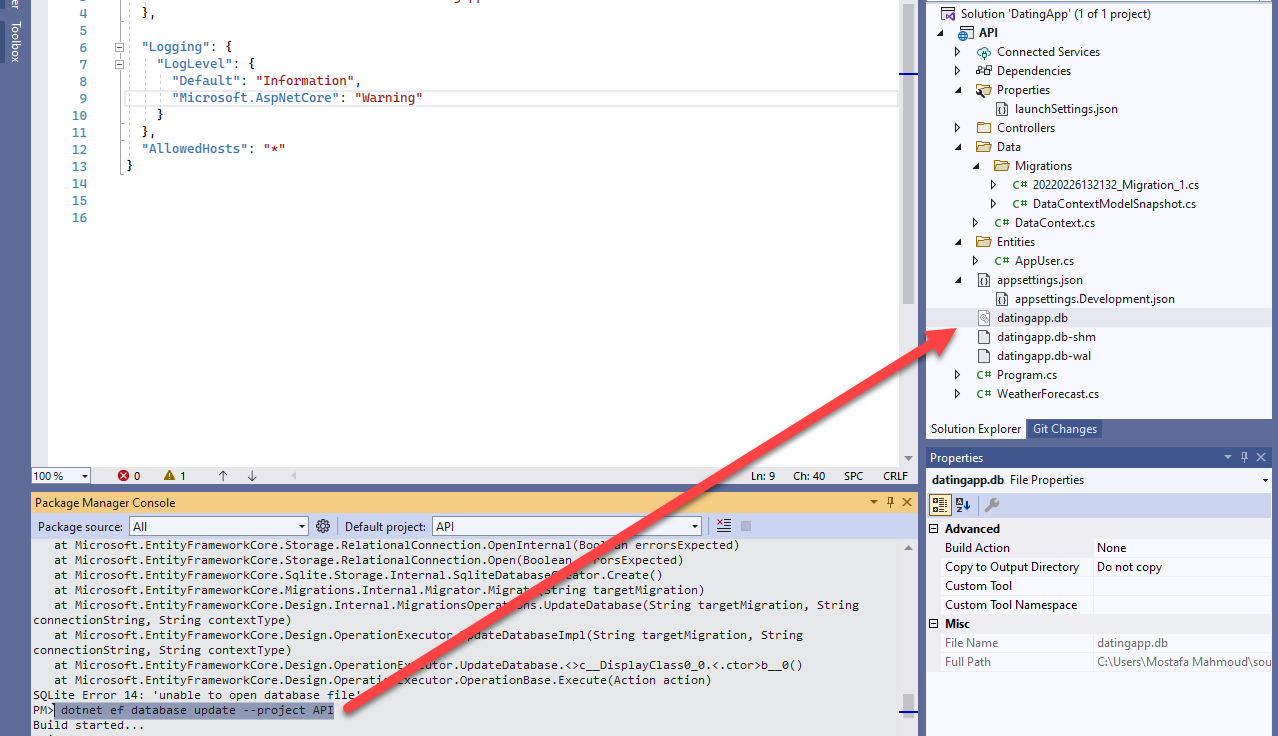
عشان نعمل الداتا بيز بقه اول حاجه روح سطب Sqlite Explorer بعد ما تحمله من هنا

