Any & Contains

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örneğimize göre idCollection içerisine, sonuca dahil etmek istediğimiz kategorilerin id değerlerini yazdık. Sonrasında Linq ile sorgularken, idCollection dizisi içerisindeki değer ile eşleşen kayıtları getirmek için Contains metoduna CategoryID bilgisini argüment olarak yazdık. Linq sorgusu çalıştığı zaman bize aşağıdaki gibi bir sonuç döndürecektir.

Biz sorgumuzu içeren olarak yazdık. Eğer tam tersi bir durumda, koleksiyon içerisindeki değerleri içermeyen yani "NOT IN" olarak yazmak için şu şekilde revize etmemiz yeterli olacaktır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Any metodunu kullanarak Contains ile yaptığımız işlemdeki aynı sonucunu elde edebiliriz. Her iki yöntem ile de aynı sonucu elde edebiliyoruz. Fakat Entity Framework' ün oluşturduğu sorgular, iki yöntem için de farklılık göstermektedir. Kullanım tercihinizi buna göre yapabilir, aynı zamanda SQL Management- Execution Plan ile performans ve güç tüketimlerini karşılaştırabilirsiniz.

Dapper

tablo içeren bir resim

Açıklama otomatik olarak oluşturulduAşağıdaki tabloda ORM araçlarının performansları sıralanmıştır. Genelde Orm araçları biraz yavaş çalışır. Tabloda gördüğümüz üzere, en çok kullandığımız ORM araçlarından biri olan Entity Framework ve LinqToSql performans sırasında en sonlarda yer almaktadır. Bu sebeple özellikle yoğun trafik alan e-ticaret projeleri gibi hız konusunda hassas projelerde bu ORM araçları pek tercih edilmez. Bu gibi durumlarda bir “Micro Orm” çözüm olabilir.

Yandaki tabloyu dikkate aldığımızda en çok göze çarpan Dapper’dır. Dapper Stackoverflow.com ekibi tarafından geliştirilmiş, ücretsiz ve açık kaynak kodlu bir C# kütüphanesidir.

Dapper Hafif (Lightweight) bir yapısı sayesinde mapping yapmak için arkada herhangi bir designer, konfigürasyon veya xml dosyası gibi şeylere ihtiyaç duymamaktadır. Bu yüzden daha basit bir kullanım sunmaktadırlar. Bunun yanında dapper ile direkt olarak katı sql sorguları yazıp çalıştırdığımız için sonuçlara çok daha hızlı ulaşırız ki bunun sonuçları yukarıdaki tabloda açık bir şekilde görülmektedir.

Burada aslında Sql sorgusunu ham hali ile yazmak geçmişte bizlere sıkıntı yaratan “Sql Injection” problemini tekrar gündeme getirebilir. Dapper’da buna önlem olarak aşağıda da örnekleri bulunan, dinamik parametre adında yapılar geliştirilmiştir. Sorgu içerisindeki parametreleri hand-coded olarak değil de dinamik olarak sorgunun içerisine gömmeyi sağlamaktadır. Buna ilaveten Dapper ile stored prosedürler ile çalışmakta mümkün.

Procedure

Bir stored procedure (yapısal yordam veya saklı yöntem) kısacası, hazırlamış olduğumuz T-SQL kodunu kaydedebildiğimiz SQL komutlarıdır. Böylece komutumuzu tekrar tekrar kullanılabilir hale getirebiliriz.Bu yüzden Stored Procedure yapısını, tekrar tekrar yazıp sıkça kullandığımız bir SQL sorgumuz varsa bu komutlarımızı Stored Procedure olarak kaydedip ardından, kodumuzu çalıştırmamız gerektiğinde Stored Procedure‘yi ismiyle çağırıp kullanırız.Ayrıca Stored Procedure yapısına parametrelerde verebiliriz. Böylece, Stored Procedure çağırdığımız yerde parametre değerlerine göre hareket edilebilir şekle sokabiliriz.

Stored Procedure, veri tabanı üzerinde saklandığından dolayı daha hızlı çalışırlar bu nedenle normal kodla yaptığımız işlemden daha iyi performans sağlar. SP’ler ilk çalıştıklarında derlenirler. Tekrar çalıştıklarında derlenmezler. Bu da performansa artı etki yapar.Son olarak, Stored Procedure kullanım şekliyle C# da sıklıkla kullandığımız metotlara benzetebiliriz. Kod tekrarını önlemek ve kodumuz istediğimiz yerde istediğimiz parametrelerle çalıştırmamıza yarar.