



PROJE

Haftalık Rapor – 11.12.2020

11 ARALIK 2020
KIRIKKALE ÜNİVERSİTESİ – BİLGİSAYAR MÜHENDİSLİĞİ İÖ
AHMET Mungan – 160255081

1. KATMANLI MİMARİ DÜZENE GEÇİŞ VE ÖĞRENİLENLER

Katmanlı mimarinin 3 ana gövdeden oluştuğu öğrenildi. Var olan proje buna entegre edilmeye çalışıldı. Neler öğrenildiği ve uygulandığı sırayla anlatılacaktır.

Veri Erişim Katmanı : Bu katmanda veritabanı işlemleri yapılıyor.

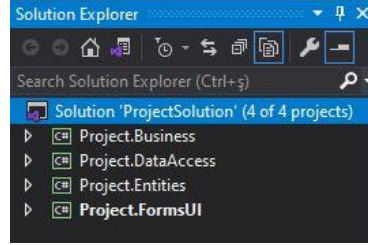
İş Katmanı : Bu katmanda işin gereği gerçekleşmesi gereken spesifik işler gerçekleştiriliyor.

Sunum Katmanı : Son kullanıcıya görünen ve kod yükünün en hafif olması gereken katmandır. Sade ve karmaşıklıktan uzak olması gerekir. Bir tek iş katmanı ile bağlantı içerisinde olması gerekir. Veri erişim katmanına doğrudan erişemez.

Bu 3 katmana göre proje düzenlendi ve veri erişim için EntityFramework kullanım şekli öğrenildi. Katmansız olarak daha evvel yazılan [Complex.pdf](#)^[1]'de: Tek başına kullanılmış olan ADO.NET gibi esnekliği düşük ve context (içerik) sağlamayan bir ORM tercih etmemek daha avantajlı olacaktır. EntityFramework'ün projem için en büyük avantajı ise tabloların varlıklar (entities) ile kontrol edilebilmesidir. Ayrıca one-point-one olarak veritabanına rahatlıkla erişebilmesi bir diğer avantajıdır. Context'in syntax olarak C#'ta sunduğu bir diğer avantaj ise `using (context) { }` bloğu içine yazılmasıdır. Bu blok veritabanı işlemleri için kullanıldığı zaman; veritabanı bağlantısını `.Open()` ya da `.Close()` gibi ek method ihtiyacı olmaksızın kullanmayı sağlıyor. Ayrıca `using` ile çağırılan kod blokları GC (Garbage Collector) yardımı olmaksızın işi bitince hafızadan silindiği için veri erişim sistemlerinde kullanım performansını artırıyor. Yüksek performanslı hafızayı yönetmek daha kolay olacağı için bir avantaj sunuyor. Fakat yine de big data'lar için EntityFramework önerilmese de bu proje bazında kullanmak avantajlı denebilir.

^[1] [Complex.pdf](#) tıklayarak gidilebilir. Önceki haftanın ek kısmındaki bağlantı

2. KATMANLARIN OLUŞTURULMASI VE İSİMLENDİRME KURALLARI



Folder 1

İsmlendirme kurallarını öğrenildi ve ona göre isimlendirmede bulunuldu. “**Projeİsmi.KatmanAdı**” gibi bir standart olduğu söylenebilir. Bu standartlara uyuldu.

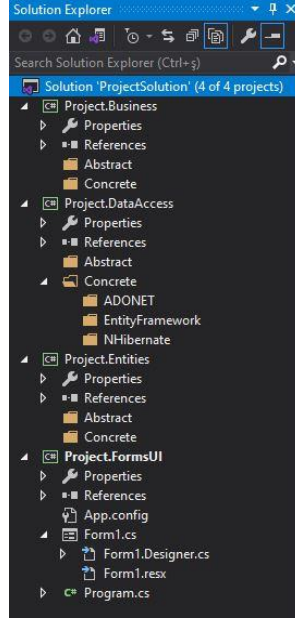
Project.DataAccess : Veri erişim katmanı. Bu katmanı Class Library’den oluşturuldu. Çünkü bu kısım görsel olmayacak ve veritabanı işlemleri için sadece Class yapısı kullanılacak.

Project.Business : Class Library tipinde bir katman ve son kullanıcı ile veriler arasındaki spesifik işleri yapılacak

Project.FormsUI : Son kullanıcı için daha sade ve görsel bir kod sunmak amacıyla bu katmanı Windows Form Application olarak oluşturuldu.

Project.Entities : Bu katman veri erişim katmanı ile birleştirilebilirdi normalde. Fakat veritabanına yeni tabloların eklenme ihtimali var. Ayrıca property’lerin göz önünde bulunmasını daha kullanışlı olacaktır.

