LEVEL 1 – Temel Kullanım Mantığı

Kütüphaneler

-Autofac
-Autofac.Extras.DynamicProxy

Level 1'de temel kullanımı göreceğiz.İlk önce interceptor yapılarımızı oluşturalım.Bunun için bölücü(interceptor) olarak kullanacağımız sınıfı IInterceptor interface'ini implement etmesi gerekir.IInterceptor interface ile gelen Intercept metodu bizim bölücü fonksiyonumuz olacak.

```
public class CacheInterceptor : IInterceptor
{
    public void Intercept(IInvocation invocation)
    {
        Console.WriteLine("Cache interceptor is run");
        invocation.Proceed();
    }
}
```

```
public class ValidationInterceptor : IInterceptor
{
    public void Intercept(IInvocation invocation)
    {
        Console.WriteLine("Validation interceptor is run");
        invocation.Proceed();
    }
}
```

Artık interceptor yapımız hazır birden fazla interceptor yapıları oluşturabilirsiniz.Fakat bu yapıları kullanabilmemiz için bellekte bir referansı olmaları gerekir.Bu yüzden ContainerBuilder (Referans Konteyneri) sınıfına register etmemiz gerekir.Aynı şekilde kullanılacak servisleride bu yapıya register ederiz.

```
static void Main(string[] args)
{
    var container = new ContainerBuilder();
    container.RegisterType<CacheInterceptor>();
    container.RegisterType<ValidationInterceptor>();
    container.RegisterType<ProductManager>().EnableClassInterceptors();


    var build=container.Build().BeginLifetimeScope();
    var item=build.Resolve<ProductManager>();
    item.Add();
 }
```

Artık AOP mimarimiz temel olarak hazır.Interceptor yapılarımızı hangi servislerde kullanmak istiyorsak bu yapıların üstüne Autofac kütüphaneleriyle birlikte gelen Intercept attribute ile belirtiriz.Birden fazla overloading kullanımlarını kendi dökümantasyonunda bulabilirsiniz.Çok uzayacağı için oralara değinmeyeceğim.Artık bu servislerinizdeki fonksiyonları kullanırken diğer fonksiyonlarınızı çalıştırabilirsiniz.

```
[Intercept(typeof(CacheInterceptor))]
[Intercept(typeof(ValidationInterceptor))]
public class ProductManager : IProductService
{
    public virtual void Add()
    {
        Console.WriteLine("Product is added!");
    }
}
```

NOT: Intercept attribute yapısı class ve interface yapılarına uygulanabilir.Bu yüzden fonksiyon bazında değil nesne bazında çalıştığından dolayı geliştiriciyi sınırlar.Bu sınırlamayı sonraki seviyelerde aşacağız.

NOT: Bir fonksiyon çalıştığında interceptor yapısının devreye girmesi için çalışan fonksiyonun virtual olması gerekir.Bunun mantığını ben de çözemedim bilen varsa linkedinden ulaşarak yardımcı olursa sevinirim :)

Sonraki seviyelerde görüşmek üzere...