<https://sqlitebrowser.org/dl/>

بعد كده اعمل update للداتا بيز بالميجريشن كوماند ده بقه

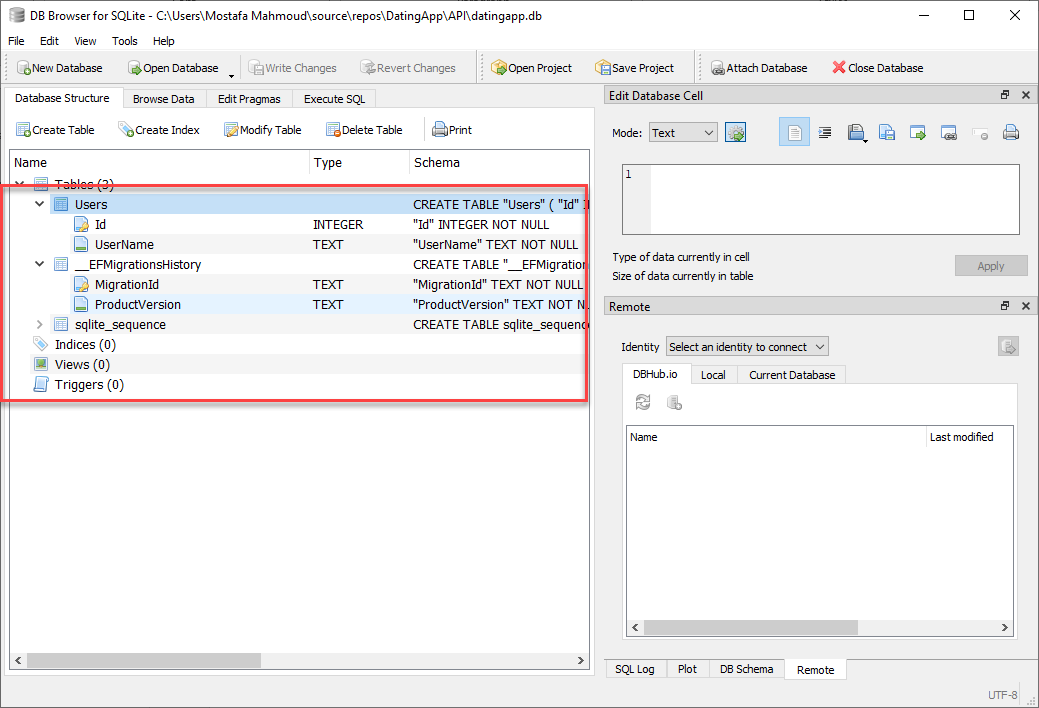
dotnet ef database update --project API

اول ما هترن الكود ده هتلاقى فايل الداتا بيز اتكريت داخل البروجيكت بتاع ال API



هتروح بقه تشوف الكلام ده ال Sqllite Explorer تشوف التيبول اتكريت فعلا و كل حاجه تمام ولا لأ ؟

هتلاقيه متكريت و فل الفل ذى ما انت شايف كده



كده انت ربطت الداتا بيز بالميجريشن و بالكود و كل حاجه ذى الفل

نجرب بقه نضيف اى داتا للجدول بقه و نشوف هيحصل ايه و الدنيا هتمشى ولا لأ

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

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

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

طبعا الحاجات دى بترن فى ال execute SQL و بتشوف الداتا الى انت ضيفتها ى فى ال Browse Data

<https://stackoverflow.com/questions/11646047/error-cannot-access-file-bin-debug-because-it-is-being-used-by-another-proc>

حل مشكله ظهرتلى فى النص

و كده يا سيدى يبقى احنا كده بدانا نكلم داتا بيز هنشتغل بقه اننا نعمل Controller جديد نكلم فيه الداتا بيز بقه يا معلمى

MB

##################################################################

First Controller That Call DB

-----------------------------------

بص يا سيدى هنروح لفولدر الكنترولارز و نعمل فيه كونترولر جديد نوعه API Controller Empty

و هيبقى الكود بالشكل ده

namespace API.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class UsersController : ControllerBase

{

private readonly DataContext \_dbContext;

public UsersController( DataContext dbContext )

{

\_dbContext = dbContext;

}

[HttpGet]

[Route("GetAllUsers")]

public ActionResult<List<AppUser>> GetAllUsers()

{

var Users = \_dbContext.Users.ToList();

return Users;

}

[HttpGet]

[Route("GetUsers/{Id}")]

public ActionResult<AppUser> GetUser(int Id)

{

var User = \_dbContext.Users.FirstOrDefault(\_=>\_.Id == Id);

return User;

}

}

}

اولا احنا بنورث من الكنترولاؤ بيز ControllerBase عشان اكتسب خواص الكنترولار كلها

تانى حاجه بعمل كونستراكتور و بعمل dependenct Injection فيه لكلاس الكونتكست علشان اقدر اتعامل مع الداتا بيز و طبعا فى طريقه كود افضل من ده بمراحل بس حاليا خلينا فى الفهم الاول و بعد كده نبقى ننضف الاستراكتشار برحتنا

تالت حاجه بعمل ال methods الى هكلم بيها الداتا بيز عن طريق ال HttpRequest

فى الفاانكشن الاولى GetAllUsers بقوله كلم الكونتكست و هاتلى لسته المسخدمين الى فى جدول ال Users

و عشان كده برجعه فى ActionResult<List<AppUser>>

اما الفانكشن التانيه GetUser بقوله انا عاوز يوزر واحد على حسب الاى دى بتاعه الى هبعتهولك فى الروت فبترجع ساعتها يوزر واحده فى ActionResult<AppUser>

