**IQueryable**

IQueryable veri tabanından verileri geniş kapsamlı sorgulamak için kullanılır. IQueryable arayüzü IEnumarable arayüzünü implement etmektedir. Bu sayede IQuaryable IEnumarable özelliklerinin hepsini barındıracaktır. IQueryable kullanıldığında sorgu alınırken öncelikle filtrelendirme yapılıp sorgu gönderilir.

1. Kod Örneği

public IList<TEntity> GetList(Expression<Func<TEntity, bool>> filter = null)

{

using (var context = new TContext())

{

return filter == null

? context.Set<TEntity>().ToList()

: context.Set<TEntity>().Where(filter).ToList();

}

}

2.Kod Örneği (IQueryable)

public IQueryable<TEntity> GetList(Expression<Func<TEntity, bool>> filter = null)

{

using (var context = new TContext())

{

var entities = context.Set<TEntity>().AsQueryable();

if (filter != null)

{

entities = entities.Where(filter);

}

return entities;

}

}

**SONUÇ:**

Yazdığımız metot ile IQueryable kullanarak oluşturduğumuz metot arasındaki fark, sorguların nasıl oluşturulup yürütüldüğü ve veri tabanına ne zaman gönderildiğidir. İşte farklar:

Yürütme Zamanı:

Yazdığımız GetList metodu, sorgunun sonuçlarını veri tabanından alıp hafızada bir liste olarak döndürür. Bu nedenle sorgu, veri tabanına gönderilip sonuç alındıktan sonra belleğe alınır ve işlem yapılır. Bu, büyük veri kümeniz olduğunda bellek tüketimi ve performans sorunlarına neden olabilir.

IQueryable kullanıldığında, sorgu sadece sorgu ifadesi oluşturulur, ancak veri tabanına gönderilmez. Sorgu sonuçlarına ihtiyaç duyulduğunda (örneğin .ToList() veya .FirstOrDefault() gibi bir sonuç alımı gerektiğinde), sorgu veri tabanına gönderilir ve sonuçlar geri alınır. Bu, yalnızca gerektiğinde veri tabanına erişilmesini sağlar ve bellek kullanımını azaltır.

Optimizasyon ve Filtreleme:

GetList metodu sorgunun sonucunu belleğe alır ve sonrasında filtreleme veya sıralama gibi işlemleri bellekteki veri üzerinde gerçekleştirir. Bu, işlem maliyetlerini ve bellek tüketimini artırabilir.

IQueryable kullanarak sorguları oluşturursanız, sorgu ifadesi üzerinde birçok operasyon gerçekleştirebilirsiniz. Bu, sorguyu daha iyi optimize etmenize ve yalnızca gerekli verileri çekmenize yardımcı olabilir. Örneğin, sadece belirli bir sayıda sonucu çekmek veya sadece belirli sütunları seçmek gibi.

Sorgu Esnekliği:

IQueryable ile sorguları oluşturmak, sorguları dinamik olarak filtrelemek ve sıralamak için daha fazla esneklik sağlar. Farklı filtreler veya sıralama kriterleri uygulayabilir ve bunları birleştirebilirsiniz.

Performans:

IQueryable ile çalışırken, sorgunun yürütme zamanı daha iyi optimize edilir, bu nedenle veri tabanından yalnızca gereken verileri alır ve bu, genellikle daha iyi performansa yol açar, özellikle büyük veri kümeleri ile çalışırken.

Sonuç olarak, IQueryable kullanarak sorguları oluşturmak, veri tabanı erişimi üzerinde daha fazla kontrol sağlar ve veri tabanından yalnızca gereken verilerin çekilmesini kolaylaştırır. Bu, performans ve bellek kullanımı açısından avantajlar sunar.