Simple Architecture

Merhaba arkadaşlar;

**Simple Architecture**, başlangıç projesi olarak kurumsal mimari yapısı hazır, SOLID prensipleri ve Clean Code yazma teknikleri ile geliştirilmiş, ORM, AOP ve çok daha fazlasını bünyesinde barındırarak sizi proje hazırlık aşamasından kurtaran hazır bir mimari yapıdır. Bu doküman ile mimari yapıyı nasıl kuracağınızı ve kullanabileceğinizi öğreneceksiniz. Ayrıca mimari hakkında detaylı bilgi sahibi olacaksınız. Yapı ve bu doküman sürekli güncel tutulacaktır. Yeni özellikler eklenecek ve yeni gelişen teknolojilerle birlikte yapı her zaman son teknolojiyi kullanacak şekilde güncellenecektir.

Şimdiden iyi projeler dilerim.

**Taner Saydam**

Projenin Kaynak kodları Github linki

Architecture

[**https://github.com/TanerSaydam/SimpleArchitectureBackEndProject**](https://github.com/TanerSaydam/SimpleArchitectureBackEndProject)

**Code Generator**

[**https://github.com/TanerSaydam/SimpleArchitectureGenVSExtensions**](https://github.com/TanerSaydam/SimpleArchitectureGenVSExtensions)

Bana ulabileceğiniz adresler

Mail => **tanersaydam.gmail.com**

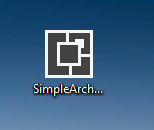
Website =>[**www.tanersaydam.net**](http://www.tanersaydam.net)

Linkedin =>[**https://www.linkedin.com/in/taner-saydam-b26336222/**](https://www.linkedin.com/in/taner-saydam-b26336222/)

Instagram =>[**https://www.instagram.com/taner.saydam.10/**](https://www.instagram.com/taner.saydam.10/)

Kurulum:

1. Kurulum için SimpleArchitecture dosyasını VisualStudio > Extensions kısmından ya da Visual Studio Market üzerinden ücretsiz olarak indirin
2. Eğer Visual Stuido’dan Extensions olarak indirdiyseniz Visual Studioyu kapatmanız yeterli, size kurulum için bir pencere gelecek, eğer dosyayı Visual Studio Market üzerinden indirirseniz onu çift tıklayarak kurulumu yapmalısınız. Visual Studio’nun kapalı olduğundan emin ol. Yoksa kurulum gerçekleşmez.



1. Dosya kurulumu oldukça basit. Install diyerek kurulumu başlatabilirsiniz.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Otomatik olarak kutulumu yapacaktır.

Kurulum tamamlandığında aşağıdaki gibi bir ekran görmeniz gerekiyor.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Projeyi Oluşturma

Proje oluşturmak için Visual Studio’yu açıyoruz.

