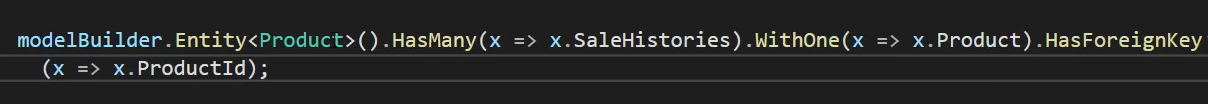
# **NOTLAR**

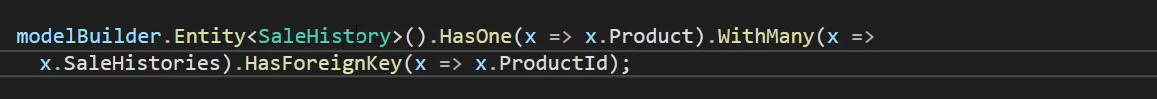
* Data Annotations aracılığıyla bir tablonun şema bilgisini, tablo adını ya da bir kolonun tipini, kolon adını değiştirebiliriz.
* Aynı şey Fluent Api ile de yapılabilir. Kod context sayfasının içerisine yazılır.

metin içeren bir resim

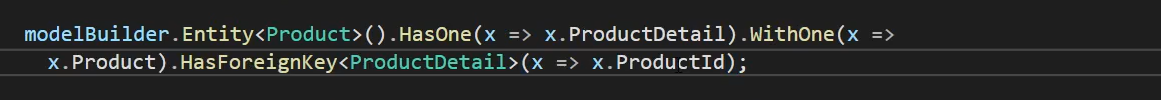
Açıklama otomatik olarak oluşturuldu

* Fluent Api her zaman Data Annotations’ ı ezer. Yani Fluent Api’ nin dediği olur.
* Fluent Api ile yapılması daha uygundur. Çünkü Data Annotations ile yapıldığı zaman Single Responsibility kuralı çiğnenmiş olur.
* Is Unique kullanarak bir verinin tekrar girilmesini engelleriz. Mesela bir ürün ismini telefon olarak girdiysek bir daha telefon isimli bir ürün giremeyiz.
* Bir olan Parent, çok olan Child’ dır.
* Child Dependent Entity, Parent ise Principal Entity olur.
* One To Many bire çok ilişki demektir. Yani bir kadının birden fazla çocuğu olabilir ama bir çocuğun sadece bir annesi olur. Burdaki One anne, Many ise çocuklardır.
* One To Many’ nin Fluent Api ile örnek gösterimi.





* İki gösterim de doğrudur. İlk görselde Satış çok olabileceği için HasMany metoduyla yazılmış ve bir ürün üzerinden olduğu için WithOne denilmiştir.
* İkinci görselde ise tam tersi başlanmış, bir ürünün birden çok satışı olmasından dolayı HasOne metoduyla yazılmış ve çok satış olduğu için WithMany denilmiştir.
* One To One bire bir ilişki demektir. Mesela resmi olarak bir kadın sadece bir adam ile evlenebilir ve bir adam da sadece bir kadın ile evlenebilir.

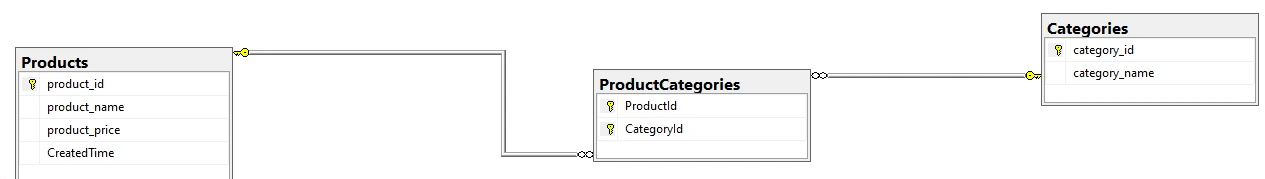


* Buradaki örneğimizde bir ürünün sadece bir açıklaması var ve bir açıklama da sadece bir ürüne bağlı olduğu için bu kodu yazdık.
* Many To Many anlamı çoğa çok demektir. Ancak çoğa çok ilişki diye bir bulunmamaktadır ve bu ilişki bire çok ilişkiyle oluşturulmaktadır. Örnek verecek olursak bir ürünün birden fazla kategorisi olabilir, bir kategoride de birden fazla ürün olabilir.

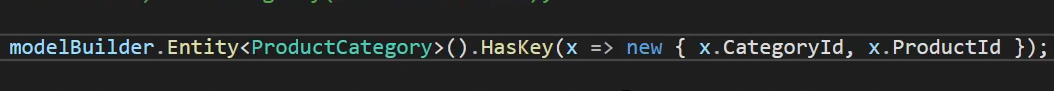
metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

* Bunu oluşturmak için iki adet bire çok ilişki yazıyoruz.



* Bu şekilde bir ürünün birden fazla kategorisi olabilir ve bir kategoride birden fazla ürün olabilir diyen bir database oluşturduk.
* Biz tablomuza ekleme yapacağımız zaman 1 id’ sine sahip bir ürünün 1 id’ li bir kategoride iki defa tekrarlı kayıt yapmasını istemiyorsak ikisini aynı anda primary key olarak setleyebiliriz.



* Table Per Hierarchy, Solid Prensiplerindeki Liskov Substitution kullanılarak yapılır. Mesela full time ve parttime çalışanlarımız olursa bunu veritabanında discriminator başlığı altında otomatik olarak hangi çalışanın full time veya part time olduğunu belirtir.
* Table Per Type, Table Per Hierarchy ile aynı mantıkta çalışır ancak burada discriminator kolonu bulunmaz. Ayrı ayrı full time part time tabloları oluşturarak içerisine o değerleri girer.