Lab 2: Setting Up the Database Context for a Retail Store

This document provides a step-by-step guide for configuring the Entity Framework Core DbContext to connect a .NET application to a SQL Server database for a retail inventory system. It covers model creation, context configuration, connection string setup, and best practices for usage.

# 1. What is DbContext in EF Core?

DbContext is the primary class in Entity Framework Core responsible for managing database connections and tracking entity objects during runtime.  
  
It acts as a bridge between your C# classes (models) and the database tables, enabling CRUD (Create, Read, Update, Delete) operations and change tracking.

# 2. Creating Entity Models

Define the data structure for your application by creating C# classes. For a retail store, you typically need Category and Product models.

public class Category  
{  
 public int Id { get; set; }  
 public string Name { get; set; }  
 public List<Product> Products { get; set; }  
}  
  
public class Product  
{  
 public int Id { get; set; }  
 public string Name { get; set; }  
 public decimal Price { get; set; }  
 public int CategoryId { get; set; }  
 public Category Category { get; set; }  
}

Each class maps to a table in the database.  
  
Navigation properties (Products in Category, Category in Product) define relationships.

# 3. Creating the AppDbContext Class

Create a new class (e.g., AppDbContext.cs) that inherits from DbContext:

using Microsoft.EntityFrameworkCore;  
  
public class AppDbContext : DbContext  
{  
 public DbSet<Product> Products { get; set; }  
 public DbSet<Category> Categories { get; set; }  
  
 protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)  
 {  
 optionsBuilder.UseSqlServer("Your\_Connection\_String\_Here");  
 }  
}

DbSet<T> properties represent tables.  
  
OnConfiguring specifies the database provider and connection string.

# 4. Setting the Connection String

The connection string tells EF Core how to connect to your SQL Server database.  
  
Example for local SQL Server:

Server=localhost;Database=RetailInventoryDb;Trusted\_Connection=True;

Example for SQL Server Express:

Server=.\SQLEXPRESS;Database=RetailInventoryDb;Trusted\_Connection=True;

Example for SQL Authentication:

Server=localhost;Database=RetailInventoryDb;User Id=YOUR\_USERNAME;Password=YOUR\_PASSWORD;

Replace "Your\_Connection\_String\_Here" in OnConfiguring with your actual connection string.  
  
For security and flexibility, consider storing the connection string in appsettings.json (recommended for ASP.NET Core apps).

# 5. Using appsettings.json (Recommended for ASP.NET Core)

appsettings.json example:

{  
 "ConnectionStrings": {  
 "DefaultConnection": "Server=localhost;Database=RetailInventoryDb;Trusted\_Connection=True;"  
 }  
}

To use this, configure your DbContext to accept options via dependency injection and read from configuration.

Submitted by: Tanishq Pandey