Geliştirilecek uygulama hakkında bir bilgilendirme yapılacaktır.

ekran görüntüsü, metin, tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Üstteki diagram uygulama hakkında temek öğeleri içermektedir. User ve roller arasında çoka çok bir ilişki var ve user id role id bir primary key. Kullanıcılar ve başvurular arasında da çoka çok bir ilişki mevcuttur. User lar member ise iş ilanlarını görüntüleyebilme ve başvurabilme durumu mevcuttur. Admin yetkisi ise başvurular oluşturabilmektedir.

Advertiesment : İlanlar

AdvertiesmentUser : kullanıcıların başvurdukları ilanlar.

Userlar : kullanıcıları temsil etmektedir.

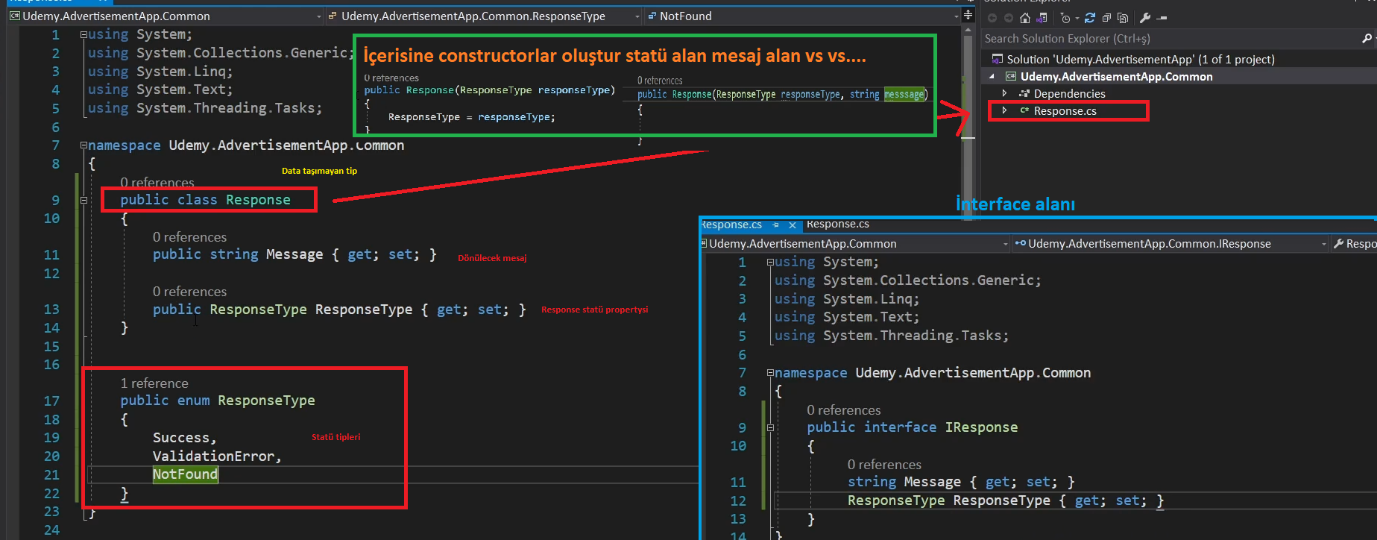
Projeleri oluşturmaya başlayalım. İlk olarak common katmanı oluşturulacaktır. Bu projemizde Core katmanına karşılık gelir. Framework katmanıdır. Extentions ları, utilities leri, temel kütüphaneleri burası barındırmaktadır. Durum neticesinde kendimiz burada business içerisinde kullanabilmemz için bir result sınıfı oluşturmamız gerekecektir.

* Bir enum alacak (iş statusu belirlenecek başarılı, hatalı, bulunamadı …)
* Bunun yanında bir Result sınıfı ve interface i (duruma göre mesaj alabilen ve statünün zorunlu olduğu)
* Data taşıyabilen (getirme giib işlemlerde generic çalışacak olan bir yapı) ya ihtiyaç duyulacaktır.

