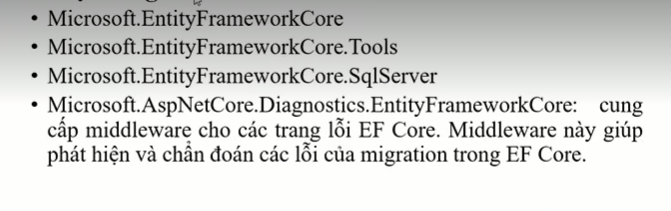
B1:Tools->>

Microsoft.EntityFrameworkCore

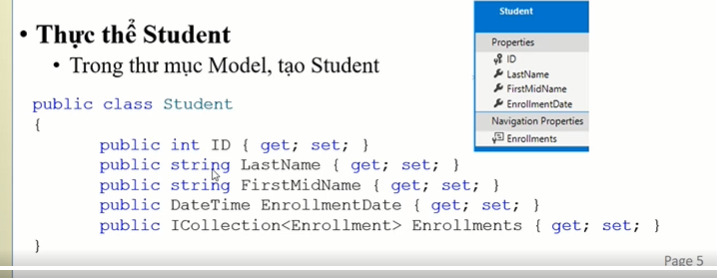
Microsoft.EntityFrameworkCore.Tools

Microsoft.EntityFrameworkCore.Sqlserver

Microsoft.AspNetCore.Diagnostics.EntityFrameworkCore



B2; Model->>



public class Student

{

public int ID { get; set; }

[Required]

[StringLength(50)]

[Display(Name = "Last Name")]

public string LastName { get; set; }

[Required]

[StringLength(50, ErrorMessage = "First name cannot be longer than 50 characters.")]

[Column("FirstName")]

[Display(Name = "First Name")]

public string FirstMidName { get; set; }

[DataType(DataType.Date)]

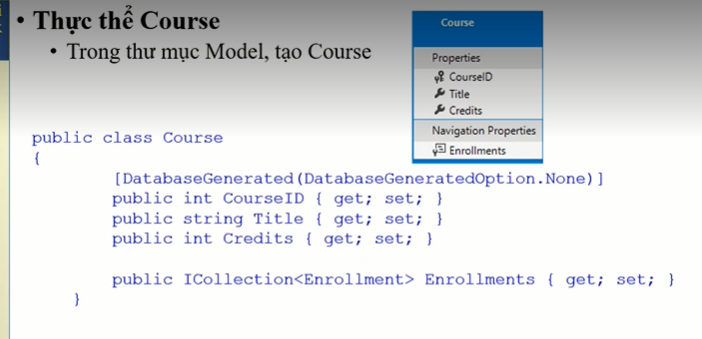
[DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]

[Display(Name = "Enrollment Date")]

public DateTime EnrollmentDate { get; set; }

public ICollection<Enrollment> Enrollments { get; set; }

}



public class Course

{

[DatabaseGenerated(DatabaseGeneratedOption.None)]

[Display(Name = "Number")]

public int CourseID { get; set; }

[StringLength(50, MinimumLength = 3)]

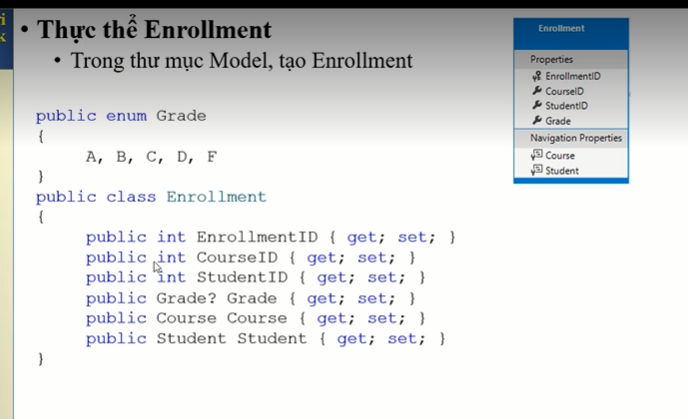
public string Title { get; set; }

[Range(0, 5)]

public int Credits { get; set; }

public ICollection<Enrollment> Enrollments { get; set; }

}



public enum Grade { A,B,C,D,E,F}

public class Enrollment

{

[Key]

public int EnrollmentID { get; set; }

public int CourseID { get; set; }

public int StudentID { get; set; }

[DisplayFormat(NullDisplayText ="No grade")]

