# **NOTLAR**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

* Sql Server bağlantısı yapılacağı zaman Context sınıfında override ettiğimiz bu metodu yazıyoruz. Localdb bizim serverımızın adı, BlogContext database’ imizin ismidir.
* Bir şey IEnumerable ise foreach ile dönülebilir, sayılabilir.
* Bir sorgu oluşturulurken Where kullanılırsa bu geriye IQueryable olarak döner.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

* Öncelikle AsEnumerable’ dır diye blog işaretlendi. Daha sonra bir Where sorgusu atılarak Titleları Blog-1 ya da Blog-2 içeren blogların getirilmesini istedik.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

* Uygulamayı ayağa kaldırdığımızda görüldüğü üzere Blog-1 ve Blog-2 geldi ancak Sql sorgusuna dikkat ettiğimizde burada giden sorguda herhangi bir Where parametresi yok. Burada böyle olmasının sebebi bizim kodlarımızda AsEnumarable yazmamız. AsEnumarable koyduk ve onu koyduğumuz anda database ile işimiz bitti. Yani db’ ye gidip bir daha sorgu atmadı.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

* AsEnumarable yerine AsQueryable yazdık. Eğer bir şey IQueryable ise Where sorgusu da gidecek. Yani biz bir şeyi IQueryable olarak bırakırsak onun sorgusu db tarafına gider.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

* Sorgumuz çalıştı Blog-1 ve Blog-2 geldi. Aynı zamanda görüldüğü üzere bu sefer Where parametremiz de Like ile geldi.
* Entity Framework Core tüm kayıtları, yapılanları izler. AsNoTracking kodunu koyarsak artık bu kayıtları izlememiş olur.
* Bizim bir blogumuz olsun ve bu bloglara yapılan yorumlar olsun. Ve istiyoruz ki ilgili blogla beraber yorumlar da gelsin. Böyle bir senaryoda farklı veri yükleme yöntemlerimiz var. Bunlardan herhangi biri kullanılarak veri yükleme işlemi gerçekleştirilebilir.
* Lazy Loading: Bu EF Core’ da default olarak kapalı. Aktif hale getirebilmek için Context sayfasında oluşturduğumuz UseSqlServer’ ın önüne, UseLazyLoadingProxies yazarak dahil ediyoruz ve artık kullanılabilir hale geliyor. Ve aynı zamanda aralarında bire çok vs. bağlantı oluşturduğumuz kod classlarına giderek bağlantılı kısımlara virtual kodunu ekliyoruz.

Örnek;

public virtual Blog Blog { get; set;}

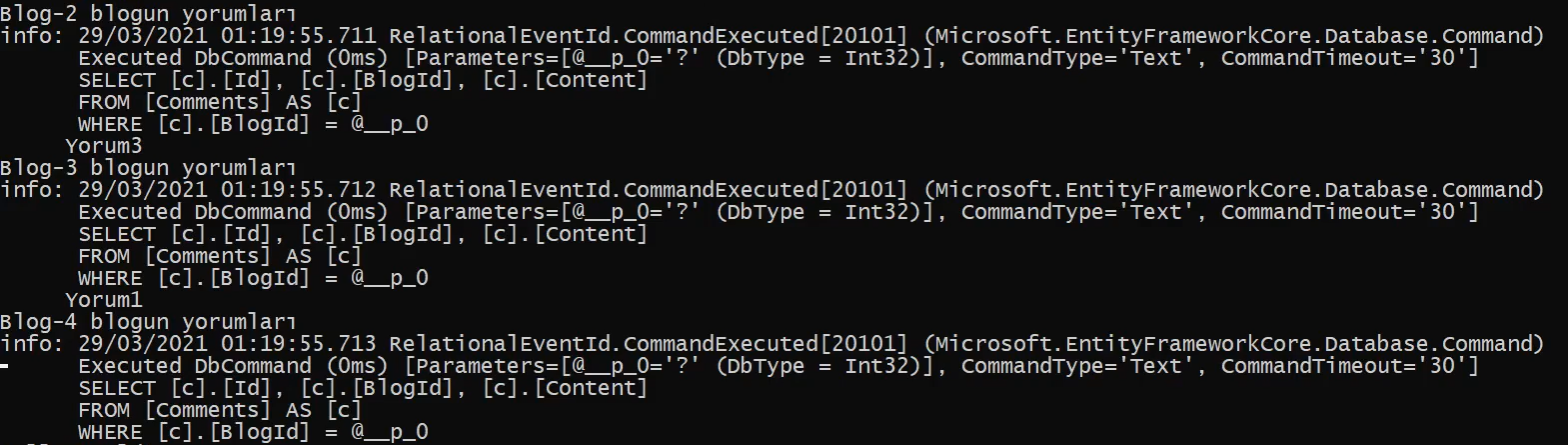
public virtual List <Comment> Comments { get; set;}

