# EF core-install

Select Tools🡪Nuget Package Manager🡪Manage NuGet Packages for Solution

1. Install EF Core

* Search Microsoft.EntityFrameworkCore.SqlServer
* Match your .NET version
* Read and accept license

1. Install EF Core Tools

* Repeat 1 with Microsoft.EntityFrameworkCore.Tools

# Create DbContext Class

Three classes in EF Core

* DbContext-primary class for database communication
* DbContextOptions-sconfigurations for DbContext object
* DbSet<Entity>-objects created for the entity stored in a collection (Set)

## XxxxxContext Class that inherits from DbContext class

namespace XxxxxList.Models

{

public class XxxxxContext : DbContext

{

public XxxxxContext(DbContextOptions<XxxxxContext> options)

: base (options)

{ }

public DbSet<Xxxxx> Xxxxx { get; set; }

}

}

## Xxxxx Class with property generated by database

using System.ComponentModel.DataAnnotations;

namespace XxxxList.Models

{

public class Xxxx

{

// EF Core will configure the database to generate this value

public int XxxxId { get; set; }

[Required(ErrorMessage = "Please enter a name.")]

public string Name { get; set; }

// Other Properties

}

}

# Seed Initial Data

## Method of DbContext class

OnModelCreating(mb) – called by Framework on context creation, can be overridden

## Seed initial Data

namespace XxxxxList.Models

{

public class XxxxxContext : DbContext

{

Public XxxxxContext(DbContextOptions<XxxxxContext> options)

: base(options)

{ }

public DbSet<Xxxxx> Xxxxx { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Xxxxx>().HasData(

new Xxxxx {

XxxxxId = 1,

// other properties

},

new Xxxxx {

XxxxxId = 2,

// other properties

// Continue for more objects

}

}

}

# Add Connection String

appsettings.json file

{

"Logging": {

"LogLevel": {

"Default": "Warning"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": {

"XxxxxContext": "Server=(localdb)\\mssqllocaldb;Database=Xxxxxs;

Trusted\_Connection=True;MultipleActiveResultSets=true"

}

}

# Enable Dependency Injection

using Microsoft.Extensions.Hosting;

using Microsoft.EntityFrameworkCore;

using XxxxxList.Xxxxx;

...

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

public void ConfigureServices(IServiceCollection services)

{

...

services.AddDbContext<XxxxxContext>(options =>

options.UseSqlServer(

Configuration.GetConnectionString("XxxxxContext")));

}

...

}

# Use Migrations to Create Database

Select Tools🡪Nuget Package Manager🡪Package Manager Console

1. Verify connection string and dependency injection
2. “Add-Migration Initial” in console and press Enter.
3. “Update-Database” at the command

## Code Up() method

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.CreateTable(

name: "Xxxxxs",

columns: table => new {

XxxxxId = table.Column<int>(nullable: false)

.Annotation("SqlServer:ValueGenerationStrategy",

SqlServerValueGenerationStrategy.IdentityColumn),

*StringProperty* = table.Column<string>(nullable: false),

*intProperty* = table.Column<int>(nullable: false),

….

},

constraints: table => { table.PrimaryKey("PK\_Xxxxxs", x => x.XxxxId);

});

migrationBuilder.InsertData( table: "Xxxxxs",

columns: new[] { "XxxxId", "*StringProperty*",” *intProperty* "},

values: new object[] { 1, " ", 5 });

// *Add more objects*

}

## View Database

1. View🡪SQL Server Object Explorer in Visual Studio.
2. Expand the (localdb)\MSSQLLocalDB
3. Expand the Databases node.
4. Expand the Xxxxxs node
5. Expand the Tables node.
6. Viewing columns
   1. Expand a table node
   2. Expand column node
7. Viewing data
   1. Right-click table
   2. ViewData