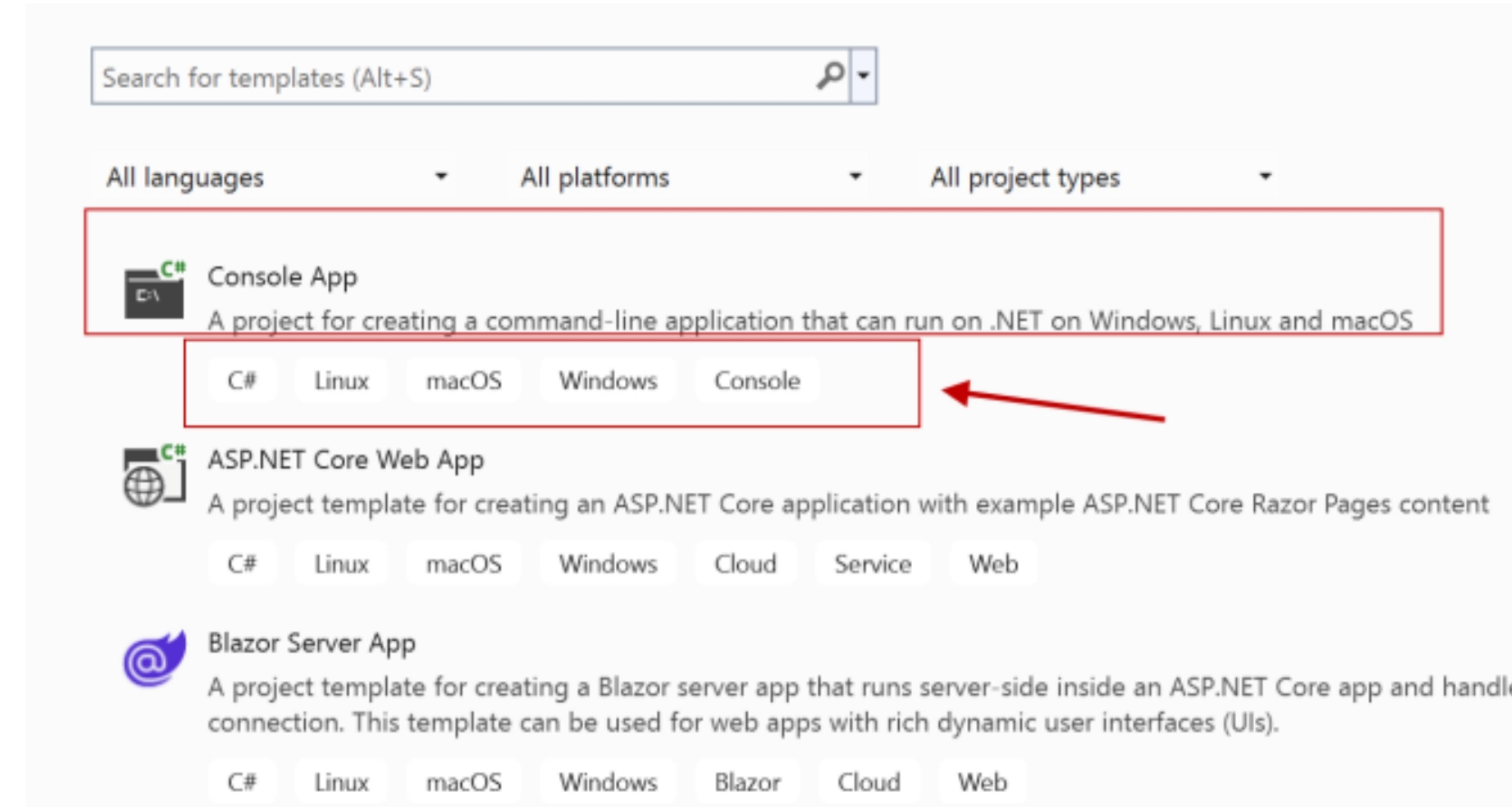


LAB 4: ENTITY FRAMEWORK CORE

Bài 1:

Sử dụng EF Core Code First Fluent API thiết kế Database gồm có hai bảng với các yêu cầu sau:

Tạo Project Console app và cài đặt các Package tương tự như bài Lab3



Microsoft.EntityFrameworkCore.SqlServer

Microsoft.EntityFrameworkCore.Tools

Tạo các lớp có các thuộc tính sau:

```
namespace BookManagement
{
    public class Publisher
    {
        public int ID { get; set; }
        public string Name { get; set; }
        public ICollection<Book> Books { get; set; }
    }
}
```

```

6 references
public class Book
{
    1 reference
    public int BookID { get; set; }
    4 references
    public string Title { get; set; }
    2 references
    public string Author { get; set; }
    2 references
    public string Language { get; set; }
    2 references
    public int Pages { get; set; }
    1 reference
    public int PublisherId { get; set; }
    5 references
    public Publisher Publisher { get; set; }
}

```

Mỗi quan hệ giữa hai bảng Publisher và Book:



Thêm lớp BookDB Context và sử dụng Fluent API tạo mối quan hệ giữa hai bảng như bên dưới:

```
namespace BookManagement
{
    public class BookDBContext : DbContext
    {
        public DbSet<Book> Books { get; set; }
        public DbSet<Publisher> Publisher { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlServer(@"Server=.; Database=BookDB; Trusted_Connection=True; TrustServerCertificate=True");
        }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);

            modelBuilder.Entity<Publisher>().Property(e => e.Name).IsRequired();
            modelBuilder.Entity<Book>()
                .HasOne(d => d.Publisher)
                .WithMany(p => p.Books)
                .HasForeignKey(e => e.PublisherId);
            modelBuilder.Entity<Book>()
                .Property(e => e.Title)
                .IsRequired();
        }
    }
}
```

Chạy lệnh:

add-migration InitDb

update-database InitDb

Method thêm dữ liệu vào Database:

```
private static void InsertSampleData()
{
    using (var context = new BookDBContext())
    {
        var publisher = new Publisher
        {
            Name = "Times Education"
        };

        context.Publisher.Add(publisher);

        //Add some books
        context.Books.Add(new Book
        {
            Title = "C# Programming",
            Author = "Alex",
            Language = "English",
            Pages = 880,
            Publisher = publisher
        });

        context.Books.Add(new Book
        {
            Title = "Java Programming",
            Author = "Andrew",
            Language = "English",
            Pages = 900,
            Publisher = publisher
        });

        //Saves changes
        context.SaveChanges();
    }
}
```

Method đọc dữ liệu từ Database và hiển thị lên màn hình Console:

```
private static void PrintData()
{
    // Gets and prints all books in database
    using (var context = new BookDBContext())
    {
        var books = context.Books.Include(p => p.Publisher);
        foreach (var book in books)
        {
            var data = new StringBuilder();
            data.AppendLine($"ID: {book.BookID}");
            data.AppendLine($"Title: {book.Title}");
            data.AppendLine($"Publisher: {book.Publisher.Name}");
            Console.WriteLine(data.ToString());
        }
    }
}
```

Kết quả:



The screenshot shows the Microsoft Visual Studio Debug Console window. It displays two entries of movie data:
ID: 1
Title: C# Programming
Publisher: Times Education

ID: 2
Title: Java Programming
Publisher: Times Education

Lý thuyết:

Cấu hình quan hệ nhiều-nhiều (Many-To-Many) sử dụng EF Core Code First:

```
public class Movie
{
    public int Id { get; set; }
    public string Name { get; set; }
    /* EF Relations */
    public ICollection<Actor> Actors { get; set; }
}

public class Actor
{
    public int Id { get; set; }
    public string Name { get; set; }
    /* EF Relations */
    public ICollection<Movie> Movies { get; set; }
}

private static void ManyToManyRelationship(EFCoreRelationshipsExamplesDbContext context)
{
    var movie1 = new Movie { Name = "The Godfather" };
    var movie2 = new Movie { Name = "Scarface" };
    context.AddRange(
        new Actor
        {
            Name = "Marlon Brando",
            Movies = new List<Movie> { movie1 }
        },
        new Actor
        {
            Name = "Al Pacino",
            Movies = new List<Movie> { movie1, movie2 }
        });
    context.SaveChanges();
}
```

SQL Query Results:

```

SELECT * FROM Actors
SELECT * FROM Movies
SELECT * FROM ActorMovie