Bu aşamada sunum katmanı hariç diğer katmanların içine, karışıklık olmaması için simetrik olarak 2 şer adet klasör eklendi. Yapılmasa da olur fakat burda standartlaşmış bir kullanım söz konusu. İlk klasör o katmanın genel yükünü taşıyıp ana Class’ların yapısını barındırıyor. İkinci klasör ise Class yapılarımız için Abstract, Interface veya Base nesne yapılarını barındırıyor. İlk klasöre “Concrete”, ikinci klasöre ise “Abstract” isimlendirmesi standartlaşmış bir durumda. Yani bir katmanda çalışan ana Class’ların Parent’ı ya da Interface’i varsa bunlar Abstract klasörünün içinde yer alıyor. EntityFramework yerine başka bir ORM tekniği kullanılabileceği düşüncesinden ötürü **Project.DataAccess** katmanının Concrete klasörünün altında bir biçimlendirmeye gidildi. “EntityFramework”, “Nhibernate” ve “ADONET” isimli 3 farklı klasör oluşturuldu. Şimdilik EntityFramework klasörünün içeri doldurulacak fakat daha sonra bu projenin devrini alan kişi için kolaylık olması açısından boş da olsa o iki klasörü oluşturup bu katmanın concrete klasörün içine dahil edildi.

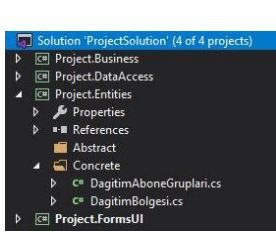


Folder 1

Son klasörleme işlemlerinden sonra Folder 1’deki son hali gösterilmiştir.

3. KATMANLI MİMARİNİN PROJEYE UYGULANMASI

EntityFramework, Visual Studio’da NuGet paket yöneticisi yardımıyla Data Access katmanına install edildi. **Project.Entities** katmanının altına proje özelinde kullanılan 2 tablonun varlık eşlemesi property yardımıyla yapıldı.



Folder 1

```
public class
DagitimAboneGruplari
{
    public int
    ProfilAboneGrupId { get;
set; }
    public int
    DagitimBolgesiId { get; set;
}
    public string Ad { get;
set; }
    public int Yil { get; set;
}
}
```

```
public class
DagitimBolgesi
{
    public int
    DagitimBolgesiId { get;
set; }
    public string Ad {
get; set; }
}
```

Project.DataAccess katmanında EntityFramework klasörünün içine veritabanı context’i oluşturmak için ProjectContext isimli bir public class oluşturup, bu class DbContext hazır class’ından inherit edildi.

VS Code 1

```
public class ProjectContext : DbContext
{
    public DbSet<DagitimAboneGruplari> DagitimAboneGruplariS { get; set;
}
    public DbSet<DagitimBolgesi> DagitimBolgesiS { get; set; }
}
```

Project.DataAccess katmanında artık eşleşmiş olan varlıklar ile ilgili veritabanı operasyonlarını gerçekleştirmek için “EfDagitimAboneGruplariDal” ve “EfDagitimBolgesiDal” isimli 2 farklı public class oluşturuldu. Burada “Dal” son eki data access layer anlamına gelmektedir. Bu tarz isimlendirmelere dikkat edilmesinin sebebi projeyi Refactoring gibi düzenlemeler kısmında işin kolaylaşması isteniyor. Bu iki Dal class’ının içeriğini kodlandı. Bir tanesini örnek için göstermek yeterli olur diye düşünüldü. Diğeri bu yapının aynısı olacaktır.

VS Code 2

```
public class EfDagitimAboneGruplariDal
{
    public List<DagitimAboneGruplari> GetAll()
    {
        using (ProjectContext context = new ProjectContext())
        {
            return context.DagitimAboneGruplariS.ToList();
        }
    }
}
```

VS Code 2’de hem context yapısına uygun olarak, hem de DataGridView nesnesi üzerinde görüntülemeye yardımcı olacak şekilde veriler burada List of DagitimAboneGruplari tipinde return edildi. VS Code 2’de using gibi ifadeler tekrar etmekte. Ayrı bir class yapısında muhafaza edip bu class’ları inherit etmek suretiyle kullanılabilir. Fakat bu tarz detayları haftaya bırakılacak ve şuan için katmanlı olarak çalışmanın ilk sonuçları alınmak isteniyor. İki hafta içerisinde proje %100’e yakın bir oranda bitmiş olacaktır. DagitimBolgesi class’ında herhangi bir kısıt olmaksızın tamamı veritabanından çekilecek, fakat DagitimAboneGruplari class’ı için bir filtreleme söz konusu olacaktır. Bu filtrelemeyi LINQ ve Delegate yapılarının sunduğu kolaylıklar kullanılarak veritabanı üzerinde procedure’ler ile uğraşılmayıp daha bütünleşik bir hale getirilmek istendi. Bunun için VS Code 2’deki EfDagitimAboneGruplariDal class’ına: Dönüş tipi DagitimAboneGruplari olan ve bir Id parametresi olan public bir method VS Code 3’te görüldüğü üzere eklendi.

