<https://docs.microsoft.com/en-us/aspnet/web-api/overview/data/using-web-api-with-entity-framework/>

Accepted Answer

public BooksContext(string connectionString) : base(GetOptions(connectionString))

{

}

private static DbContextOptions GetOptions(string connectionString)

{

return SqlServerDbContextOptionsExtensions.UseSqlServer(new DbContextOptionsBuilder(), connectionString).Options;

}

Popular Answer

1) Add a line to your appsettings.json:

"conn": "Server=s;Database=db;Trusted\_Connection=True;",

2) Read the line in you Startup.cs class (after the Startup method is called to build the Configuration - so usually in the ConfigureServices method) like this:

var **Conn** = Configuration["conn"];

3) If using Entity Framework add a database context service (MyDbContext is the context class generated by EF). You also want to tell the built-in dependency injection how to instantiate your database context:

services.AddDbContext<MyDbContext>(options => options.UseSqlServer(**Conn**));

services.AddScoped<**IMyDbContext**, MyDbContext>();

Where IMyDbContext is (at it's simplist) just an interfa/ce you've extracted from your MyDbContext

4) Now you can define your controller to take a MyDbContext and the DI will take care of building it and passing it in when the controller is invoked:

public MyController(**IMyDbContext** context)

{

\_context = context // store for later use

}

<https://docs.microsoft.com/en-us/ef/core/dbcontext-configuration/>

start.cs

services.AddDbContext<ApplicationDbContext>(

options => options.UseSqlServer("name=ConnectionStrings:DefaultConnection"));

public class **ApplicationDbContext** : DbContext

{

public **ApplicationDbContext**(DbContextOptions<**ApplicationDbContext**> options)

: base(options)

{

}

}

public class MyController

{

private readonly ApplicationDbContext \_context;

public MyController(ApplicationDbContext context)

{

\_context = context;

}

}

public class ApplicationDbContext : DbContext

{

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseSqlServer(@"Server=(localdb)\mssqllocaldb;Database=Test");

}

}

When using ApplicationDbContext defined for ASP.NET Core web apps above:

public class ApplicationDbContext : DbContext

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)

: base(options)

{

}

}

The DbContextOptions can be created and the constructor can be called explicitly:

var contextOptions = new DbContextOptionsBuilder<ApplicationDbContext>()

.UseSqlServer(@"Server=(localdb)\mssqllocaldb;Database=Test")

.Options;

using var context = new ApplicationDbContext(contextOptions);