# Connecting C# BE to MySQL database

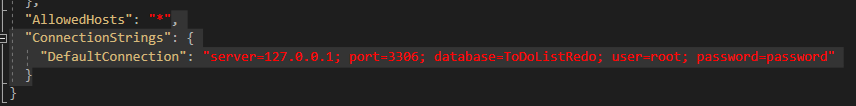
Modify appsettings.json:

,

"ConnectionStrings": {

"DefaultConnection": "server=127.0.0.1; port=3306; database=ToDoListRedo; user=root; password=password"

}



Modify Startup:

private void ConfigureDb(IServiceCollection services)

{

var connectionString = AppConfig.GetConnectionString("DefaultConnection");

var serverVersion = new MySqlServerVersion(new Version(8, 0));

services.AddDbContext<ApplicationDbContext>(

options => options

.UseMySql(connectionString, serverVersion)

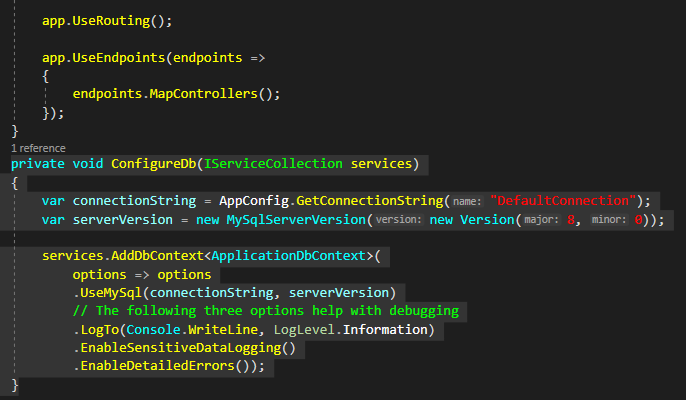
// The following three options help with debugging

.LogTo(Console.WriteLine, LogLevel.Information)

.EnableSensitiveDataLogging()

.EnableDetailedErrors());

}



private IConfiguration AppConfig { get; }

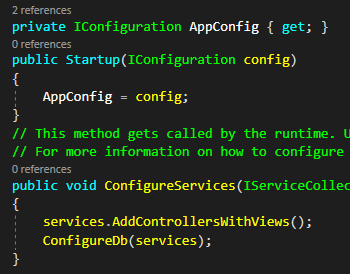
public Startup(IConfiguration config)

{

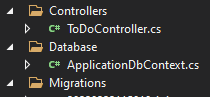
AppConfig = config;

}

ConfigureDb(services);



Create folder Database with class in it



public class ApplicationDbContext : DbContext

{

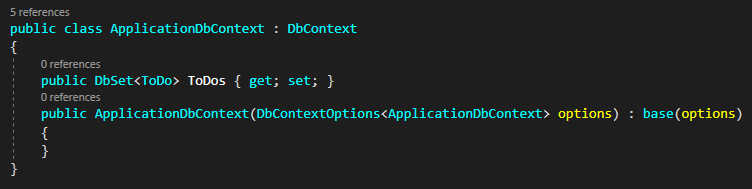
public DbSet<ToDo> ToDos { get; set; }

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options)

{

}

}



Run CLI commands for creating Db:

dotnet ef migrations add init

* Init = name of the migration

No you can check that there is a new folder called Migrations in your project

If everything is correct (columns etc.), run:

dotnet ef database update

This will create your Db, and you can review it in workbench

PS: To run dotnet commands, you need to install this:

dotnet tool install --global dotnet-ef

## Working with Db within code

In your service/s:

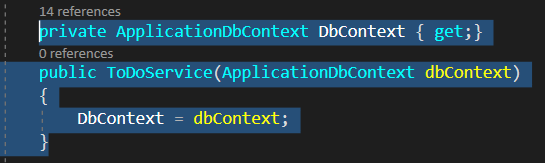
private ApplicationDbContext DbContext { get;}

public ToDoService(ApplicationDbContext dbContext)

{

DbContext = dbContext;

}



On dbContext you are gonna call all the commands for db

Example:

var todo = DbContext.ToDos.Where(f => f.IsDone == false).ToList();

DbContext.Add(todo);

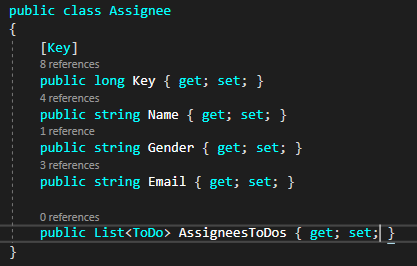
DbContext.SaveChanges();

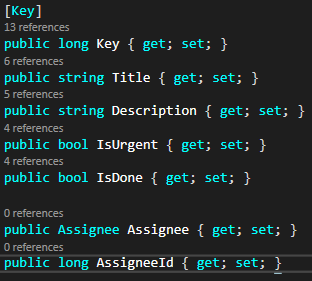
DbContext.ToDos.RemoveRange(todo);

# Connecting Tables

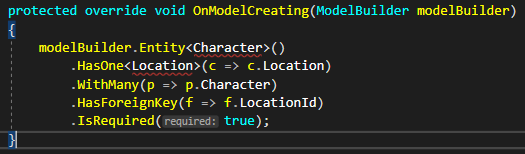
First you need to create variables in your models, which you want to connect.

Example:





And then you have to add a method to your ApplicationDbContext class:



Where you will specific the relations between the tables

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Character>()

.HasOne<Location>(c => c.Location)

.WithMany(p => p.Character)

.HasForeignKey(f => f.LocationId)

.IsRequired(true);

}