VS Code 3

```
public DagitimAboneGruplari Get(int id)
{
    using (ProjectContext context = new ProjectContext())
    {
        return context.DagitimAboneGruplariS.SingleOrDefault
            (p => p.DagitimBolgesiId
            == id);
    }
}
```

Project.Business katmanında Concrete klasörünün altına her iki varlığın yönetilmesi için “*DagitimBolgesiManager*” ve “*DagitimAboneGruplariManager*” isimli 2 class oluşturuldu. Birisinin kod yapısını paylaşmak yeterli olacaktır. (VS Code 4)

VS Code 4

```
public class DagitimBolgesiManager
{
    EfDagitimBolgesiDal _dagitimBolgesiDal = new EfDagitimBolgesiDal();
    public List<DagitimBolgesi> GetAll()
    {
        return _dagitimBolgesiDal.GetAll();
    }
}
```

Bu çerçevede kodlar yazılırken gerekli referanslandırma ve kütüphane olarak ekleme işlemlerini de yapıyor. Arka plan işlemleri olduğu için onların ayrıca söylenmesine gereksinim duyulmadı, fakat nasıl yapıldığı ve temel mantığı öğrenildi. Şuan sadece mimari olarak test edilip ilk sonuçlarının alınması için bu şekilde tanımlamalar yapıldı.

Project.FormsUI katmanının sadece iş katmanını çalıştırabileceği bilindiği için Form1'in içine iş katmanına uygun kodlar yazılması gerekiyor.

VS Code 5

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private void Form1_Load(object sender, EventArgs e)
    {
        DagitimBolgesiManager dagitimBolgesiManager = new
        DagitimBolgesiManager();
        dgwDagitimBolgeleri.DataSource = dagitimBolgesiManager.GetAll();
    }
}
```

Project.Business katmanını ekledikten sonra VS Code 5'teki şeklini alacaktır. Yine **Project.FormsUI** katmanında yapılması gereken bir diğer konfigürasyon App.config dosyası içerisine XML diliyle bir connection string oluşturmaktır. XML bu proje özelinde çok kullanılmayacak olmasına rağmen projenin görünür kısımlarına uzun uzun connection string'ler eklememek görsel olarak büyük bir avantaj sağlayacaktır.

VS Code 6

```
<connectionStrings>
    <add name="ProjectContext"
        connectionString="data source=(localdb)\MSSQLLocalDB;initial
        catalog=project;" />
</connectionStrings>
```

```
    integrated security=true"  
    providerName="System.Data.SqlClient"/>  
</connectionStrings>
```

VS Code 6'daki gibi bir ekleme yaparak veritabanı bağlantısı oluşturulduğu öğrenildi.

ÖZET

Araştırmaların neticesine göre EntityFramework yapısının ve kullandığı bileşenlerinin daha reflektif bir yapı sunduğuna kanaat getirildi. Ayrıca mimari yapının kurumsal projeler üzerindeki pozitif etkisi gözler önüne serilmiştir. Bu çerçevede katmanlı mimarinin kurulması sağlanmış, bu katmanlar ile ilgili işlemler gösterilmiştir. Geçen haftaki özensiz ve küt kodlamadan sonra, projenin bu haftaki gelişimi gözlenebilir. Bu gözlemin yapılabilmesi için Bkz. **EKLER**

ANAHTAR KELİMELER

DagitimBolgeşi : İş özelinde kullanılmış olan bir isimlendirme.

DagitimAboneGrupları : İş özelinde kullanılmış olan bir isimlendirme.

VS Code X : Visual Studio kodlarından bir kesit içerir.

Folder X : Microsoft ürünü Visual Studio geliştirme ortamında proje genelinde yazılmış kod enstantenelerinden X. kod bloğu.

X.dbo : SQL veritabanı üzerinde X tablosunun .dbo uzantılı gösterimi.

EKLER

Projenin geçen haftaki hali: [TIKLAYINIZ](#). (Drive'a yüklüdür, güvenli bağlantı.)

Projenin bu haftaki sunum katmanı hali: [TIKLAYINIZ](#). (Drive'a yüklüdür, güvenli bağlantı.)