metin içeren bir resim

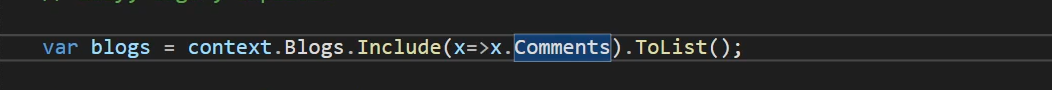
Açıklama otomatik olarak oluşturuldu

* Lazy Loading’ in sıkıntısı her bir istekte bulunduğumda database tarafına gider ve tekrardan bir sql sorgusu oluştururak çalıştırır.

Örnek;



* Eager Loading: En çok tercih edilen loaddır. Program.cs içerisinde Include yazılması bunun için yeterlidir. İstersek where ile şart da koyabiliriz.

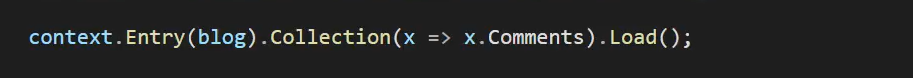


* Burada sadece tek bir sorguda komutları çekmiş olduk.

metin içeren bir resim

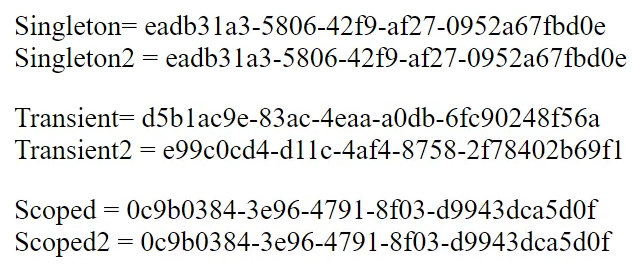
Açıklama otomatik olarak oluşturuldu

* Az öncekinin aksine burada sadece bir tane sorgu çalışarak tüm isteklerimizi yerine getirdi.
* Explicit Loading: Explicit loading olması için elimizde bir data olması lazım ve biz bu datanın navigation propertysine ulaşıp o propertyi load etmemiz gerekiyor. Bu loading yönteminde load metodu ne zaman gözükürse o zaman ilgili loading işlemini gerçekleştirir.
* Explicitte doğrudan ilgili entity üstünden hareket edemiyoruz. Bunun için ilgili contextin entrysinin üstünden hareket etmemiz gerekiyor. Ve burada collection metodundan yararlanıyoruz.

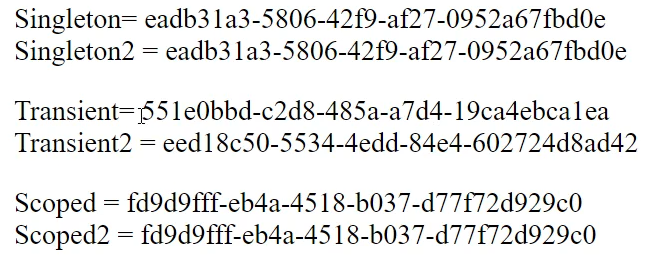


* Singleton: Ne kadar istek yaparsak yapalım hiç değişmiyor hep aynı değeri veriyor.
* Transient: Her bir çağrıda ikisi de farklı örnekler verir.
* Scoped: Her istekte değişir ama iki istek de aynı örneği verir.

Örnek;



* Sayfayı yenileyip tekrar inceleyince singleton değerleri değişmemiş, Transient her ikisi de değişmiş ve scoped ise değişmiş ama aynı örneği verecek şekilde değişmiş olduğunu gözlemliyoruz.



* Bir proje isimlendirilirken öncelikle firmamızın adı yazılır. Daha sonra projenin ismi yazılır. Eğer web uygulaması geliştiriliyorsa sonuna da .Web denilebilir veya masaüstü uygulaması ise .Desktop denilebilir.

Örnek;

Udemy.BankApp.Web

* DRY- Dont Repeat Yourself: Core’ da Controllerımızda sürekli üst üste elle giriş yaparak kendimizi uğraştırıyoruz. Bu kodlar bazen çok daha fazla olabiliyor. Kendimizi tekrar ediyoruz. Burada da devreye Repositories giriyor. Yani bize diyor ki sen farklı classlar oluştur, kümelere böl ve bunların içerisinde kodlarını yaz, her bir action metot içerisinde de tekrar tekrar yazmaktan kurtul. Bir noktada değiştir, bütün proje değişsin.

Örnek;

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu



* Bu şekilde bir Repositories adında klasör oluşturulurarak, altında işimize göre classlar oluşturup çalışabiliriz.