NUGET

Step-by-Step Guide to Apply EF Core with SQL Server (Code First Approach)

1. Create a .NET 6+ Console Application:

- Open Visual Studio or your preferred IDE.

- Create a new .NET Console Application targeting .NET 6+.

2. Install Entity Framework Core NuGet Packages:

- Right-click on your project in Solution Explorer.

- Select "Manage NuGet Packages..."

- Install the following packages:

**- Microsoft.EntityFrameworkCore.SqlServer**

**- Microsoft.EntityFrameworkCore.Tools (for migrations and scaffolding)**

3. Define Your Entity Classes:

- Create C# classes that represent your database entities. For example:

POCO 🡺 Plain Old CLR Objects

CSharp

public class Product

{

public int ProductId { get; set; }

public string Name { get; set; }

public int Price { get; set; }

}

4. Create a DbContext Class:

- Define a class that inherits from `DbContext` and includes properties for your entities:

public class AppDbContext: DbContext

{

public DbSet<Product> Products { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseSqlServer("YourConnectionString");

// Replace "YourConnectionString" with your actual SQL Server connection string

}

}

5. Generate Migrations and Apply to Database:

- Open the Package Manager Console (PMC):

- Tools -> NuGet Package Manager -> Package Manager Console

- Run the following commands to create migrations and apply them to your database:

// Add a migration

Add-Migration InitialCreate -Context AppDbContext

// Update the database with the generated migration

Update-Database -Context AppDbContext

6. Access Your Data Using EF Core

- You can now use EF Core to interact with your database. For example, to add a new product:

CSHarp

using (var dbContext = new AppDbContext())

{

var newProduct = new Product { Name = "New Product", Price = 9.99 };

dbContext.Products.Add(newProduct);

dbContext.SaveChanges();

}

Fetch Data from the Table

using (var dbContext = new AppDbContext())

{

var products = dbContext.Products.ToList(); // Retrieve all products

// Display the products

foreach (var product in products)

{

Console.WriteLine($"Product ID: {product.ProductId}, Name: {product.Name}, Price: {product.Price}");

}

}

Update Data …

public static void UpdateProduct(int productId, string newName, decimal newPrice)

{

using (var dbContext = new AppDbContext())

{

var product = dbContext.Products.Find(productId);

if (product != null)

{

product.Name = newName;

product.Price = newPrice;

dbContext.SaveChanges();

}

}

}

Delete Data

public static void DeleteProduct(int productId)

{

using (var dbContext = new AppDbContext())

{

var product = dbContext.Products.Find(productId);

if (product != null)

{

dbContext.Products.Remove(product);

dbContext.SaveChanges();

}

}

}