Migrations

Il progetto si torva su github all'indirizzo:

https://github.com/lucaSchiavon/Migrations_AppWebDiContosoUniversity

Questo il tutorial microsoft su cui si è realizzato l'esercizio:

https://docs.microsoft.com/it-it/aspnet/core/data/ef-mvc/intro?view=aspnetcore-3.1

Il pacchetto EF è incluso in netcore

Creare le entities

Es:

```
using System;
using System.Collections.Generic;

namespace ContosoUniversity.Models
{
    public class Student
    {
        public int ID { get; set; }
        public string LastName { get; set; }
        public string FirstMidName { get; set; }
        public DateTime EnrollmentDate { get; set; }

        public ICollection<Enrollment> Enrollments { get; set; }
    }
}
namespace ContosoUniversity.Models
{
    public enum Grade
    {
        A, B, C, D, F
    }

    public class Enrollment
    {
        public int EnrollmentID { get; set; }
        public int CourseID { get; set; }
        public int StudentID { get; set; }
        public Grade? Grade { get; set; }

        public Course Course { get; set; }
        public Student Student { get; set; }
    }
}

using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;

namespace ContosoUniversity.Models
{
    public class Course
    {
        [DatabaseGenerated(DatabaseGeneratedOption.None)]
```

```
        public int CourseID { get; set; }
        public string Title { get; set; }
        public int Credits { get; set; }

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

Creare il contesto di db:

```
using ContosoUniversity.Models;
using Microsoft.EntityFrameworkCore;

namespace ContosoUniversity.Data
{
    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<Enrollment>().ToTable("Enrollment");
modelBuilder.Entity<Student>().ToTable("Student"); }
    }
```

Registrare SchoolContext

```
public void ConfigureServices(IServiceCollection services)
{
    services.Configure<CookiePolicyOptions>(options =>
    {
        options.CheckConsentNeeded = context => true;
        options.MinimumSameSitePolicy = SameSiteMode.None;
    });

 services.AddDbContext<SchoolContext>(options =>
options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));

    services.AddMvc();
}
```

```
Impostare la stringa di connessione
{
 "ConnectionStrings": { "DefaultConnection":
"Server=(localdb)\\mssqllocaldb;Database=ContosoUniversity1;Trusted_Conne
ction=True;MultipleActiveResultSets=true" },
  "Logging": {
    "IncludeScopes": false,
    "LogLevel": {
      "Default": "Warning"
```

```
    }
  }
}


}
```

Creare routines per inizializzare i dati:

```
using ContosoUniversity.Models;
using System;
using System.Linq;

namespace ContosoUniversity.Data
{
    public static class DbInitializer
    {
        public static void Initialize(SchoolContext context)
        {
            context.Database.EnsureCreated();

            // 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("2
005-09-01")},
            new
Student{FirstMidName="Meredith",LastName="Alonso",EnrollmentDate=DateTime.Parse("20
02-09-01")},
            new
Student{FirstMidName="Arturo",LastName="Anand",EnrollmentDate=DateTime.Parse("2003-
09-01")},
            new
Student{FirstMidName="Gytis",LastName="Barzdukas",EnrollmentDate=DateTime.Parse("20
02-09-01")},
            new
Student{FirstMidName="Yan",LastName="Li",EnrollmentDate=DateTime.Parse("2002-09-
01")},
            new
Student{FirstMidName="Peggy",LastName="Justice",EnrollmentDate=DateTime.Parse("2001
-09-01")},
            new
Student{FirstMidName="Laura",LastName="Norman",EnrollmentDate=DateTime.Parse("2003-
09-01")},
            new
Student{FirstMidName="Nino",LastName="Olivetto",EnrollmentDate=DateTime.Parse("2005
-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 e in enrollments)
            {
                context.Enrollments.Add(e);
            }
            context.SaveChanges();
        }
    }
}
```

Modificare il main per eseguire tali operazioni di popolamento solo all'avvio:

```
public static void Main(string[] args)
{
 var host = CreateWebHostBuilder(args).Build();
using (var scope = host.Services.CreateScope()) {
var services = scope.ServiceProvider;
try {
var context = services.GetRequiredService<SchoolContext>();
DbInitializer.Initialize(context);
}
catch (Exception ex) {
```

```
var logger = services.GetRequiredService<ILogger<Program>>();
logger.LogError(ex, "An error occurred while seeding the database.");
}
}
host.Run();
}
```

Poi creare i controller con gli scaffolding entity framework (questo crea il controller e le view)

Lanciato il comando

