# How to using Entity Framework DB First in ASP.NET Core

## Introduction

ASP.NET Core is the new cross platform framework for web, and it cannot using the Entity Framework for the .NET Framework version.

This sample will show how to using the Entity Framework .NET Core version in ASP.NET Core.

## Sample prerequisites

* .NET Core 1.0 or later version(s). [[Visual Studio 2015 Home page installer](https://www.visualstudio.com/en-us/visual-studio-homepage-vs.aspx)]
* Microsoft Visual Studio 2015 update3 or above. [[.NET Core + Visual Studio tooling](http://go.microsoft.com/fwlink/?LinkID=798306)]
* You need have a Sql Server database, with below structure,

CREATE DATABASE TestNetCoreEF

GO

USE TestNetCoreEF

GO

CREATE TABLE Student(

ID int identity primary key,

Name nvarchar(50),

Age int

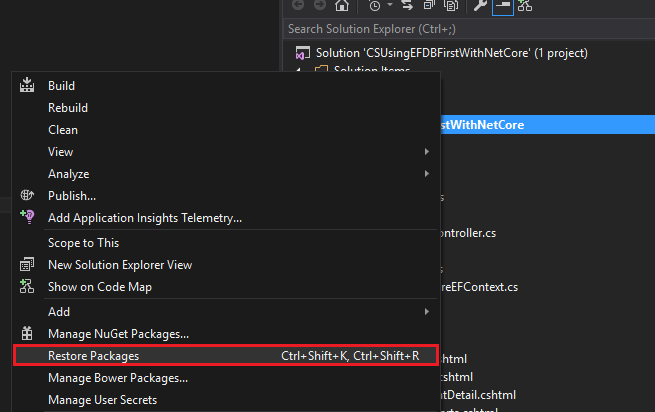
)

INSERT INTO Student VALUES('Bear',18)

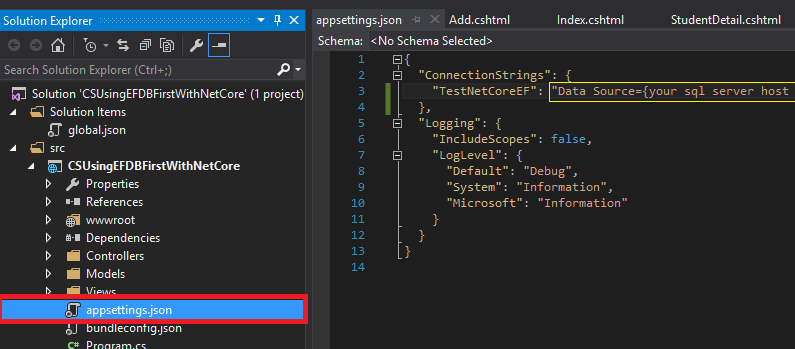
INSERT INTO Student VALUES('Frank',20)

## Building the sample

* **For run this sample**
  + Open the sample solution “**CSUsingEFDBFirstWithNetCore.sln**” using Visula Sutdio.
  + Right click on project “**CSUsingEFDBFirstWithNetCore**” and select Restore packages.



* + Open the file “**appsettings.json**” in project “**CSUsingEFDBFirstWithNetCore**”, and in the ConnectionStrings, config the “TestNetCoreEF” as your SQL DB connect string.



* + Press **F6 Key** or select **Build -> Build Solution** from the menu to build the sample.
* **For build a new project**
  + Create a new ASP.NET Core MVC project
  + Install Entity Framework
    - At menu bar **Tools** -> **NuGet Package Manager** -> **Package Manager Console**
    - Run **Install-Package Microsoft.EntityFrameworkCore.SqlServer**
    - Run **Install-Package Microsoft.EntityFrameworkCore.Tools –Pre**
    - Run **Install-Package Microsoft.EntityFrameworkCore.SqlServer.Design**
  + Open **Project.json**, Locate the Tools section and add the config as shown below

"tools": {

"Microsoft.EntityFrameworkCore.Tools": "1.0.0-preview2-final",

………….

},

* + Reverse engineer your model by command

Run below command at Package Manager Console

Scaffold-DbContext "{ Your DB connect string }" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models

* + The reverse engineer process created entity classes and a derived context based on the schema of the existing database. The entity classes are simple C# objects that represent the data you will be querying and saving.
  + Open the \*\*\*\*Context.cs, add a static field ConnectingString, and update the OnConfiguring event handler as below.

public static string ConnectionString { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseSqlServer(ConnectionString);

}

* + Open the appsettings.json, and config connection string.

{

"ConnectionStrings": {

"TestNetCoreEF": "Data Source={your sql server host address};Initial Catalog=TestNetCoreEF;user id={your username};password={your password};"

},

"Logging": {

"IncludeScopes": false,

"LogLevel": {

"Default": "Debug",

"System": "Information",

"Microsoft": "Information"

}

}

}

* + Open Startup.cs, and add flowing code in ConfigureServices(IServiceCollection services)

public void ConfigureServices(IServiceCollection services)

{

//config the db connection string

TestNetCoreEFContext.ConnectionString = Configuration.GetConnectionString("TestNetCoreEF");

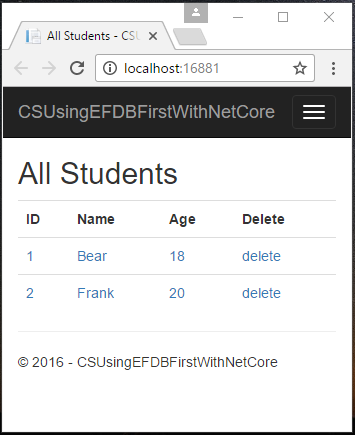
// Add framework services.

services.AddMvc();

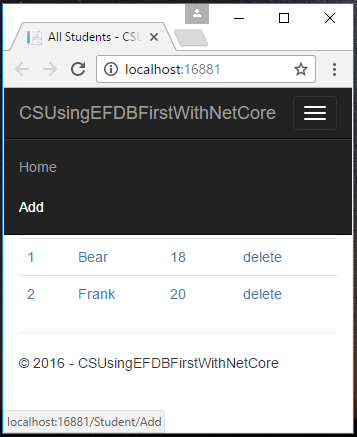
}

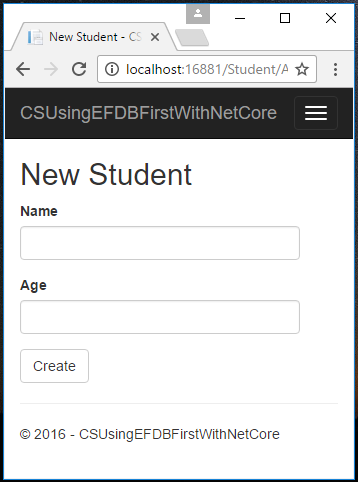
## Running the sample

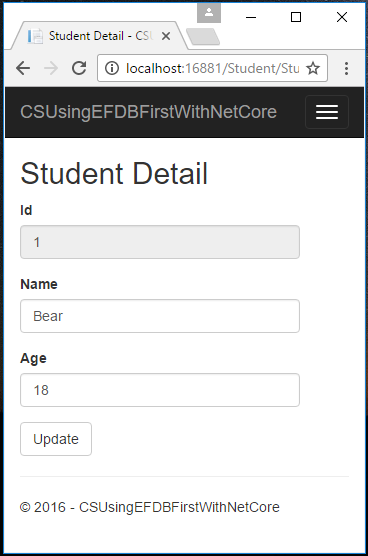
* Open the sample solution using Visual Studio, then press **F5 Key** or select **Debug -> Start Debugging** from the menu.
* When the web application is running, you can see the page in browser.



* You can list or detail student, add student, update student or delete student, feel free in this sample web app.







## Using the code

**Appsettings.json**

{

"ConnectionStrings": {

"TestNetCoreEF": "Data Source={your sql server host address};Initial Catalog=TestNetCoreEF;user id={your username};password={your password};"

},

"Logging": {

"IncludeScopes": false,

"LogLevel": {

"Default": "Debug",

"System": "Information",

"Microsoft": "Information"

}

}

}

**Models.Student.cs**

public partial class Student

{

public int Id { get; set; }

public string Name { get; set; }

public int? Age { get; set; }

}

**Models.TestNetCoreEFContext.cs**

public partial class TestNetCoreEFContext : DbContext

{

public static string ConnectionString { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseSqlServer(ConnectionString);

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Student>(entity =>

{

entity.Property(e => e.Id).HasColumnName("ID");

entity.Property(e => e.Name).HasMaxLength(50);

});

}

public virtual DbSet<Student> Student { get; set; }

}

**Startup.cs**

public class Startup

{

………………

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

//config the db connection string

TestNetCoreEFContext.ConnectionString = Configuration.GetConnectionString("TestNetCoreEF");

// Add framework services.

services.AddMvc();

}

………………

}

**Use demo**

Models.TestNetCoreEFContext context = new Models.TestNetCoreEFContext();

var StudentList = context.Student.ToList();

## More information

ASP.NET Core Application to Existing Database(Database First)

<https://docs.efproject.net/en/latest/platforms/aspnetcore/existing-db.html>