Burada **Creat New Project** => **Yeni Proje Oluştur** diyoruz

metin içeren bir resim

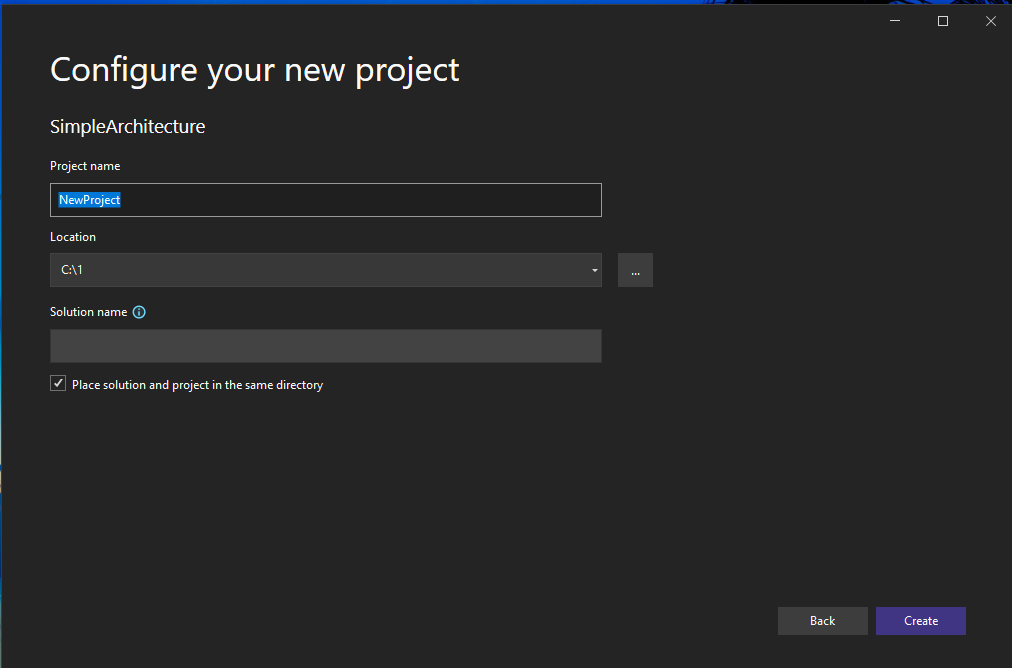
Açıklama otomatik olarak oluşturuldu

Arama çubuğuna **SimpleArchitecture** yazarak proje şablonunu bulabilirsiniz

metin, ekran görüntüsü, ekran içeren bir resim

Açıklama otomatik olarak oluşturuldu

Projeyi seçip devam diyoruz sonrasında Projenize isim verin



Create tuşuna bastığınızda proje hazır olarak ekrana gelecektir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Proje Yapısı

Proje yapısında hazır gelen 4 class – entiy var

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**User** => Kullanıcı için

**OperationClaim** => Yetki tanımlamaları için

**UserOperationClaim** => Kullanıcın yetkilendirilmesi için

**EmailParameter** => Mail service üzerinden mail gönderebilmek için parametrelerin tutulduğu class

**Context** – **Database bağlantısı**

DataAccess => Context => EntityFramework =>**SimpleContextDb.cs**

metin içeren bir resim

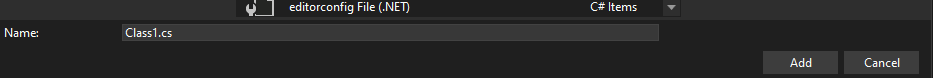
Açıklama otomatik olarak oluşturuldu

Yeni Entity - Class Ekleme

Projeye yeni bir entity eklemek için **Entities > Concrete** klasörüne bir **Class** oluşturmalısınız

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu



metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Önemli Not: Bundan sonraki işlemleri otomatik yapmak için **CodeGenerator**’u kullanabilirsiniz. Kullanım detayları için

**26. sayfaya** göz atın

Sonrasında **DataAccess** katmanındaki **Repositories** klasörüne Class isminin sonuna **Repository** ekleyerek bir klasör oluşturmalısınız

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Oluşturduğunuz klasör içerisinde **I** tagı eklerek **Class adınız** ve sonuna **Dal** tagı ekleyerek bir **Interface** oluşturmalısınız Örn: Class adınız **NewClass** ise => **INewClassDal**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Interface içerisinde **IEntityInterface**’i implement edip class adınızı **<>** tagları içinde vermelisiniz

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Not: Oluşturduğunuz class ve tüm ekleyeceklerinizin Public olmasına dikkat edin.

**Ef** tagı eklerek **Class adınız** ve sonuna **Dal** tagı ekleyerek bir **Class** oluşturmalısınız Örn: Class adınız **NewClass** ise => **EfNewClassDal**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Class içerisinde **EfEntityRepository**’i inherit etmeli ve <> tagları içerisine **Class**’ınızı ve **Context** dosyasınız olan ve default ismi **SimpleContextDb** olan context adınızı göndermelisiniz sonrasında virgül koyarak klasınızın interface’ini yazmalısınız

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Business katmanında **Repositories** klasörüne **Class adınız** + **Repository** isimli bir klasör oluşturmalı, içerisine de **Validation**, **Constans** isimli 2 klasör oluşturmalısınız. Sonrasın da da **I + Class adınız + Service** isimli bir **Interface** eklemelisiniz.

metin içeren bir resim

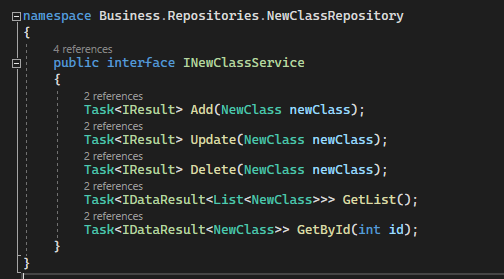
Açıklama otomatik olarak oluşturuldu

Interface’in içerisinde yazmak istediğiniz metotları göndermelisiniz. Tavsiye edilen yöntem **Result** Dönüş tipinde yapı kurmanızdır.

**IResult** => Success bilgisi ve Message dönerir

**IDataResult** => Data, Success, Message bilgisi dönerir

**Not: V.1.2 ile metotlar Asenkron metotlara döüştürüldü.**

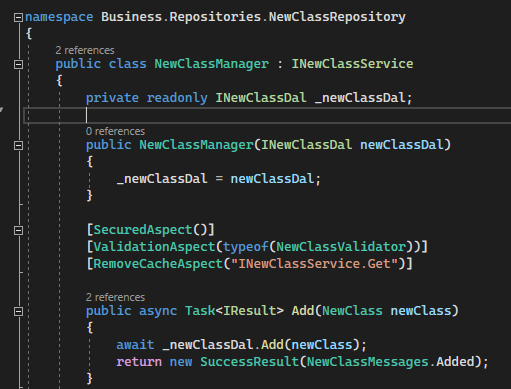


**Class adınız + Manager** isimli bir Class oluşturun.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Class** içerisinde az önce oluşturduğunuz **Service Interface**’ini implement edin ve bu class’a ait **Dal** interface’ini projenize **constructor** yapısıyla dahil edin. Sonrasında da implement interface yaparak metotlarınızı yazın



IResult yapısının dönüşü SuccessResult ya da ErrorResult’dur

IDataResult yapısının dönüşü SuccessDataResult ya da ErrorDataResult’dur

**Manager içerisinde kullanabileceğimiz aspectlet**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

[SecuredAspect()] => Kullanıcı girişini kontrol eder

[SecuredAspect(“Roles”)] => eğer parametre olarak role gönderirlirse token’da o rolleri arar yoksa sadece token’a bakar. Birden fazla parametre için virgül kullanmanız yeterli

[SecuredAspect(“Role1,Role2”)]

[ValidationAspect(typeof(ClassAdıValidator)] => Validation kontrolü için bu aspecti kullanmalısınız

[CacheAspect()] => Eğer bu metotu cache olarak tutmak istiyorsanuz bu aspecti kullanmalısınız. Parametre olarak dakika göndermezseniz cache otomatik 60 dk hafızada saklanır, isterseniz dakika belirtebilirsiniz.

[CacheAspect(60)]

[RemoveCacheAspect(“IServiceAdıSerivce.Get”)] => Cachelerinizi silmek için bu aspect kullanılmalı ve parametre olarak servis adınız ve metot adınızı ya da kısaltmasını göndermelisiniz.

[PerfomanceAspect()] => Metotun ne kadar sürede tamamlandığını takip eder ve işlem sonunda eğer parametre olarak saniye cinsinden süre belirtmezseniz default değeri 3 saniye olarak alır, 3 saniyeyi geçen işlemler için istediğiniz işlemleri yapar.

[PerfomanceAspect(5)]

İşlem yazmak için Core > Aspect > Performance > klasöründeki PerformanceAspect classına gelip

OnAfter metotunun altına işlemleri yazmalısınız

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Entity İşlemlerinin devamı

Oluşturduğunuz Validation klasörüne **Class adınız + Validator** isimli bir class oluşturun

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran, ekran görüntüsü, düz içeren bir resim

Açıklama otomatik olarak oluşturuldu

Constans klasörüne Class adınız + Messages isimli bir class oluşturun ve aşağıdaki gibi static mesajlar yazın

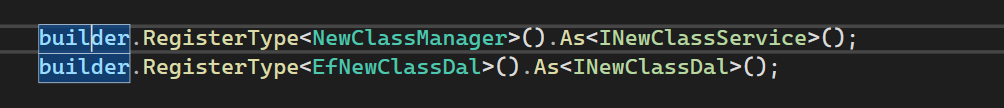
metin, ekran görüntüsü, ekran içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, iç mekan, gümüş içeren bir resim

Açıklama otomatik olarak oluşturuldu

Business katmanındaki DependencyResolvers > Autofac klasörü altında bulunan AutofacBusinessModule’ı açın ve eklediğiniz entitiy’e ait Dependency Injection’u gerçekleştirin



Entity için hazırlık aşamanız bitmiştir. WebApi katmanına apilerinizi yazabilirsiniz.

Mimariyi Tanıyalım

Projemiz 5 katmandan oluşmaktadır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Business Katmanı

Bu katman WebApi katmanına gelen tüm isteklerin iletildiği katmandır. Görevi gelen istekleri, söylenen kurallar çerçevesinde kontrol/uygulama yapıp sonuç olarak WebApi katmanına bilgi aktarmaktır.

Katman içerisinde ki klasör yapısı aşağıdaki gibidir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Aspect Klasörü => Business katmanını ilgilendiren aspect’lerin bulunduğu klasör

Authentication => Register ve Login işlemleri için oluşturulmuş Auth Serivice bulunan klasördür. Burada Kullanıcı kayıt ve kullanıcı giriş işlemleri yapılır

DependencyResolvers => Business içerisinde Dependency Injection Module’un bulunduğu klasör. Bu modüle ile dependency Injectionlar bu katmanda yapılabilir durumda.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Repositories => Bu klasör tüm entitilerimizin Service ve Manager yapısının tutulduğu klasördür. Her entity için önerilen yapı

EntityRepository Klasörü altına

IEntityService

IEntityManager

Validation Klasörü Altında

EntityValidator

Constans Klasörü Altında

EntityMessages

Bulunmaktadır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Utilities => Bu klasör Business katmanına ait araçların bulunduğu klasördür. İçerisinde örnek yapı olarak File klasörü altında IFileService bulunmaktadır. Bu araç yani service ile dosya kaydetme işlemlerini buradaki metotlar ile yapabilirsiniz.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Core Katmanı

Bu katman tüm projeye hizmet eden araçları ve yapıları içine barındırır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

DataAccess Katmanı

Bu katman Veri Tabanı ile bağlantıyı sağlar ve CRUD işlemlerini yapar.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Contenxt => Bu klasörde Database bağlantısı için context yapısı yer alır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Repositories => Bu klasörde tüm entitilerin DataAccessLayer yapısındaki DAL karşılıkları bulunur. Bu yapılar Repository yapısına bağlıdır ve CRUD işlemlerinin temeli bu yapılarda bulunur.

metin, elektronik eşyalar içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Entities Katmanı

Bu katman sadece entitilerin bulunduğu katmandır.

metin, elektronik eşyalar içeren bir resim

Açıklama otomatik olarak oluşturuldu

Concrete => Bu klasörde Database yapısında kayıtlı entitiler bulunur

Dtos => Bu klasörde Database’de bulunmayan, kullanım açısından ihtiyaç duyulan entitiy yapılarının oluşturulduğu ve tutulduğu yerdir

WebApi Katmanı

Bu katman son kullanıcı ile projeyi haberleştiren katmandır. Apiler burada üretilir ve son kullanıcıya verilir. Gelen tüm istekleri Business katmanına gönderir. İçerisinde kendine ait bir işlem yapmaz. Klasör yapısında sonradan eklenmiş tek klasör bulunur o da dosya kaydetme için oluşturulan Content klasörüdür

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu