**MOBIROLLER**

**.NET CORE CASE**

Mobiroller tarafından verilen bu case çalışmasında, bir endpoint yardımıyla, kullanıcı Türkçe ve İtalyanca json dosyalarını import edebilecek ve import ettikleri veriler, MSSQL veritabanına kayıt olacaktır. Request Localization yardımıyla, Türk kullanıcılar için Türkçe, İtalyan kullanıcılar için İtalyanca bilgilerin sorgulanabilmesi sağlanacaktır.

Yapılan ödev, Visual Studio ortamında tasarlanmış, .Net Core Api projesidir.

1. **Proje çerçevesinde oluşturulmuş tablolar**
   1. History
   2. HistoryCategory
   3. Language
   4. User
   5. UserRole
   6. Role

Tablolar migration ile MSSQL veritabanına kurulmuştur. “Update-Database” komutuyla tabloları, veritabanınıza hızlıca oluşturabilirsiniz. Veritabanı yolu DataAccess/Concrete/AppDbContext içerisinde yer almaktadır. Bu konumdan veritabanı yolunu değiştirebilirsiniz.

1. **Proje çerçevesinde oluşturulan katmanlar**
   1. Entities
   2. DataAccess
   3. Core
   4. Business
   5. WebAPI

Entities katmanında, modeller ve dto’lar oluşturulmuştur. DataAccess katmanında, veritabanına yönelik işlemler ve sorgular gerçekleştirilmiştir. Core katmanında, yapılacak işlemlere yönelik generic yapılar, modellerde bulunan ortak sütunları barındıran interface ve classlar; cache, validation, ImethodInterceptor, aspect, token oluşturma gibi işlemler yapılmıştır. Business katmanında, webAPI ile repository katmanı arasında iletişim sağlanmış, ilgili veri filtrelemeleri vb. işlemler gerçekleştirilmiştir. WebAPI katmanında ise apiler oluşturulmuş olup, client ile ilk iletişime geçilen bölgedir.

1. **WebAPI katmanında oluşturulan controllerlar ve apiler**
   1. **BaseController**
      1. []AddAsync
      2. AddListAsync
      3. UpdateAsync
      4. DeleteAsync
      5. DeleteListAsync
      6. GetByIdAsync/{Id}
      7. GetAllAsync
   2. **HistoryController**
      1. AddAsync
      2. AddListAsync
      3. UpdateAsync
      4. DeleteAsync
      5. DeleteListAsync
      6. GetByIdAsync
      7. GetAllAsync
   3. **HistoryCategoryController**
      1. AddAsync
      2. AddListAsync
      3. UpdateAsync
      4. DeleteAsync
      5. DeleteListAsync
      6. GetByIdAsync
      7. GetAllAsync
   4. **ImportTRController**
      1. ImportFileAsync
   5. **ImportITController**
      1. ImportFileAsync
   6. **AuthController**
      1. Login
      2. Register
2. **Yapılan İşlemler**

AuthController ve ImportControllerlar hariç, diğerleri BaseControllerdan kalıtım almaktadırlar. Tüm ortak işlemler BaseController üzerinden gerçekleşmektedir.

Kullanıcı İtalyanca verileri ImportITController üzerinde oluşturulmuş Import adlı api üzerinden, Türkçe verileri ise ImportTRController üzerinde oluşturulmuş Import adlı api ile yükleyebilir. Kullanıcının yüklemiş olduğu konuma göre, verilere dil id’si verilir.

Dil kayıtları, AppDbContext sınıfında, OnModelCreating metodu içerisinde, migration çalıştırıldığı anda, otomatik olarak veritabanına eklenmektedir. 1 numaralı LanguageId’si Türkçe verilere, 2 numaralı LanguageId’si ise İtalyanca verilere verilmektedir.

Localization için, resources klasöründe ilgili dosyalar oluşturulmuştur. Burada da Türkçe veriler için 1, İtalyanca veriler için ise 2, anahtar kelime olarak tutulmuştur. Yapılan işlemlerde, resources içinden bu sayılar alınıp, veritabanında dil ayrım işlemi gerçekleştirilmektedir.

Startup içerisinde ExceptionHandling eklenmiştir. Burada TechBuddy.Middlewares.ExceptionHandling paketinden faydalanılmıştır. Kullanıcıya hatanın kendisi değil, default olarak yazmış olduğum mesaj iletiliyor olacaktır.

Entities ve WebAPI tarafında modeller oluşturulmuş, mapleme işleminde AutoMapper paketinden faydalanılmıştır.

Business katmanında, IService’ler için ortak IBaseService interface i oluşturulmuştur. Aynı şekilde ortak işlemlerin yapımı için, generic olarak BaseManager sınıfı oluşturulmuştur. IHistoryService, IHistoryCategoryService gibi interfaceler, IBaseService interface inden implemente edilmiştir. Aynı şekilde, HistoryManager, HistoryCategoryManager gibi classlar ise, BaseManager sınıfından kalıtım almıştır.

DependencyResolvers klasöründe yer alan AutofacBusinessModule classı içerisinde, Aspect tetikleme işlemleri gerçekleştirilmiştir. IOC klasöründe yer alan MicrosoftDependencies classında ise, bağımlılıklar giderilmiştir. Normalde, MicrosoftDependencies classı içerisinde yer alan ifadeleri, AutofacBusinessModule classı içerisine yazıp, Autofac kütüphanesinden faydalanılarak bağımlılıklar giderilebilir. Fakat orada almış olduğum bir hata ve süre kısıtımın olması nedeniyle, hatayla çok ilgilenemedim. Ödevi teslim edebilmek adına, şimdilik bu şekilde bıraktım.

ValidationRules klasöründe, proje içerisinde validation kullanmış olmak amacıyla, FluentValidation kütüphanesinden faydalanılarak, History model adına validasyon kuralları yazılmıştır.

DataAccess katmanında, abstract klasörü içerisinde interfaceler, Concrete klasörü içerisinde ise, bu interfacelerden implemente olacak classlar yer almaktadır. IDal interfaceleri, Core katmanında oluşturulmuş, ortak işlemleri barındıran IEntityRepository interfacesine bağlıdır. EfDal classları ise, yine Core katmanında oluşturulmuş, EfEntityRepositoryBase sınıfına bağlıdır.

Core katmanında, cache ve validasyon için aspectler oluşturulmuştur. IMethodInterceptorlar ile, metod çağırma işlemleri gerçekleştiğinde, araya girerek yapmak istediğimiz işlemleri gerçekleştirebilmemiz sağlanmıştır.

Kullanıcıya işlem başarılı-başarısız sonuçları dönebilmek amacıyla Utilities/Results klasörü altında gerekli yapı kurulmuştur.

Authorize işlemlerinden dolayı, işlemlerinizi Postman gibi uygulamalar üzerinden yapmanız faydanıza olacaktır.

Veritabanı işlemlerini migration ile oluşturabileceğiniz gibi, proje ana dizininde yer alan db klasörü içerisindeki script.sql