```
D:\L_SCHIAVON_DOCUMENTS\LAVORO\NOGIT\EsercitazioniAspNetCore_Microsoft\AppWebCon
tosoUniversity\AppWebDiContosoUniversity>dotnet ef migrations add InitialCreate
info: Microsoft.EntityFrameworkCore.Infrastructure[10403]
      Entity Framework Core 2.2.4-servicing-10062 initialized 'SchoolContext' us
ing provider 'Microsoft.EntityFrameworkCore.SqlServer' with options: None
Done. To undo this action, use 'ef migrations remove'
```

Si crea una cartella migrations con i file di migrazione

EsercitazioniAspNetCore_Microsoft › AppWebContosoUniversity › AppWebDiContosoUniversity › Migrations

Nome	Ultima modifica	Tipo	Dimensione
20200325211209_InitialCreate.cs	25/03/2020 22:12	Visual C# Source f...	4 KB
20200325211209_InitialCreate.Designer.cs	25/03/2020 22:12	Visual C# Source f...	4 KB
SchoolContextModelSnapshot.cs	25/03/2020 22:12	Visual C# Source f...	4 KB

Dentro la cartella c'è una classe il cui metodo up crea il db ed il down cancella

C# Copia

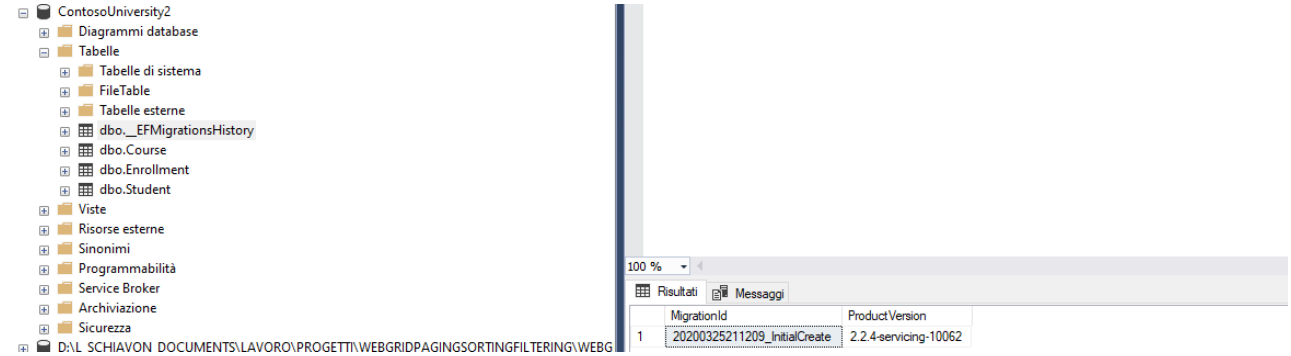
```
public partial class InitialCreate : Migration
{
    protected override void Up(MigrationBuilder migrationBuilder)
    {
        migrationBuilder.CreateTable(
            name: "Course",
            columns: table => new
            {
                CourseID = table.Column<int>(nullable: false),
                Credits = table.Column<int>(nullable: false),
                Title = table.Column<string>(nullable: true)
            },
            constraints: table =>
            {
                table.PrimaryKey("PK_Course", x => x.CourseID);
            });

        // Additional code not shown
    }

    protected override void Down(MigrationBuilder migrationBuilder)
    {
        migrationBuilder.DropTable(
            name: "Enrollment");
        // Additional code not shown
    }
}
```

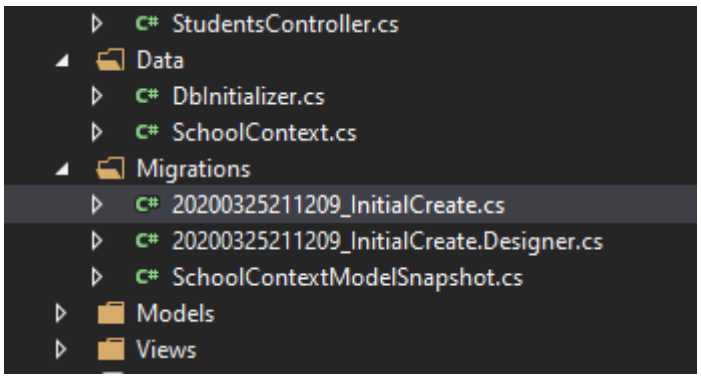
Le migrazioni creano uno *snapshot* dello schema del database corrente in *Migrations/SchoolContextModelSnapshot.cs*. Quando si aggiunge una migrazione, EF determina le modifiche apportate confrontando il modello di dati con il file dello snapshot.

Dotnet ef migrations remove rimuove una migrazione.

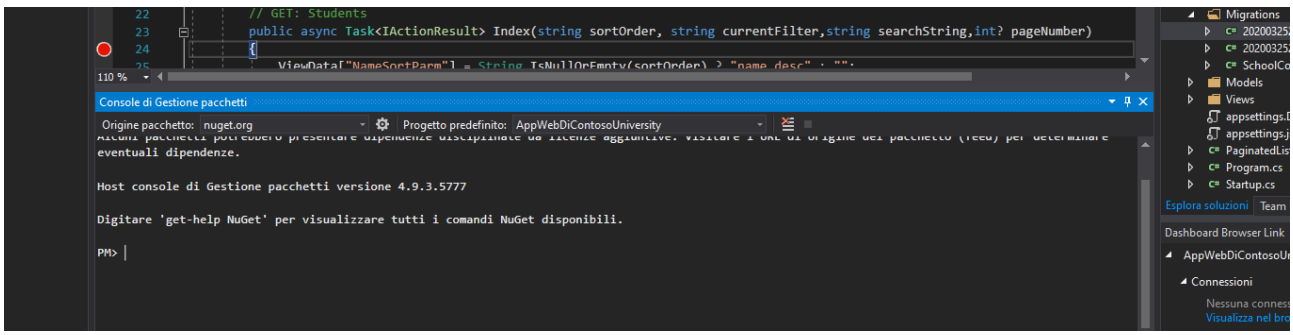
Ora lanciando il comando dotnet ef database update viene creato il DB

E se si apre la tabella migration si nota che la migrazione 1 è stata applicata

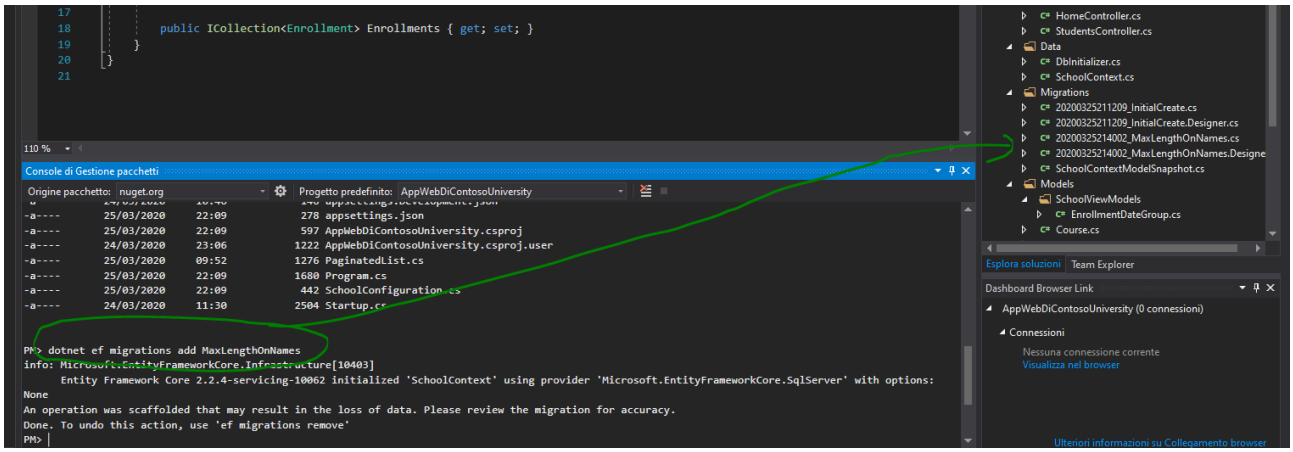




In questa esercitazione viene illustrato come usare la CLI, ma se si preferisce è possibile usare la console di Gestione pacchetti



Se poi si modificano dei dati e si vuole riapplicare la migration ai db occorre lanciare questo comando che creerà gli script di migration:



E successivamente applicare la migrazione

Includere dati tra tabelle correlate:

(questo tipo di caricamento, se si necessita di tutti i dati una sola volta è il caricamento più efficace: con include

```
var departments = _context.Departments.Include(d => d.Courses);
foreach (Department d in departments)
{
    foreach(Course c in d.Courses)
    {
        courseList.Add(d.Name + c.Title);
    }
}
```

Query: all Department entities and related Course entities

Includere dati all'occorrenza:

```
var departments = _context.Departments;
foreach (Department d in departments)
{
    _context.Entry(d).Collection(p => p.Courses).Load();
    foreach (Course c in d.Courses)
    {
        courseList.Add(d.Name + c.Title);
    }
}
```

Query: all Department rows

Query: Course rows related to Department d

Caricamento lazy. Quando un'entità viene letta per la prima volta, i dati correlati non vengono recuperati. La prima volta che si tenta di accedere a una proprietà di navigazione, tuttavia, i dati necessari per quest'ultima vengono recuperati automaticamente.