**ENTITY FRAMEWORK KURULUM KOMUTLARI DAL**

Install-Package Microsoft.EntityFrameworkCore

Install-Package Npgsql.EntityFrameworkCore.PostgreSQL

Install-Package Microsoft.EntityFrameworkCore.Tools

Scaffold-DbContext "Host=localhost;Port=5432;Username=postgres;Password=123456;Database=obs\_db;" Npgsql.EntityFrameworkCore.PostgreSQL -OutputDir Models -Context ObsDbContext –Force

Aldığınız hata, **dotnet ef** komutunun bilgisayarınızda tanımlı olmadığını veya eksik olduğunu gösteriyor. Bu durum genellikle **Entity Framework Core Tools**'un yüklü olmadığı anlamına gelir. Bu problemi çözmek için aşağıdaki adımları izleyebilirsiniz:

**1. Entity Framework Core Tools Kurulumu**

**EF Core Tools**'u yüklemek için şu komutu kullanın:

dotnet tool install --global dotnet-ef

Bu komut, **dotnet ef** aracını bilgisayarınıza yükleyecektir. Eğer **dotnet-ef** zaten kuruluysa, bunu güncelleyebilirsiniz:

dotnet tool update --global dotnet-ef

**2. Proje Bağımlılıklarını Güncelleme**

Ardından, **DAL** projenizin kök dizininde **Entity Framework Core** ve **Npgsql.EntityFrameworkCore.PostgreSQL** NuGet paketlerinin yüklü olduğundan emin olun. Bunları yüklemek için şu komutları çalıştırabilirsiniz:

dotnet add package Microsoft.EntityFrameworkCore

dotnet add package Npgsql.EntityFrameworkCore.PostgreSQL

Bu komutlar, Entity Framework Core ve PostgreSQL için gerekli bağımlılıkları projeye ekleyecektir.

**3. Proje Dosyasını Kontrol Etme**

Son olarak, **DbContext** sınıfının ve **DbSet**'lerinin doğru şekilde tanımlandığından emin olun. **ObsDbContext** sınıfının bulunduğu dosyanın doğru olduğuna ve **DbContext**'in doğru şekilde yapılandırıldığına emin olun.

**4. Komutu Tekrar Çalıştırma**

Yukarıdaki adımları tamamladıktan sonra, tekrar **Scaffold-DbContext** komutunu çalıştırmayı deneyin:

dotnet ef dbcontext scaffold "Host=localhost;Port=5432;Username=postgres;Password=123456;Database=obs\_db;" Npgsql.EntityFrameworkCore.PostgreSQL --output-dir Models --context ObsDbContext

Eğer yukarıdaki adımlar doğru şekilde uygulandıysa, komut başarılı bir şekilde çalışmalıdır ve veritabanı tablolarını **Models** klasörüne Entity modelleri olarak oluşturacaktır.

**Sonuç**

* **dotnet ef** komutunun eksik olduğu hatasını alıyorsanız, öncelikle **Entity Framework Core Tools**'u yüklemeniz gerekiyor.
* Bağımlılıkları doğru şekilde ekledikten sonra, tekrar komutu çalıştırabilirsiniz.

**Konum: dal.csproj un dosyasının shelli ve cmd**

Not: veritabanı giriş bilgilerinin jsonda saklanması: güvenlik için:

Bu sefer **Scaffold-DbContext** komutunun başarılı bir şekilde çalıştığını ve veritabanı bağlantınızın düzgün şekilde yapıldığını görüyoruz. Ancak, mesajda birkaç önemli uyarı da var. Bunları açıklayayım:

**1. Bağlantı Dizesinin Güvenliği:**

* **Uyarı Mesajı**: "To protect potentially sensitive information in your connection string, you should move it out of source code."
* Bu uyarı, bağlantı dizesindeki hassas bilgilerin (kullanıcı adı, şifre, vb.) doğrudan kaynak kodunda saklanmasının güvenlik riskleri oluşturabileceğini belirtiyor.

**Çözüm**:

* Bağlantı dizesini appsettings.json dosyasına veya benzer bir yapılandırma dosyasına taşıyabilirsiniz. Böylece, kodunuzu kaynak kontrol sistemine (örneğin Git) eklerken hassas bilgileri dışarıda tutabilirsiniz.

**Örnek appsettings.json bağlantı dizesi**:

{

"ConnectionStrings": {

"DefaultConnection": "Host=localhost;Port=5432;Username=postgres;Password=123456;Database=obs\_db;"

}

}

**Kodda bağlantıyı kullanmak için**:

var connectionString = Configuration.GetConnectionString("DefaultConnection");

**2. Başarıyla Scaffold Edildi:**

* **Model ve DbContext Sınıfları**: Scaffold-DbContext komutunun başarıyla çalışması, veritabanınızdaki tabloların **Models** klasörüne ve **ObsDbContext** sınıfına dönüştürüldüğünü gösterir.
* Artık **Entity Framework**'ün **DbContext** sınıfı ve veritabanındaki her tabloyu temsil eden **model sınıflarınız** olacak. Bu sınıflar üzerinden veritabanı işlemleri yapabilirsiniz.

**3. Yapmanız Gereken Sonraki Adımlar:**

* **DbContext Kullanımı**: Şimdi **ObsDbContext** sınıfını kullanarak veritabanı işlemlerinizi (CRUD operasyonları, sorgular, vs.) gerçekleştirebilirsiniz.
* **Bağlantı Dizesini Yapılandırma**: Yukarıda belirtilen uyarıya dikkat ederek bağlantı dizesini güvenli bir şekilde yapılandırın.
* **Test Etme**: Veritabanı bağlanabilirliğini test etmek için basit bir sorgu yazabilir veya yeni bir veriyi ekleyerek işlem yaptığınızdan emin olabilirsiniz.

Örneğin, bir DbContext sınıfı ile veri ekleme işlemi şöyle yapılabilir:

using (var context = new ObsDbContext())

{

var ogrenci = new Ogrenci { Name = "Ali", Surname = "Yılmaz" };

context.Ogrenciler.Add(ogrenci);

context.SaveChanges();

}

Bu işlem, veritabanınıza yeni bir öğrenci kaydedecektir.

**Sonuç:**

* Veritabanı başarıyla DbContext ve model sınıflarına dönüştürülmüş durumda.
* Bağlantı dizesinin güvenliği için gerekli önlemleri almayı unutmayın.
* Artık **Entity Framework** ile veritabanı işlemlerine başlayabilirsiniz.