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.

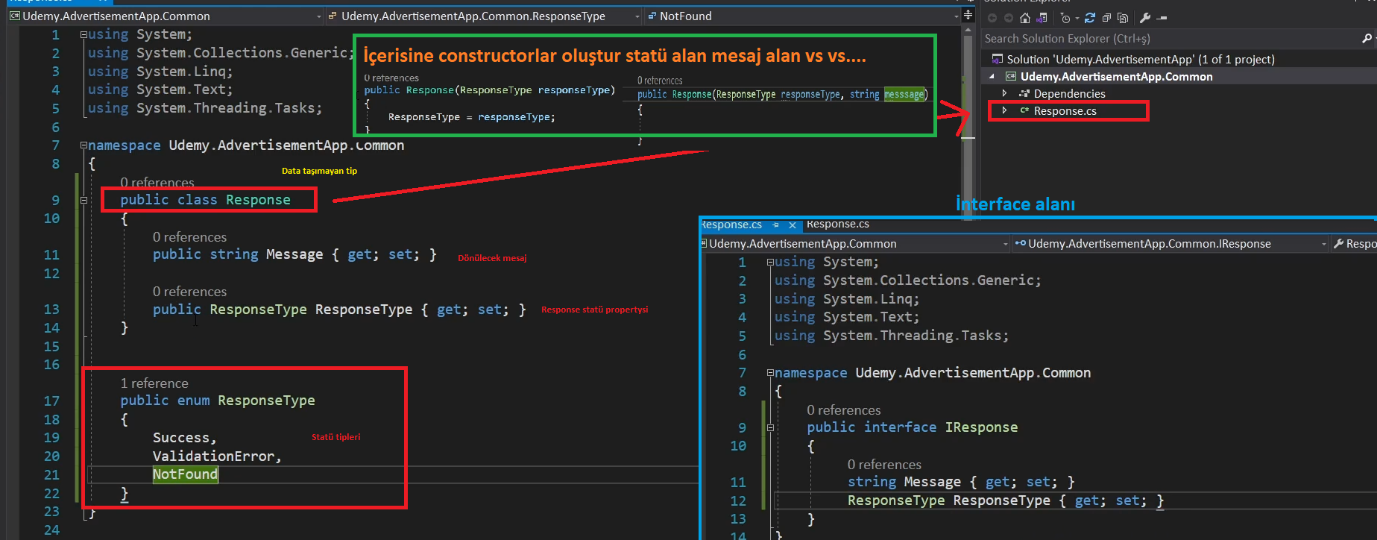
Projeleri oluşturmaya başlayalım. İlk olarak common katmanı oluşturulacaktır. Bu projemizde Core katmanına karşılık gelir. Framework katmanıdır. Extentions ları, utilities leri, temel kütüphaneleri burası barındırmaktadır. Durum neticesinde kendimiz burada business içerisinde kullanabilmemz için bir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

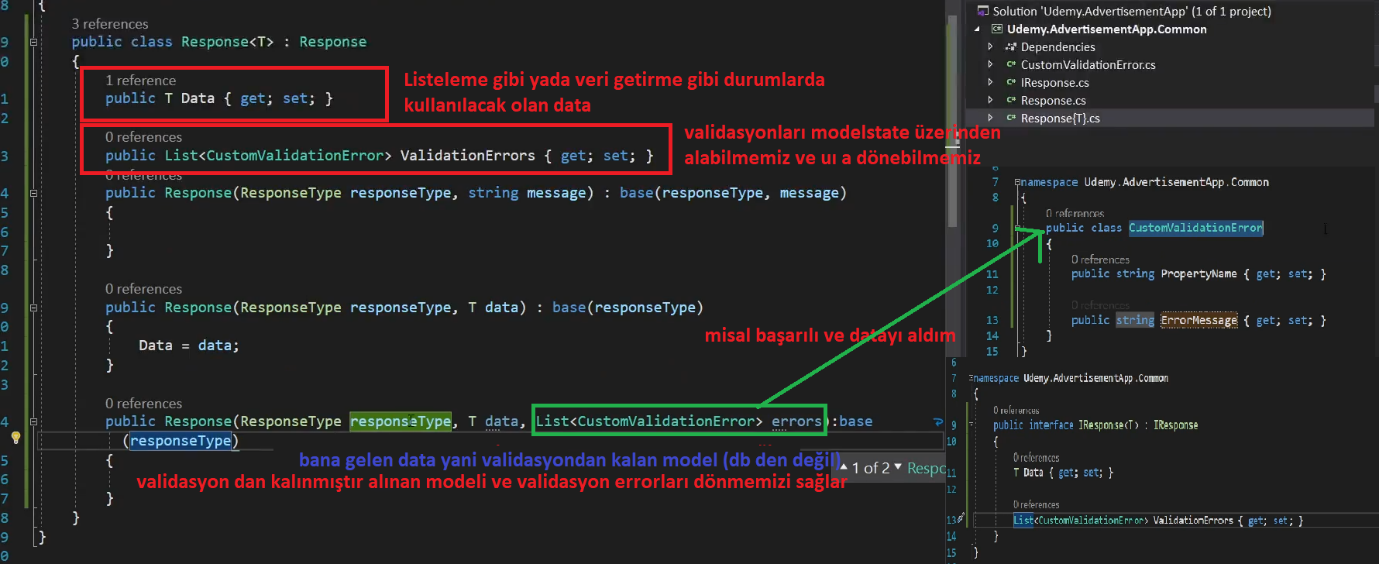
Çalışılan şirket adı + uygulama adı -> Solution isimlendirmesi.

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

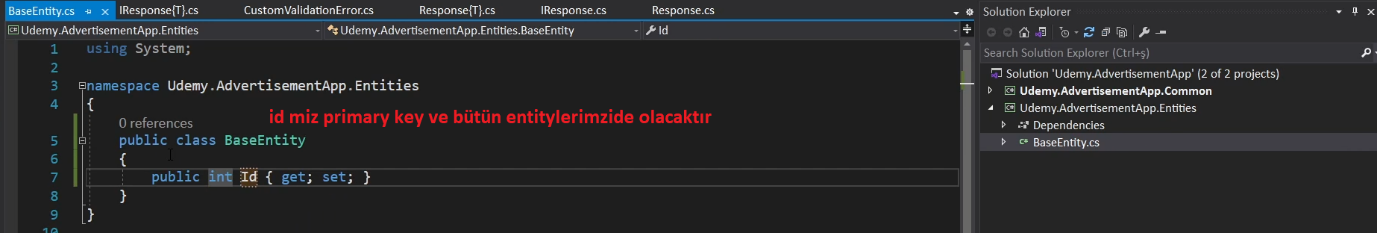
* İlk olarak solution oluşturumaktadır. Uygulamamız genel hatları ile birlikte iş ilan listeleme, başvurma gibi işlemler yapılacağı bir sayfadır. Respone yapısını olduğu gibi Common (shared - core) katmanı içerisinde konumlandırılmaktadır.



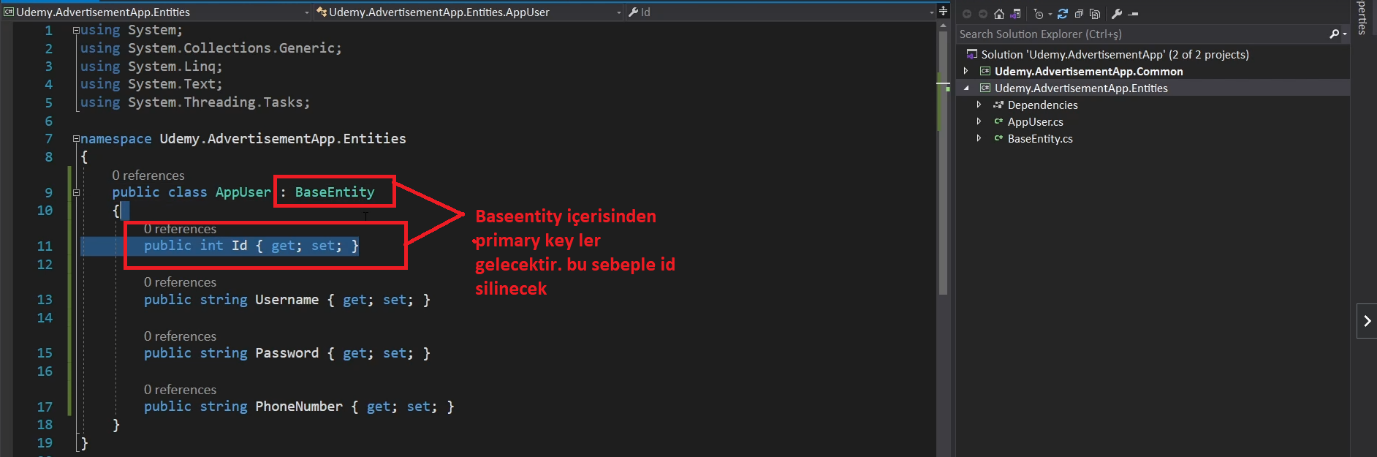
* Consturcotlar içerisinde alınan propertyler setlenenecktir. İnterface ten de kalıtım alınması gerektiği unutulmasın. Her consturcotr da type belirlenmesi gerekiyor. Bunu zorunlu tutabilmek için ve oluşturma tiplerini belirleyerek propertylerini setleyebilmekj için uğraşıyoruz. Constuctor içerisinde 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.



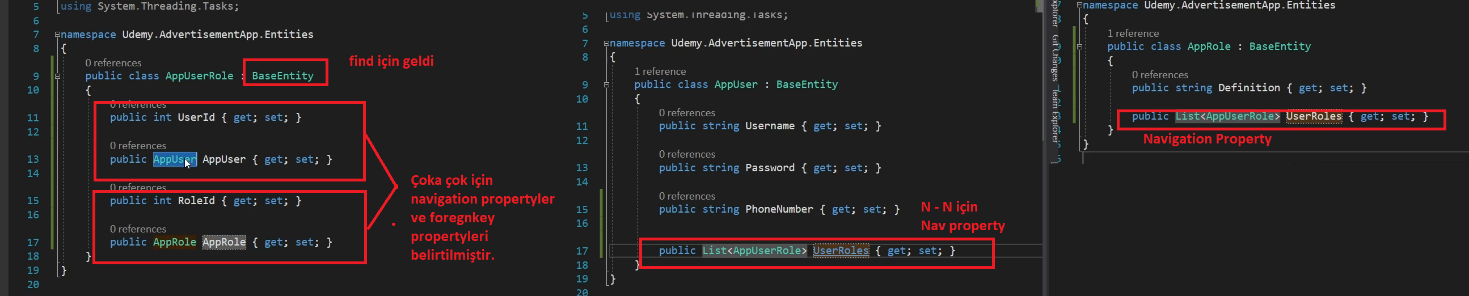
* Görüldüğü üzere Result sınıfımız bu şekilde interfaceleri modelleri ile birlikte gerçekleştirilmiştir. Ben projemde ayriyetten success olma durumu validasyon hatası durumu gibi durumlar için ayriyetten bu base olan response ve generic response sınıflarımdan kalıtım aldığım farklı özel (costume sınıflarda 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.



* Tüm entityler Base Entity sınıfından kalıtım alacaklardır.
* Bu başlık altında bazı entitylerden bahsetmek istiyorum.

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

* Hatırlanacağı üzere **tüm entitylerimizin find methodu** ile çalışabilmeleri için **BaseEntity** sınıfından kalıtım alması gerekmiyordu. Buna ek olarak aşağıdaki maddeler de eklenecektir. (find içerisinde id ile arama yapacağız )
* Hem çoka çok ilişki ve hem bir takım business lar içermektedir bu tablo yani iş ilanı ve user arası çoka çok ilişki mevcut ve işe başvurular bu tabloya kaydedilmektedir. **Buraya atılan her bir kayıt ilgili user in bu tabloya başvuru yapması manasına gelmektedir.** İlgili bazı alanlar mevcuttur status : ilgili başvurunun durumunu temsil eder. Başvuru bir takım süreçlerden geçecektir. Değerlendirilecek, mülakata çağrılacak vs gibi işlemlerden geçirilecektir. Bu aşamada bir Lookuptable a ihtiyacmızı var
* **LookUpTable** : cinsiyet, iş statüsü gibi durumlar için statu bilgilerinin yazıldığı master tablodur. (birden fazla statü durumu olacaktır.) Parametre tablosu olarak düşünün bu id ler başka tabloda değer olarak tutulur. Genellikle id ve definition alanları tutulmaktadır. Cinsiyet, askerlik durumu ve iş durumu LookUpTable şeklinde tutulacaktır. (look up table aslında cinsiyet tablosu gibi birden fazla seçeneğin olduğu durumda normalizasyon sağlanabilmesi için (ikiden fazla) referans vermek amaçlı kullanılmaktadır. ) (askerlik yapıldı muaf tecillii gibi çeşitli seçenekler mevcuttur. )
* Kısaca özetlemek gerekirse ilgili başvurunun durumu olan (birtakım süreç barındıran) status alanı (reddedildi, mülakata çağrıldı, onaylandı beklemede gibi) (onaylandı yada reddedildi den ziyade ) birden fazla durumu olan süreçlerde böyle senaryolarda lookup table şeklinde verinizi tutarak burada sadece Foregn key kullanabilirsiniz. Aynı şekilde erkek, kadın belirtmek istemiyor gibi 2 den fazla durum var gibi senaryoda ilgili datayı id şeklinde tutmamız gerekecektir. Askerlik durumu (yapıldı, yapımadı, muaf, tecilli …. ) Lookup Table lar da genellikle açıklama ve key olur.
* Entitylerimizi aşağıdaki desing e göre şekillendirelim.

metin, diyagram, paralel, plan içeren bir resim

Açıklama otomatik olarak oluşturuldu

* Services içerisinden hatırlanacağı üzere bizler direk bir Response alıyorduk. Bu kısmı dinamikleştirelim. Services içerisinden aslında bizler response u ele almaktayız. Bu noktayı dinamikleştirelim. Bu işlem için bir extention method yazalım.
* Bundan öncesinde view e direk data dönmek yerine bir takım kontroller yapabileceğimiz bir method yazalım. Burada kontrollerimiz şu şekilde olacak. ResponseType notfound ise notfound sayfasına redirect edecek, Validaton error varsa model errorları ekle, success ise datayı dön veya redirect et gibi işlemleri gerçekleştirelim.
* bu işlem projeye özgü olacağı için business içerisine yazılabilir. İstenildiği taktirde Core katmanı içersinde de yapılabilir. Ben Business içerisinde yapılmasını uygun gördüm.

A screenshot of a computer

Description automatically generated

Controller extend edileceği unutulmaması gerekmektedir. Class ve method static olması gerekecektir. Bu etapta controller extend edilecektir.

A screenshot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

* Sonrasında ilgili controller ve viewlerimizi oluşturabiliriz.

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

* Bu aşamadan sonra İnsan kaynakları bölümü oluşturalım ve mevcut iş ilanlarını paylaşalım. Bu işlem için ilk olarak bizler Listeleme işlemini yapabileceğimiz Dto nesnemizi oluşturalım.

A screenshot of a computer

Description automatically generated

* Advertiesment için ilgili mapping profile i yazdıktan sonrasında business kodunu yazalım. Business sürecinde şöyle bir durum mevcut. Ben bir Pager kullanmak istiyorum. Bu sayede ben Pagerlama işlemini yürütebilirim. **X.PagedList** gibi libraryler kullanabileceğim gibi manuel bir pagerlama süreci de mevcuttur. Ben bu süreçte manüel paging yapacağım.