EfCore Tekrar Notları

1. Tekil veri getiren sorgulama fonksıyonları
   1. SingleAsync: Eğer ki, sorgu neticesinde birden fazla veri gelıyorsa yada hiç gelmiyorsa her iki durumdada exception fırlatır
   2. SingleOrDefaultAsync: Eğerki sorgu neticesinde birden fazla veri geliyorsa exception fırlatır ,hiç veri gelmiyorsa null döner.
   3. FirstAsync: Sorgu neticesinde elde edilen verilerden ilkini getirir.Eğerki hiç veri gelmiyorsa hata fırlatır
   4. FirstOrDefaultAsync: Sorgu netıcesınde elde edilen verilerden ilkini getirir egerkı hıc verı gelmıyorsa null değerini dondurur
   5. FindAsync: Find fonksıyonu prymerıy key kolonuna ozel hılı bır sekılde sorgulama yapmamızı sagalar
   6. LastAsync, LastOrdDefaultAsync: orderby kullanılmalıdır, sorgu netıcesınde gelen verilerden en sonuncusunu getırır hıc verı gelmıyorsa hata fırlatır
2. Diğer sorgulama Fonksıyonları
   1. CountAsync: olusturulan sorgunun execute edılmesı netıcesınde kac adet satırın elde edileceği sayısal olarak (int) bızlere bıldıren fonktur
      1. var kişiler20 =(await contex.kişis.ToListAsync()).Count();//burası kac kısının oldugunu gosterır ama malıyetlıdır burası
      2. var kişiler21 =await contex.kişis.CountAsync(); //burada daha az bır malıyet yapmıs olduk uste ılk basta hepsını sorgulatıp sayıyorduk
   2. LongCountAsync: aynı sekılde uste kı gıbı burada daha buyuk datalar ıcın kullanılır
   3. AnyAsync: sorgu netıcesınde verının gelıp gelmedıgını bool turunde donen fonkdur "Varmı Yokmu"
   4. AllAsync: Bir sorgu netıcesındekı gelen verilerin verilen şarta uyup uymadığını kontrol etmektır. Eğerki tüm veriler şarta uyuyorsa true,uymuyorsa false döndürecektir
   5. SumAsync: Vermiş olduğumuz Sayısal propertynin toplamını alır
   6. AverageAsync: vermiş olduğumuz sayısal propertynin aritmatik ortalamasını alır
   7. Contains: Like '&...&' sorgusu olusturmaızı sagalr
   8. StartsWith: burada ...% olanları yapar
   9. EndssWith: burada %... olanları yapar
      1. Kullanımları: var kişiler30 = contex.kişis.Where(u => u.Email.EndsWith("a")); gibi
   10. ToDictionaryAsync: var kişiler31 =await contex.kişis.ToDictionaryAsync(u=>u.Şifre,u=>u.Email);
       1. toList ile aynı amaca hizmet etmektedir yanı olusturulan sorguyu execude edıp neticesinde alırlar
       2. ToDictionary: Gelen s orgu netıcesını Dictionery bır koleksıyona dondurecektır .---> Dictionery key ve value olarak kaydetmemize yarayan bir sınıfdır.
   11. Select: Select fonk işlevsel olarak bırden fazla davranısı sozkonusudur ---->soradan execude etmek gerekır
       1. 1.Select fonk Generate edılecek sorgunun çekılecek kolonlarını ayarlamamızı saglamaktadır
       2. 2. Select fonk Gelen verileri farklı türde karşılamamızı sağlar.T , anonım
   12. SelectMany: Selec ile aynı amaca hızmet eder lakin ilişkisel tablolar neticesinde gelen koleksıyonel verılerı de inner join yapılandırması olusturur burası
   13. Groupby Fonksiyonu: Guruplama yapmamızı saglıyan fonktur GroupBy(i => i.Id)
3. Veri silme durumları
   1. Cascade: esas tablodan prıncıpal dan sılınen verı ıle karsı bagımlı tablodan bulunan ılısklı verılerın sılınmesını saglar default olarak efcore bunu kullanır
   2. SettNull: esas tablodan prıncıpal dan sılınen verı ıle karsı bagımlı tablodan bulunan ılısklı verılere null degerın atanmasını saglar
      1. bıre bırde senaryolarda egerkı foren key ve prımary key kolonları aynı ıse setnull davranısını kullanamayız null alamaz burada pk kolonu ---->forenkey kolonunu null yapıyor ztn
      2. bunu kullanırken ılgılı foreın keyı IsRequired(false) yapmamız gerekir değilse hata alıcaz database mıgrat ederken buna dıkkat edelım silme islemı yaparkende bıre e cok olandan sılme ıslemı yapmayı tercıh edelım coka bır de hata alınıyor arada
   3. Configurations | Data Annotations & Fluent API:
      1. ToTable: Generate edilecek tablonun ismini belirlememizi sağlayan yapılandırmadır.
      2. Column - HasColumnName, HasColumnType, HasColumnOrder
      3. HasForeignKey: lişkisel tablo tasarımlarında, bağımlı tabloda esas tabloya karşılık gelecek verilerin tutulduğu kolonu foreign key olarak temsil etmekteyiz.
      4. Ignore: Bazen bizler entity sınıfları içerisinde tabloda bir kolona karşılık gelmeyen propertyler tanımlamak mecburiyetinde kalabiliriz.
      5. HasKey: EF Core'da, default convention olarak bir entity'nin içerisinde Id, ID, EntityId, EntityID vs. şeklinde tanımlanan tüm proeprtylere varsayılan olarak primary key constraint uygulanır.
      6. IsRequired: Bir kolonun nullable ya da not null olup olmamasını bu konfigürasyonla belirleyebiliriz. -->null olamaz