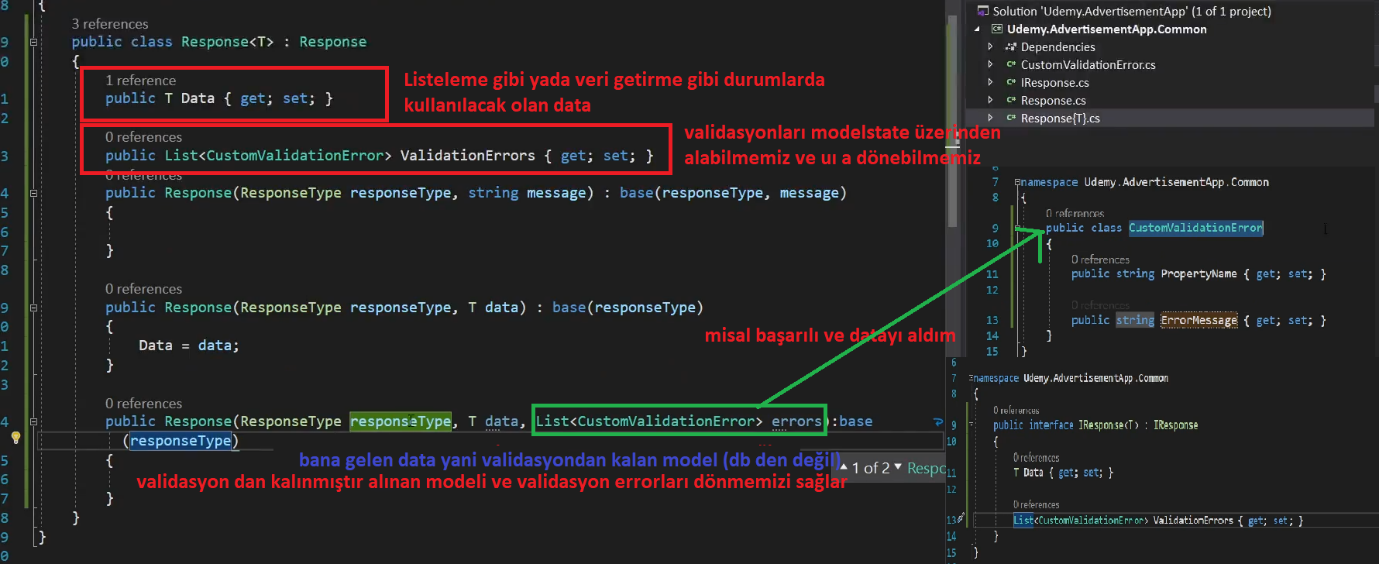
Çalışılan şirket adı + uygulama adı -> Solution isimlendirmesi.

Çalışılan şirket adı + uygulama adı + hizmet adı -> Katman isimleri.

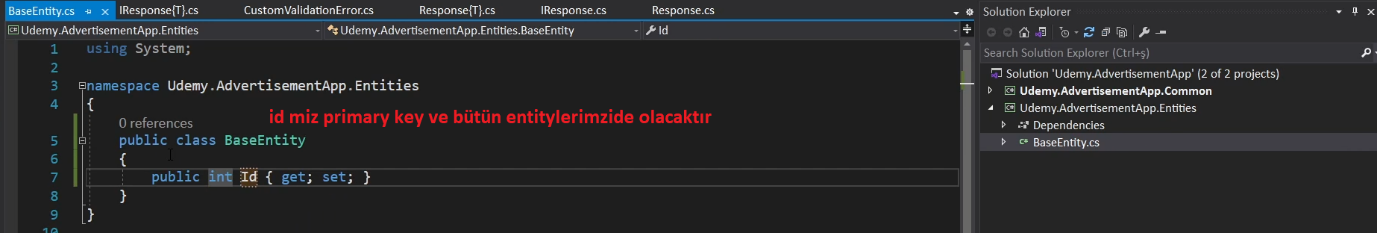
* İlk olarak solution oluşturumaktadır. Uygulamamız genel hatları ile birlikte iş ilan listeleme, başvurma gibi işlemler yapılacağı bir sayfadır. Respone yapısını olduğu gibi Common (shared - core) katmanı içerisinde konumlandırılmaktadır.