```

	Id	Name
1	1	Marlon Brando
2	2	Al Pacino

	Id	Name
1	1	The Godfather
2	2	Scarface

	ActorsId	MoviesId
1	1	1
2	2	1
3	2	2

Truy xuất dữ liệu từ Database:

```

private static void PrintManyToManyRelationship()
{
    using (var context = new EFCoreRelationshipsExamplesDbContext())
    {
        foreach (var actor in context.Actors.Include(a => a.Movies))
        {
            Console.WriteLine($"Actor: {actor.Name}");
            foreach (var movie in actor.Movies)
            {
                Console.WriteLine($"Movie: " + movie.Name);
            }
        }
    }
}

```

Select Microsoft Visual Studio Debug Console

```

Actor: Marlon Brando
Movie: The Godfather

Actor: Al Pacino
Movie: The Godfather
Movie: Scarface

```

Bài tập thực hành:

Bài 1: Thiết kế CSDL một hệ thống quản lý tài xế cho một ứng dụng gọi xe. Hệ thống này bao gồm các thực thể sau: Driver, Car, và RideRequest. Hãy tạo mới các lớp sau:

- Driver.cs
- Car.cs
- RideRequest.cs

Chi tiết các lớp:

Tài xế (Driver):

```
public class Driver  
{
```

```
    public int DriverId { get; set; }  
  
    public string FullName { get; set; }  
  
    public DateTime BirthDate { get; set; }
```

```
}
```

Xe (Car):

```
public class Car  
{
```

```
    public int CarId { get; set; }  
  
    public string Model { get; set; }  
  
    public int Year { get; set; }  
  
    public int Seats { get; set; }
```

```
}
```

Yêu cầu đi xe (RideRequest):

```
public class RideRequest  
{
```

```
    public int RideRequestId { get; set; }  
  
    public DateTime RequestTime { get; set; }  
  
    public string PickupLocation { get; set; }  
  
    public string DropoffLocation { get; set; }
```

```
}
```

Yêu cầu:

- Một tài xế có thể sở hữu nhiều xe, nhưng mỗi xe chỉ thuộc về một tài xế.
- Một yêu cầu đi xe có thể được nhận bởi một tài xế, và một tài xế có thể nhận nhiều yêu cầu đi xe.
- Cấu hình các thuộc tính sao cho tên tài xế và mẫu xe có độ dài tối đa là 90 ký tự.
- Sử dụng EF Core Code First thiết kế cơ sở dữ liệu cho hệ thống này.

Hướng dẫn:

- Tạo các lớp Driver, Car, và RideRequest với các thuộc tính được chỉ định.
- Cấu hình DbContext để thiết lập các mối quan hệ giữa các thực thể.
- Đảm bảo rằng các thuộc tính có giới hạn độ dài như đã nêu.

Bài 2: Cho các lớp sau biểu diễn các thực thể của giải bóng đá, câu lạc bộ và cầu thủ như sau:

Tạo mới các Class sau:

[LeagueFootball.cs](#)

[Player.cs](#)

[FootballClub.cs](#)

Thêm các dữ liệu cho các Class như sau:

//Giải đấu (**LeagueFootball**)

```
public class LeagueFootball
{
    public int LeagueFootballId { get; set; }

    public string LeagueName { get; set; }

    public int TeamCount { get; set; }

}
```

//Cầu thủ (**Player**)

```
public class Player
{
    public int PlayerId { get; set; }

    public string FullName { get; set; }

    public DateTime BirthDate { get; set; }

    public DateTime LicenseDate { get; set; }

    public FootballClub CurrentTeam { get; set; }

}
```

//Câu lạc bộ bóng đá (**FootballClub**)

```
public class FootballClub
{
    public int FootballClubId { get; set; }

    public string FootballName { get; set; }

    public DateTime DateEstablished { get; set; }

    public List<Player> Players { get; set; }

}
```

Yêu cầu:

- Một cầu thủ chỉ thuộc về một đội bóng chủ quản.
- Một giải đấu có nhiều câu lạc bộ tham gia, một câu lạc bộ có thể tham gia nhiều giải đấu. Ví dụ trong một mùa giải câu lạc bộ có thể vừa tham gia Premier League vừa có thể tham gia UEFA Champions League.

Cấu hình các thuộc tính:

- Tên câu lạc bộ và tên cầu thủ có độ dài tối đa là 95 ký tự.
- Sử dụng EF Core Code First thiết kế cơ sở dữ liệu trên.