public Grade? Grade { get; set; }

public Course Course { get; set; }

public Student Student { get; set; }

}

B3:SchoolContext

public class SchoolContext : DbContext

{

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

{

}

public DbSet<Course> Courses { get; set; }

public DbSet<Enrollment> Enrollments { get; set; }

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

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

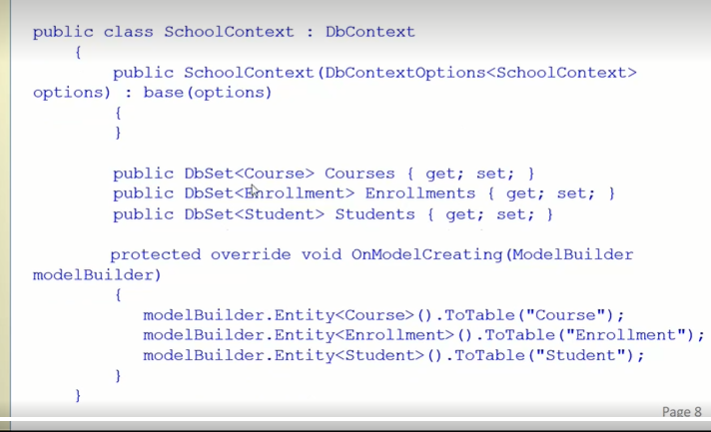
modelBuilder.Entity<Course>().ToTable("Course");

modelBuilder.Entity<Student>().ToTable("Student");

modelBuilder.Entity<Enrollment>().ToTable("Enrollment");

}

}



B4:



public void ConfigureServices(IServiceCollection services)

{

services.AddRazorPages();

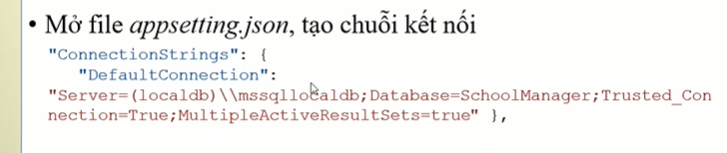
services.AddDbContext<SchoolContext>(options =>

options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));

services.AddDatabaseDeveloperPageExceptionFilter();

}

B5:



"ConnectionStrings": {

"DefaultConnection": "Server=(localdb)\\mssqllocaldb;Database=SchoolManager;Trusted\_Connection=True;MultipleActiveResultSets=true"

}

B6:

public static class DbInitializer

{

public static void Initialize(SchoolContext context)

{

// Look for any students.

if (context.Students.Any())

{

return; // DB has been seeded

}

var students = new Student[] {

new Student{FirstMidName="Carson",LastName="Alexander",EnrollmentDate=DateTime.Parse("2019-09-01")},

new Student{FirstMidName="Meredith",LastName="Alonso",EnrollmentDate=DateTime.Parse("2017-09-01")},

new Student{FirstMidName="Arturo",LastName="Anand",EnrollmentDate=DateTime.Parse("2018-09-01")},

new Student{FirstMidName="Gytis",LastName="Barzdukas",EnrollmentDate=DateTime.Parse("2017-09-01")},

new Student{FirstMidName="Yan",LastName="Li",EnrollmentDate=DateTime.Parse("2017-09-01")},

new Student{FirstMidName="Peggy",LastName="Justice",EnrollmentDate=DateTime.Parse("2016-09-01")},

new Student{FirstMidName="Laura",LastName="Norman",EnrollmentDate=DateTime.Parse("2018-09-01")},

new Student{FirstMidName="Nino",LastName="Olivetto",EnrollmentDate=DateTime.Parse("2019-09-01")}

};

foreach (Student s in students)

{

context.Students.Add(s);

}

context.SaveChanges();

var courses = new Course[] {

new Course{CourseID=1050,Title="Chemistry",Credits=3},

new Course{CourseID=4022,Title="Microeconomics",Credits=3},

new Course{CourseID=4041,Title="Macroeconomics",Credits=3},

new Course{CourseID=1045,Title="Calculus",Credits=4},

new Course{CourseID=3141,Title="Trigonometry",Credits=4},

new Course{CourseID=2021,Title="Composition",Credits=3},

new Course{CourseID=2042,Title="Literature",Credits=4}

};

foreach (Course c in courses)

{

context.Courses.Add(c);

