SPRING FRAMEWORK DESIGN PATTERNS

(TASARIM DESENLERİ)

Tasarım desenleri yani design pattern, yazılım geliştirmenin önemli unsurlarındandır. Geliştiricileri ortak kalıplar çevresinde toplayarak tasarımın anlaşılmasında önemli rol oynar. Spring Framework için çok sayıda tasarım deseni bulunur.

Bunların arasında; Singleton Pattern, Factory Method Pattern, Proxy Pattern ve Template Pattern yer alır. Sırasıyla bunların nasıl kullanıldığını bu yazı ile inceleyeceğiz.

SINGLETON PATTERN

Singleton pattern modeli bize bir sınıf tipinden yalnızca tek nesnenin oluşturulmasına olanak sağlayarak global bir erişim noktası sağlayan modeldir. Yani uygulama çalışma zamanında yalnızca bir nesne oluşturalacağını garanti eder.

Bu kullanım sayesinde tüm uygulama içerisinde yalnızca bir nesne olması gerekmektedir. Birden çok sınıfın aynı instance kullanması gerekmektedir. Her istekte yeniden bir nesne oluşturulmadığı için bellekten tasarruf sağlar. Veritabanı uygulamalarında kullanımı yaygındır.

Early Instantiation (Erken Örnekleme) ve Lazy Instantiation (Tembel Örnekleme) olarak 2 tipte kullanılabilir. Erken örnekleme ile örnek yükleme zamanında oluşturulur. Tembel örnekleme ile örnek gerektiği zaman oluşturulur.

```
class Database{

  private static Database object = new Database();
  //EARLY

  private Database(){}

  public static Database getDatabase(){
    return object;

  }

  public void doDatabase(){

    //Write our codes
  }

}
```

Erken Örnekleme Örneği

Factory Method Pattern

BeanFactory ve Application kullanarak bean'leri yüklemek için Spring tarafından kullanılır. İstenen nesnenin oluşması için soyut bir factory sınıfını gerektirir. Spring bu yöntemi genellikle Dependency Injection (DI) etrafında kullanır.

Bu yöntem sayesinde alt sınıfların (sub-classes) oluşturulacak nesne türünü seçmesine olanak sağlar. Özel sınıfları koda bağlama ihtiyacını ortadan kaldırdığı için gevşek bağlamayı destekler. Böylelikle kodumuzun ara eleman veya abstract (soyut) sınıf ile etkileşime girdiğini anlayabiliriz.

PROXY Pattern

Yapısal (Structural) Pattern içerisinde yer alan bir modeldir. Proxy modeli vekil veya yer tutan olarak bilinir. Orijinal olan nesneye dış dünyadan koruma sağlar. Bir nesnenin başka bir nesneye erişimini kontrol etmesine izin veren bir tekniktir.

Template Pattern

Davranış (Behavioral) Pattern başlığı altında yer alan bir modeldir. Bir fonksiyon için iskelet oluşturulmasını ve bazı adımların alt sınıflara ertelenmesi üzerine çalışan bir tasarım desenidir. Kodun yeniden kullanılması için kullanılan çok yaygın bir tekniktir. Alt sınıflar arasındaki ortak davranışların tekrar etmemesini ve ortak bir sınıfa alınarak gerektiğinde kullanılmasına dayanan bir modeldir.