Title

Spring Framework'te Kullanılan Tasarım Desenleri

. . .

Bu yazıda Spring Framework'te kullanılan tasarım desenleri hakkında bilgiler verip, bu tasarım desenlerini daha iyi anlamaya çalışacağız ve en sık faydalandığımız tasarım desenlerini daha yakından inceleyeceğiz.

. . .

Öncelikle Tasarım Desenleri (Design Patterns) kavramına bir göz atalım:

- Tasarım Desenleri; nesne-yönelimli (Object-Oriented) yazılım geliştirmede en iyi yöntemleri (Best Practices) belirtir ve buna uygun geliştirmeler yapmamızı sağlar.

. . .

En çok yararlanacağımız Tasarım Desenleri hangileridir ?

- Dependency Injection veya Inversion of Control (IoC)
- Singleton
- Factory
- Decorator
- Strategy
- Proxy
- Template Method

. . .

- DEPENDENCY INJECTION veya INVERSION of CONTROL (IoC) -
Spring framework; nesnenin (Bean) oluşturulması, birbirine bağlanması, yapılandırılması sürecinden başlayarak tamamen yok olması sürecine kadar ki yaşam döngüsünün (Life Cycle) yönetilmesinden sorumlu bir IoC konteynerine sahiptir.

Dependency Injection nedir ?
Dependency Injection; IoC'yi uygulamak için kullanılan bir tasarım desenidir. Bir sınıfın (class) dışında, bağımlı nesnelerin oluşturulmasına izin verir ve bu nesneleri farklı yollarla bir diğer sınıfa aktarır. Dependency Injection kullanarak, bağımlı nesnelerin oluşturulmasını ve bağlanmasını, onlara bağlı olan sınıfın dışına taşırız.

Inversion of Control nedir ?
Inversion of Control (IoC); bir tasarım ilkesidir. Adından da anlaşılacağı gibi, serbest bağlantı (loose coupling) elde etmek için nesne yönelimli tasarımda farklı türdeki kontrolleri tersine çevirmek için kullanılır. Burada kontroller, bir sınıfın ana sorumluluğu dışında sahip olduğu ek sorumlulukları ifade eder.

Peki, kontrolleri tersine çevirmek daha açıkça ne anlama geliyor, bunu biraz daha basite indirgemek için günlük hayatımızdan bir örnek verelim:
Her gün işe araba ile gittiğinizi ve bu arabayı kendinizin kullandığını varsayalım. Bu durum, arabayı sizin kontrol ettiğiniz anlamına gelir. IoC ilkesi, kontrolü tersine çevirmeyi önerir. Yani arabayı kendiniz sürmek yerine başka birinin arabayı kullandığı bir taksi çağırırsınız ve iş yerinize bu şekilde gidersiniz. İşte buna kontrolün tersine çevrilmesi denir. Yani araç kullanma kontrolü sizden —> taksiciye geçer. Artık kendiniz araba kullanmak zorunda değilsiniz ve asıl işinize odaklanabilmeniz için sürüşü taksiciye bırakabilirsiniz.

. . .

- SINGLETON -
Singleton; Gang of Four'un tasarım desenlerinden biridir ve Creational Design Patterns kategorisine girer.
Singleton tasarım deseni, bellekte (memory) hizmet sağlayabilecek nesnenin yalnızca tek bir örneğinin (instance) var olmasını sağlar.
Spring framework de Singleton tasarım deseni varsayılan kapsamdır (default scope).

. . .

- FACTORY -
Factory tasarım deseni; birden fazla alt sınıfı olan bir üst sınıfımız olduğunda ve girdiye bağlı olarak alt sınıflardan birini döndürmemiz (return) gerektiğinde kullanılır.
Alt sınıflara, oluşturulacak nesnelerin türünü seçme şansı verir. Uygulamaya özel sınıfları koda bağlama ihtiyacını ortadan kaldırarak gevşek bağlamayı (loose-coupling) destekler.

. . .

- DECORATOR -
Decorator tasarım deseni; önemli ve birçok ünlü frameworkte kullanılmıştır. Decorator tasarım deseni; çalışma zamanında bir nesnenin işlevselliğini değiştirmek için kullanılır. Aynı zamanda, aynı sınıfın diğer instanceları bundan etkilenmeyecektir. Kısacası, Decorator tasarım deseni "Nesnelere dinamik olarak sorumluluklar ekleyin" demektir.

. . .

- STRATEGY -
Temel olarak Strategy tasarım deseni, çalışma zamanında bir algoritma seçmeyi sağlayan davranışsal (behavioral) bir yazılım tasarım desenidir. Kodumuzda Strategy tasarım desenini uygulamak, kodumuzun Açık/Kapalı İlkesine (Open-Closed Principle) uymasını garanti edecektir. Bu ilke her yerde yaygın olarak uygulanmaktadır.

. . .

- PROXY -
AOP'de yaygın olarak uygulanır. Başka bir nesnenin kendisine erişimi kontrol etmesi için bir yer tutucudur. Yani bir işlevselliğe kontrollü erişim sağlamak istediğimizde Proxy tasarım desenini kullanırız.

. . .

- TEMPLATE METHOD -
Template Method tasarım deseni, bir method taslağı oluşturmak ve uygulama adımlarından bazılarını alt sınıflara (sub-class) implemente (implementation) etmek için kullanılır.

. . .

Umarım bu makale Java geliştiricilerine bir nebze de olsa katkı sağlamıştır.

Başka bir yazıda görüşmek üzere.

İstek ve önerileriniz için bana batuhankiltac@gmail.com adresinden ulaşabilirsiniz.