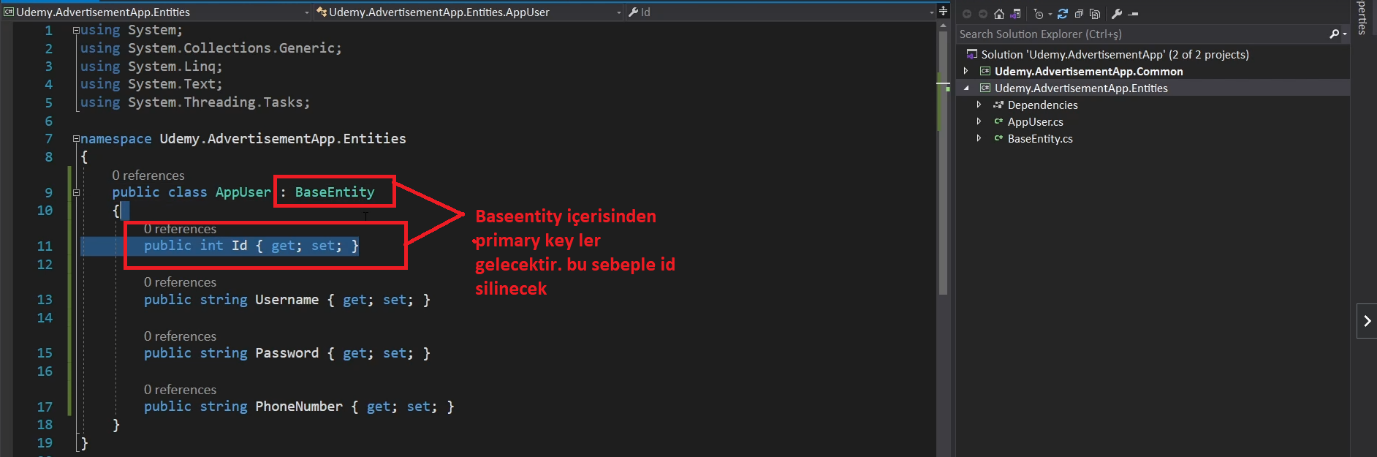
* Consturcotlar içerisinde alınan propertyler setlenenecktir. İnterface ten de kalıtım alınması gerektiği unutulmasın. Her consturcotr da type belirlenmesi gerekiyor. Bunu zorunlu tutabilmek için ve oluşturma tiplerini belirleyerek propertylerini setleyebilmekj için uğraşıyoruz. Constuctor içerisinde alınan değerler propertylere setlenmektedir. Messgae hata yada başarılı dönmek içinde kullanılabilir.
* (işlem gerçekleştirilemsi durumunda message oluşturabilmek için kullanılabilmesi yada hata durumunda mesaj dönülebilmesi için message oluşturduk. ) işlem bu kadar.
* Aynısının data taşıyan versiyonu ise aşağıda mevcuttur. (validaton errorlarımı koyabilmemz için bir complextype oluşturalım fluent validasyondan gelecek olan mesajları liste şeklinde property ve message olarak dönebilmek için. )
* Generic Response larımız aslında birer response tur. Base deki consturcor lar ile örtüşecek şekilde consturtorlarınızı oluşturun.
* Validaton errorlarımız için bir complex type oluşturarak property ve validaton message bazında ilgili validasyon resultlarımızı taşıyalım.



* Görüldüğü üzere Result sınıfımız bu şekilde interfaceleri modelleri ile birlikte gerçekleştirilmiştir. Ben projemde ayriyetten success olma durumu validasyon hatası durumu gibi durumlar için ayriyetten bu base olan response ve generic response sınıflarımdan kalıtım aldığım farklı özel (costume sınıflarda oluşturdum)
* Entity katmanı kodlamasına bir başlayalım. (bir önceki bölümde Response<T> interface inden implemente edilmemiş bunu ekleyin) Hatırlanacağı üzere bizler entitylerimizi oluştururken yine common katmanında (shared – core ) sistemde kullandığımız sınıfların entity olduklarını, dto ların dto olduklarını ve kurallar geçebilmek için ientity, baseentity, ıdto ve basedto gibi sınıflar ve interfaceler oluşturmamız gerekmekteydi. Core katmanımız framework olarak nitelendirilir ve ilgili sınıflarımızı – interfacelerimizi buralara tanımlayabiliriz. Core içerisinde tüm projelerde kullanılabilecek entityler (cinsiyet, role, base role, user gibi tablolar) kodlanabilmektedir. Ek bilgi olarak burada eklemek istedim.
* İsimlendirme kuralı şirket+proje+katmanadı şeklinde olacaktır.
* Generic repository içerisinde find methodu bulunmaktaydı. Bu find methodu bir primary key almaktaydı. User – role arasındaki ilişki tekrara düşmemesi için ikili olarak primary key setlendi ama tabloya ayriyetten bir primary key daha eklenmelidiri Find methodunun işleyebilmesi için.
* Base entitylerimizde tüm entitylerde bulunması gereken properytleri ekleyebiliriz (isactive, updateddate, createddate, primary key …) buradan da tüm entitylerimizin bu yapıdan kalıtım alması gerekecektir. (generic find methodu ile tüm entitylerdeki primary key e göre iş yapabilmesi için baseentity içerisinde ilgili propertymizi yazalım. (userroles ün ayrı bir primary keyi olmalı ve userid- role id yine birlikte unique olmalıdır. Çoka çok tablolarda aynı olayın gerçekleşmesi gerekecektir. Advertiesment user için de geçerli bu durum aynı kişiye aynı rol iki defa verilemez yada başvurusu olan birisi ilgili işe bi daha başvuramaz.))



* Bu aşamadan sonra tüm entitylerimiz tek tek kodlanmaya başlanabilir ve baseentity den kalıtım alması sağlanabilir.



* Benzer şekilde diğer entitylerimiz oluşturulabilir.