}

context.SaveChanges();

var enrollments = new Enrollment[]

{

new Enrollment{StudentID=1,CourseID=1050,Grade=Grade.A},

new Enrollment{StudentID=1,CourseID=4022,Grade=Grade.C},

new Enrollment{StudentID=1,CourseID=4041,Grade=Grade.B},

new Enrollment{StudentID=2,CourseID=1045,Grade=Grade.B},

new Enrollment{StudentID=2,CourseID=3141,Grade=Grade.F},

new Enrollment{StudentID=2,CourseID=2021,Grade=Grade.F},

new Enrollment{StudentID=3,CourseID=1050},

new Enrollment{StudentID=4,CourseID=1050},

new Enrollment{StudentID=4,CourseID=4022,Grade=Grade.F},

new Enrollment{StudentID=5,CourseID=4041,Grade=Grade.C},

new Enrollment{StudentID=6,CourseID=1045},

new Enrollment{StudentID=7,CourseID=3141,Grade=Grade.A},

};

foreach (Enrollment s in enrollments)

{

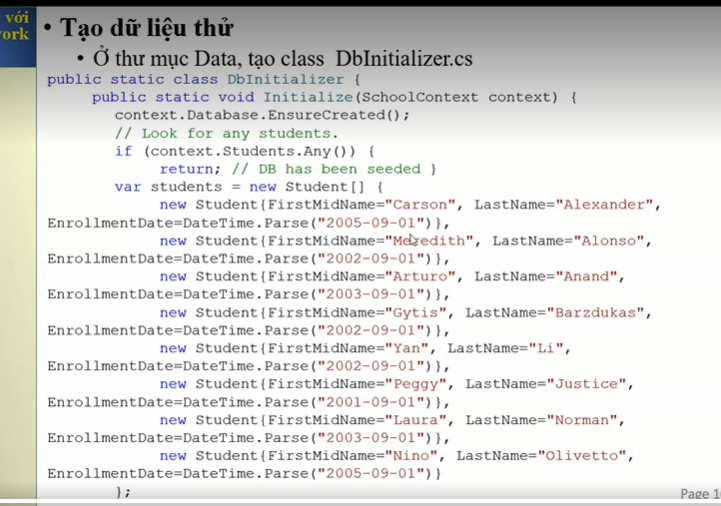
context.Enrollments.Add(s);

}

context.SaveChanges();

}

}





B7: program

public class Program

{

public static void Main(string[] args)

{

var host = CreateHostBuilder(args).Build();

CreateDbIfNotExists(host);

host.Run();

}

private static void CreateDbIfNotExists(IHost host)

{

using (var scope = host.Services.CreateScope())

{

var services = scope.ServiceProvider;

try

{

var context = services.GetRequiredService<SchoolContext>();

/\* context.Database.EnsureCreated();\*/

DbInitializer.Initialize(context);

}

catch (Exception ex)

{

var logger = services.GetRequiredService<ILogger<Program>>();

logger.LogError(ex, "An error occurred creating the DB.");

}

}

}

public static IHostBuilder CreateHostBuilder(string[] args) =>

Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(webBuilder =>

{

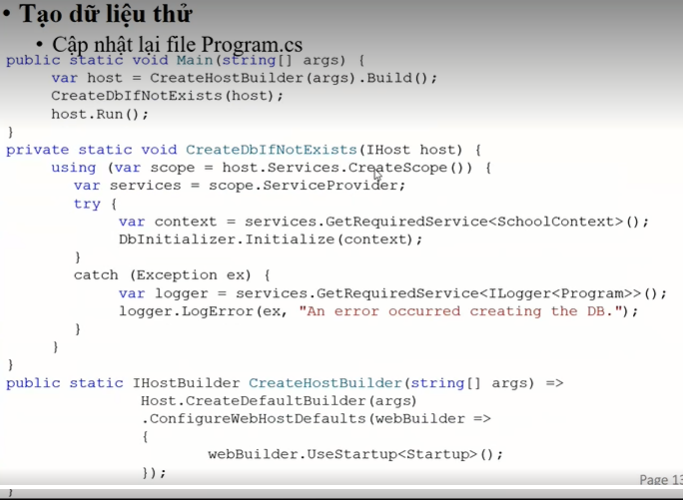
webBuilder.UseStartup<Startup>();

});

}

add-migration AddCategoryToDatabase

update-database



B8:

