Mocking

Mocking kavramı, yazdığımız kodumuzu test ederken bağımlılığımız bulunan nesnelerimizin sahtelerinin yaratılması işlemine verilen isimdir. Bu işlemin yapılmasının temel amacı Birim (unit) test yaparken gerçekten de birim düzeyde tutarak bu testlerin yapılmasını sağlamaktır. Gerçek nesnelere ulaşmaya çalışmak zaman kaybına neden olacaktır. Eğer sahte nesne yaratmayıp, gerçek bulunan nesnelerimiz üzerinden test yapmaya çalışırsak bu test Birim (Unit) test olmaktan çıkacaktır.

Mockito ise Java'da Birim (Unit) test için hazırlayacağımız sınıflarda kullanabileceğimiz bir mock kütüphanesidir. Nesnelerimizi mocklama(sahtesini üretme) ihtiyacı üzerine doğan bu kütüphane ile birlikte Java'da testlerimizi gerçekleştiririz. Mocklama ihtiyacı bizim projelerimizde bulunan bağımlılıklarımızdan ortaya çıkmış ve bizim için işleri kolaylaştırmak için tasarlanmış bu tarz kütüphaneler ile birlikte kullanılmaktadır.

Örnek olarak bizim elimizde bulunan **OgrenciDao** isminde bir sınıfımız olsun. Bu sınıf içerisinde database kısmında çeşitli işlemler, operasyonlar yaptığımızı düşünelim. Sonrasında buna ek olarak öğrencilerin kayıtları için **KayıtService** adında bir servis sınıfı için test yazmak istediğimizi düşünelim. Bu durumda ben kayıt işlemini OgrenciDao üzerinden yapabileceğim için burada ister istemez bir bağımlılığım olmuş olacaktır. İşte bu işlem için doğrudan OgreciDao sınıfımı kullanmak yerine ben bu nesnemi mocklayarak, KayıtService için yazabileceğim testleri yazabilirim. Mocklama işlemi için verdiğimiz örneğe göre;

```
public class KayitServiceTest{


    //Mock oluşturma işlemi...
    @Mock
    private OgrenciDao ogrenciDao;


    //Oluşturduğumuz Mock sınıfını asıl service sınıfımıza enjekte etmemiz gerekiyor...
    @Inject
    private KayitService kayitService;

}
```

Bu şekilde hem mock üreterek sonrasında ürettiğimiz mock nesnemizi kullanmak istediğimiz sınıfa enjekte ederek gerekli Birim(unit) testlerin yapılmasını sağlamış oluruz. Bunu Mockito kütüphanesi bizim yerimize yaparak işlerimizi kolaylaştırır. Yoksa her oluşturacağımız nesneler için biz somut olarak mock nesnesi üretmek zorunda kalacaktık. Bu yönüyle Mockito, bize avantajlar sağlayan bir kütüphanedir.

Bu tarz mock kütüphanelerinin diğer bir avantajı ise, içerisinde doğrulama mekanizmalarını bulundurmaları ve mocklanan nesne üzerindeki beklenen değerleri karşılaştırıp, testin başarı durumuna etki etmeleridir. Java ile birlikte sektörde en fazla kullanılan mocklama kütüphanelerinden olan Mockito kütüphanesinin yanı sıra JMock ve EasyMock kütüphaneleri de mevcuttur.