      7. StringLength – HasMaxLength: Bir kolonun max karakter sayısını belirlememizi sağlar.
      8. HasPrecision: Küsüratlı sayılarda bir kesinlik belirtmemizi ve noktanın hanesini bildirmemizi sağlayan bir yapılandırmadır.
      9. IsUnicode: Kolon içerisinde unicode karakterler kullanılacaksa bu yapılandırmadan istifade edilebilir.
      10. HasComment: EF Core üzerinden oluşturulmuş olan veritabanı nesneleri üzerinde bir açıkalama/yorum yapmak istiyorsanız Comment'i kullanblirsiniz.
      11. InverseProperty: İki entity arasında birden fazla ilişki varsa eğer bu ilişkilerin hangi navigation property üzerinden oılacağını ayarlamamızı sağlayan bir konfigrasyondur.
      12. Composite Key: Tablolarda birden fazla kolonu kümülatif olarak primary key yapmak istiyorsak buna composite key denir.
      13. HasDefaultSchema: EF Core üzerinden inşa edilen herhangi bir veritabanı nesnesi default olarak dbo şemasına sahiptir. Bunu özelleştirebilmek için kullanılan bir yapılandırmadır.
      14. HasDefaultValue: Tablodaki herhangi bir kolonun değer gönderilmediği durumlarda default olarak hangi değeri alacağını belirler.
      15. HasDefaultValueSql: Tablodaki herhangi bir kolonun değer gönderilmediği durumlarda default olarak hangi sql cümleciğinden değeri alacağını belirler.
      16. HasComputedColumnSql: ablolarda birden fazla kolondaki veirleri işleyerek değerini oluşturan kolonlara Computed Column denmektedir
      17. HasConstraintName: EF Core üzerinden oluşturulkan constraint'lere default isim yerine özelleştirilmiş bir isim verebilmek için kullanılan yapılandırmadır.+
      18. HasData: Sonraki derslerimizde Seed Data isimli bir konuyu incleyeceğiz. Bu konuda migrate sürecinde veritabanını inşa ederken bir yandan da yazılım üzerinden hazır veriler oluşturmak istiyorsak eğer buunun yöntemini usulünü inceliyor olacağız.
      19. HasDiscriminator: İleride entityler arasında kalıtımsal ilişkilerin olduğu TPT ve TPH isminde konuları inceliyor olacağız. İşte bu konularla ilgili yapılandırmalarımız HasDiscriminator ve HasValue fonksiyonlarıdır.
      20. HasField: Backing Field özelliğini kullanmamızı sağlayan bir yapılandırmadır.
      21. HasNoKey: Normal şartlarda EF Core'da tüm entitylerin bir PK kolonu olmak zorundadır. Eğer ki entity'de pk kolonu olmayacaksa bunun bildirilmesi gerekmektedir! İşte bunun için kullanuılan fonksiyondur.
      22. HasIndex: Sonraki derslerimizde EF Core üzerinden Index yapılanmasını detaylıca inceliyor olacağız.
      23. HasQueryFilter: İleride göreceğimiz Global QUery Filter başlıklı dersimizin yapılandırmasıdır.
   4. IsUnique:
4. IsUnique yöntemi, bir EntityTypeBuilder nesnesi için çağrılabilir ve bir sütunun benzersiz olması gerektiğini belirler. Bu yöntem, veritabanındaki ilgili sütunun benzersiz olmasını sağlayan bir benzersiz kısıtı ekler.
5. protected override void OnModelCreating(ModelBuilder modelBuilder)
6. { modelBuilder.Entity<Person>()
7. .HasIndex(p => p.Email)
8. .IsUnique();
9. }
   1. Yukarıdakı olay prımary key ıle karıştırılmamalıdır cunku pk farklı bır olay mesela ıd,isim,soyisim beraber alırsanız ıd:1isim:ahmet:soyisim:taha, ıd:1isim:ahmet:soyisim:osman ıkısınıde kabul eder toplu bır sekılde kontrol eder
   2. Bir tabloda yalnızca bir adet birincil anahtar (primary key) olabilir. Birincil anahtar, bir tablodaki her kaydı benzersiz bir şekilde tanımlayan bir veya daha fazla sütunun bir kombinasyonudur.
   3. Include: Yani üretilen bir sorguya diğer ilişkisel tabloların dahil edilmesini sağlayan bir işleve sahiptir..
   4. ThenInclude: ThenInclude, üretilen sorguda Include edilen tabloların ilişkili olduğu diğer tablolarıda sorguya ekleyebilmek için kullanılan bir fonksiyondur. Tekillerde bu yapıla bılır cogullarda hata oluşur
   5. Eager Loading İçin Kritik Bir Bilgi: EF Core, önceden üretilmiş ve execute edilerek verileri belleğe alınmış olan sorguların verileri, sonraki sorgularda KULLANIR!
   6. AutoInclude - EF Core 6: Uygulama seviyesinde bir entitye karşılık yapılan tüm sorgulamalarda "kesinlikle" bir tabloya Include işlemi gerçekleştirlecekse eğer bunu her bir sorgu için tek tek yapmaktansa merkezi bir hale getirmemizi sağlayan özelliktir.

modelBuilder.Entity<Employee>()

.Navigation(e => e.Region)

.AutoInclude();

* 1. Lazy Loading: pek kullanmıycaz malıyetlı bır teknolojıdırı eger kullanıcaksak ilk oncelıkler nugetlerden:Microsoft.EntityFrameworkCore.Proxies indirmemiz gerekir
     1. Sonradan navıgatıon propertylerı vırtual olarak ısaretlenmesı gerekır bunun yanında clasların publıc olması gerekır