ال ActionResult كلاس دى هى اخر تحديث لل HttpResponse و دى يعنى النتيجه الى برجعها فى ال Http فانت لو فاهم يعنى ايه HttpResponse هتفهم قصدى ايه هنا

اول ما هتجرب الفانكشنز و تلقيها بترجع داتا يبقى كده خلاص عنب اوى

لو مرجعتش داتا يبقى اعمل save DB اتاكد ان البيانات اتحفظت فى فايل الداتا بيز يا حج بس او لو قالك فى مشكل فى الكونيكشن حلها بقه ساعتها بالنسبه للسكوال لايت طالما الفايل ده اتعمل يبقى البيانات اتسجلت datingapp.sqbpro

MB

##################################################################

Async Controller Methods

-----------------------------------

ليه بقه احول الفانكشنز الى جوه الكنترولار دى لفانكشنز نوعها Async ؟

بص ياسيدى لو انا عندى ملايين المستخدمين و بعمل كذا فانكشن ورا بعض ايه الى هيحصل فى الحاله دى ؟؟

الى هيحصل ان السى شارب هتشتغل sync فهتفتح كذا thread تروح تجيب شويه داتا فى الفانكشن الاولانيه و بعد كده تقفل البلوك ده شويه و تروح تشتشغل فى الفانكشن التانيه ب thread اخرى و ترجع تكمل الاولى و هيصه كده سعدتك ممكن لو الداتا كبيره فى كذا فانكشن مع بعض و كذا مستخدم الدنيا تنهار بسبب الريكويستات دى

طب ايه حل المشكله ؟؟

اننا نعمل الكود async بحيث يبقى ماشى بمبدا انا همشى فى الميثود الاولى لما اخلصها كلها و ارجع بياناتها و بعد كده انقل على التانيه اخلصها كلها و ارجع بياناتها و بعد كده التالته و هكذل لحد ما اخلص كل الشغل الى عليا ورا بعضه بالترتيب فيحمينا من الاخطاء و يسرع الشغل برضوا

اذاى بقه احول الفانكشنز الى جوه الكنترولار دى لفانكشنز نوعها Async ؟

كل الى هعمله انى هكتب async بعد الاكسيس موديفاير ثم اعمل Task و جوه الكود اثناء الريتيرن بتاع الداتا هقوله قبله Await هيبقى الكود كده فى الاخر بعد التعديل

using API.Data;

using API.Entities;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

namespace API.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class UsersController : ControllerBase

{

private readonly DataContext \_dbContext;

public UsersController( DataContext dbContext )

{

\_dbContext = dbContext;

}

[HttpGet]

[Route("GetAllUsers")]

public async Task< ActionResult<List<AppUser>>> GetAllUsers()

{

var Users = await \_dbContext.Users.ToListAsync();

return Users;

}

[HttpGet]

[Route("GetUsers/{Id}")]

public async Task<ActionResult<AppUser>> GetUser(int Id)

{

var User = await \_dbContext.Users.FirstOrDefaultAsync(\_=>\_.Id == Id);

return User;

}

}

}

نصيحه فى كور اشتغل كل شغلك Async افضلك

بعد الكام سيكشن دول هرفع بقه الباك الى معمول ده على Git و ده عن طريق visual Studio 2022 مسهل الموضوع جدا كل الى بتعمله بتكريت ريبو و بتعمل عليه كوميت و بوش و بول و حاجه اخر هبل و سهوله

MB

##################################################################

Auditing / tracking Changes in EF

---------------------------

<https://docs.microsoft.com/en-us/dotnet/api/microsoft.entityframeworkcore.changetracking.changetracker?view=efcore-6.0>

MB

##################################################################

BaseApiController

---------------------------

بص ياعم ده كونترول هحط فيه كل الى انا الخصائص و الفانكشنز و الاتربيوتس الى انا بحتاجها فى كل الكترولارز الى موجوده فى ال ControllerBase عن طريق الوراثه منه و زياده عليها الحاجات الى هحطهاله من زياده من عندى فيبقى لامم الدنيا اكتر بص يا سيدى الكود اهه و هفهمك الدنيا و احنا بنكتبه برضوا

BaseApiController.cs

namespace API.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class BaseApiController : ControllerBase

{

}

}

بس كده يا سيدى انا عملته اهه و ده معناه ان الكنترولار الجديد ده بيورث من ControllerBase كل خواصه و كمان عليهم الاتربيوت ده

