**SaveChangeAsync Interceptor**

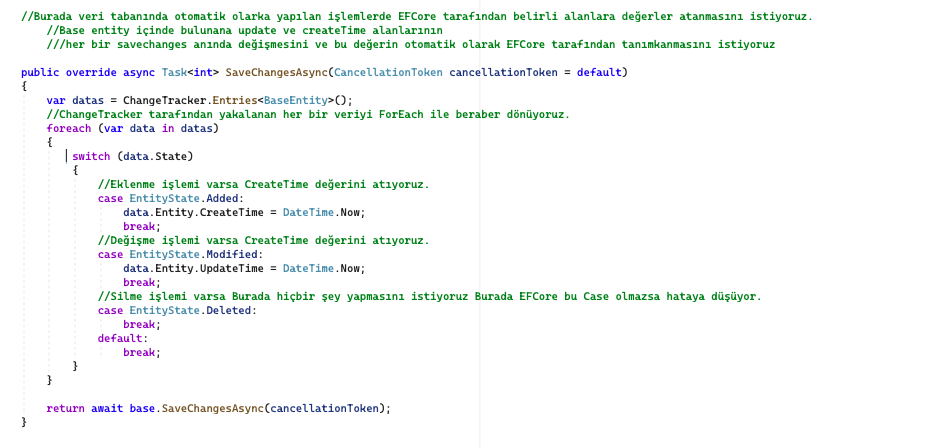
Burada amacımız bizim Entitylerimiz içinde bulunan BaseEntity’den gelen CreateTime ve UpdateTime değerlerinin tamamının bizim tarafımızdan değil o an o işlem olduğu anda gerçekleşmesini sağlamak.

metin, ekran görüntüsü, yazı tipi, tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Görselde görüldüğü gibi bir veri insert edilirken bir T anında bu değerin oluşturulmasını isteyebiliriz. Örnek olarak bizim projemizde bulunan CreateTime ve değerlerini verebiliriz.  
  
Şimdi BaseEntity dosyamızı düzenleyelim.

|  |
| --- |
| using System;  namespace OnionArch.Domain.Entities.Common  {  public class BaseEntity  {  public Guid ID { get; set; }  public DateTime CreateTime { get; set; }  public DateTime UpdateTime { get; set; }  }  } |

Şimdi amacımız her bir veri ekleme ve düzenleme işleminde veri tabanında bulunan CreateTime ve UpdateTime alanlarını EFCore tarafından otomatik olarak eklenecek düzenlenecek ve güncellenecek hale getirmek.  
  
Bunun için yapmamız gereken ,   
**OnionArch.Persistance** altında bulunan Context yapımızda birkaç adet Configürasyon yapmamız gerekmekte bunun için ilk olarak SaveChangesAsync methodunu override yaparak bu işlemi gerçekleştirebiliriz.  
  


Şimdi artık ekleme çıkarma işlemlerini yaparken hiçbir şekilde kullanıcının yada geliştiricinin bu değerleri atamasına gerek yok .   
  
Şimdi yeni bir veri oluşturma ve bir veriyi düzenleme işlemlerimize geçiş yapalım.

Öncelikle OnionArch.WebApi katmanına geçiş yapalım yani bizim Presentation katmanımıza.

Burada Controller yapımızın içinde Update ve Create işlemleri için geliştirdiğimiz kodlarımıza tekrar göz gezdirelim.  
  
metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Görselde göreceğimiz üzerine biz giriş değerleri için Product Entity’sini direkt olarak Swager içinde tanımladık.  
Swager içinde hiçbir zaman Gelecek olan tip Entityl olarak verilmez !

Bunun için ViewModel denen kavram kullanılır .

**ViewModel**

OOP yapılanmasında bir modelin kullanıcıyla etkileşimi neticesinde kullanılan/doldurulan memberlarını temsil eden ve süreçte ilgili model yerine kullanılan/taşınan/transfer edilen/post edilen vs. bir nesnedir.  
  
Yani daha basit bir anlatımla EFCore’da bulunan Entitylerde kullanıcı yada geliştiricilerin Veri Tabanında müdahale edeceği alanları içeren ve bu Objeden oluşmuş verileri hedef objeye çevirirken geliştiriciler için velinimet bir yapıdır.

OnionArch.Application Katmanımıza geçelim.

* Bir adet klasör oluşturalım klasörümüzün adını View\_Models olarak adlandıralım.
* Daha sonra projemizde bulunan bütün Entityler için burada Modeller oluşturalım örnek olarak biz burada Product Entity değerimize göre bir ViewModel oluşturalım.
* İhtiyaca göre Create ve Update işlemlerinde farklı farklı senaryolar olabileceğinden dolayı burada farklı farklı işlemler için farklı farklı ViewModeller oluşturmak her zaman daha mantıklıdır. (Best Practice açısından)

metin, yazı tipi, çizgi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu **metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Şimdi ViewModelimizi oluşturduğumuza göre Create işlemi ile burada örneklerimize göz atalım.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Update fonksiyonumuzu geliştirelim.

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu