Lab 3: Using EF Core CLI to Create and Apply Migrations

# Introduction

In modern application development, the data model often evolves as new features are added. This means the underlying database schema must also change to stay in sync with the application’s requirements. Entity Framework Core (EF Core) Migrations provide a structured and incremental way to update the database schema as your application's data model changes, all while preserving existing data.

# What are EF Core Migrations?

Migrations are a feature in EF Core that allow developers to:  
  
- Incrementally update the database schema to match changes in the application's data model.  
- Track and apply schema changes over time, preserving data and schema history.  
- Revert to previous schema versions if necessary.  
  
EF Core compares your current data model with the previous state (snapshot) and generates migration files that describe the required updates. These files are tracked in your project's source control, just like any other code.

# Why Use Migrations?

- Keeps database and code in sync: Ensures your database structure matches your C# models.  
- Preserves data: Updates the schema without dropping existing tables or losing data.  
- Version control: Migration files can be committed to source control, allowing for team collaboration and rollback if needed.  
- Automates SQL generation: Developers don’t need to manually write SQL for schema changes.

# How EF Core Migrations Work

1. Model Change: You modify your C# model classes or DbContext.  
2. Migration Creation: You use the EF Core CLI to create a migration. EF Core analyzes the differences and generates code files describing the schema changes.  
3. Migration Application: You apply the migration to the database, updating its schema.  
4. History Tracking: EF Core records applied migrations in a special table (\_\_EFMigrationsHistory) in the database, so it knows which migrations have been applied.

# Step-by-Step Procedure

## Install EF Core CLI Tools

dotnet tool install --global dotnet-ef

## Create the Initial Migration

dotnet ef migrations add InitialCreate

## Apply the Migration to Create the Database

dotnet ef database update

## Verify in SQL Server

Open SQL Server Management Studio (SSMS) or Azure Data Studio. Connect to your database server. Expand the database and view the Tables node. You should see tables corresponding to your models (e.g., Products, Categories), as well as the \_\_EFMigrationsHistory table.

# Example Migration File

public partial class InitialCreate : Migration  
{  
 protected override void Up(MigrationBuilder migrationBuilder)  
 {  
 migrationBuilder.CreateTable(  
 name: "Categories",  
 columns: table => new  
 {  
 Id = table.Column<int>(nullable: false)  
 .Annotation("SqlServer:Identity", "1, 1"),  
 // other columns...  
 },  
 constraints: table =>  
 {  
 table.PrimaryKey("PK\_Categories", x => x.Id);  
 });  
  
 migrationBuilder.CreateTable(  
 name: "Products",  
 columns: table => new  
 {  
 Id = table.Column<int>(nullable: false)  
 .Annotation("SqlServer:Identity", "1, 1"),  
 // other columns...  
 },  
 constraints: table =>  
 {  
 table.PrimaryKey("PK\_Products", x => x.Id);  
 });  
 }  
  
 protected override void Down(MigrationBuilder migrationBuilder)  
 {  
 migrationBuilder.DropTable(name: "Products");  
 migrationBuilder.DropTable(name: "Categories");  
 }  
}

Submitted by : Tanishq Pandey