[Route("api/[controller]")]

[ApiController]

و بالتالى لو انا ورثت فى كنترولار اليوزر كنترول BaseApiController فكده انا بقتش محتاج انى احط لكنترول اليوزر الاتربيوت الى ذى الاتب ده خالص و طبعا اقدر اوفر على نفسى كتابه كودات كتير لو فى حاجه ثابته فى كل الابلكيشن طبعا و حتى لو هغير فيها يبقى التغييره فى مكان واحد كتيان

و ده شكل ال UserController بعد ما ورق من BaseApiController

UserController.cs

namespace API.Controllers

{

//[Route("api/[controller]")]

//[ApiController] // No need with inherting from BaseApiController

public class UsersController : BaseApiController

{

private readonly DataContext \_dbContext;

public UsersController( DataContext dbContext )

{

\_dbContext = dbContext;

}

[HttpGet]

[Route("GetAllUsers")]

public async Task< ActionResult<List<AppUser>>> GetAllUsers()

{

var Users = await \_dbContext.Users.ToListAsync();

return Users;

}

[HttpGet]

[Route("GetUsers/{Id}")]

public async Task<ActionResult<AppUser>> GetUser(int Id)

{

var User = await \_dbContext.Users.FirstOrDefaultAsync(\_=>\_.Id == Id);

return User;

}

}

}

MB

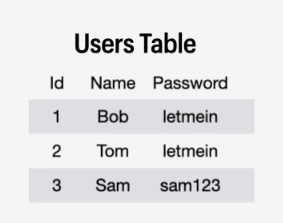
##################################################################

Safe storage of passwords

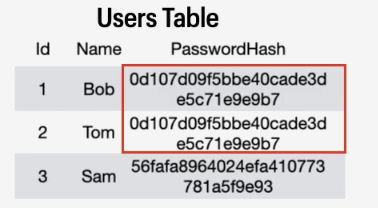
---------------------------

اذاى بحفظ بقه الباسورد احنا عملنا اليوزر اى دى و عملنا الاسم عاوزين بقه نعمل الباسوردات و كده فتعالى نشرح المفروض الباسوردات بتتعمل اذاى يا صديقى

بص يا سيدى اول حاجه لازم تبقى فاهما ان مينفعش احفظ الباسورد بالشكل العادى الى هو 123456 و الكلام العبيط ده لان اى هاكر هيفتح الداتا بيز هيبقى الايميلات الله يرحمها يعنى الشكل الى فى الصوره ده مينفعش نهائى



و لكن المفروض اعمل هاش للباسورد (اشفره يعنى) يبقى بالشكل ده



و لكن الحوار ده ظهر فيه مشكله ان لو الباسوردات الى انا مشفرها هى هياها ساعتها هتبقى متشابهه فلو عرفت واحد منهم هعرف الباقى بكل سهوله فقالك احنا نعمل ايه بقه ؟

احنا نعمل توقيع الكترونى او Salt و يكون عباره عن كلام عشاوئى مشفر و يتسجل مع الباسورد و هو بيتعمله هاش فيكون الهاش الناتج من كل الباسوردات ميتكررش ابدا ابدا لان حتى لو نفس الباسوردات فاكيد مش هيبقى نفس ال salt و بالتالى شكل الهاش كل مره مهما حصل لازم هيبقى مختلف عن الى قبله و بالتالى الشكل هيبقى بالمنظر ده فى الداتابيز



تعالى بقه نعمل الكلام ده كود يا معلم

اول حاجه هنروح نزود ال Entity بتاع ال User ال PasswordHashed , PasswordSalt

و اوعى تنسى تعمل save عشان لو نسيت الميحريشن مش هيطلع الحاجات الجديده يا كبير

namespace API.Entities

{

public class AppUser

{

public int Id { get; set; }

public string UserName { get; set; }

public byte[] PasswordHashed { get; set; }

public byte[] PasswordSalt { get; set; }

}

}

ثم داخل ال package Manager Console هنعمل الميجريشن

dotnet ef migrations add Migration\_AddPasswordHashed --project API -o Data/Migrations

و نابديت الداتا بيز

dotnet ef database update --project API

بس كده انت ظبطت الداتا بيز   
تعالى بقه نظبط الكود

MB

##################################################################

Accounts Controller

---------------------------

ده بقه الكنترولار الى هعمل فيه تسجيل اليوزرات الجديده بالباسوردات بتعتهم و بالتالى هيبقى فيه فانكشن ال Register