1. IOC and DI means?

1.1. IOC

Inversion of control bir yazılım tasarım presibidir ve projedeki bağımlılıkları en aza indirgemek amaçlanmaktadır. Projedeki bağımlılıkların oluşturulmasını ve yönetilmesini geliştiricinin yerine, framework'ün yapması olarak da açıklanabilir. Frameworkler birçok işi kendisi yapmakta ve bizim kodumuzu çalıştırmak için framework gerekli kaynakları ve çalışması gereken metotları oluşturup yönetmekte. Yazdığımızı kod bloğu çalışacağı zaman, framework bizim kodumuzu çağırır ve çalıştırır, ardından kontrol yeniden framework'e geçmesi olayının tümüne Inversion of Control adı verilmektedir.

Kendi kod bloğumuz hariç, bütün her şeyin yönetimi frameowrk tarafından kontrol edilmektedir. Örnek olarak, Spring ile not defteri uygulaması geliştirdik. Spring bir framework olduğu için bütün kaynakları kendisi ayarlayacak ve yönetecektir. Yani projeyi Spring başlatacak ve hazır olduğunda kendi kod bloğumuzu çalıştıracak. Not defteri için bir kayıt oluşturduğumuzda bizim kodumuzu çalıştıracak ve bittiğinde kontrolü yeniden Spring Framework'ü eline alacak. Bu olaya Inversion of Control adı verilmekte.

Inversion of Control'ün avantajları;

1. Bir methodun implementasyonundan izole bir şekilde çalıştırılabilmesini sağlar.
2. Farklı implementasyonlar arasında, kolayca geçiş yapabilmemizi sağlar.
3. Program modülerliğini arttırır.
4. Bağımlılıklar en aza indiği için test etmeyi/yazmayı kolaylaştırır.

Inversion of Control aşağıdaki gibi mekanizmalar ile uygulabilir;

1. Strategy Pattern
2. Service Lacator Pattern
3. Factory Pattern
4. Dependency Injection

Dependency Injection Spring'in en önemli parçasıdır ve Inversion of Control'ün bir uygulama metodudur.

1.2. Dependency Injection

Dependency Injection bir sınıfın ya da nesnenin bağımlılıklardan kurtulmasını amaçlayan ve o nesnyi olabildiğince bağımsızlaştıran bir programlama tekniğidir. DI uygulayarak; bir sınıfın bağımlı olduğu nesneden bağımsız hareket edebilmesini sağlayabilir ve kod üzerinde olası geliştirmelere karşın değişiklik yapma ihtiyacını ortadan kaldırabilirsiniz.

Framework'ler sadece business logic ile ilgilenmeyi sağlar ve diğer tüm operasyonları is kendisi halleder. Bunu da framework'ten bağımsız bir şekilde kod yazabilmemiz için genellikle Dependency Injection uygulayarak sağları. **Dependency Injection'ın avantajları;**

1. Bağımlılık oluşturacak nesneleri direkt olarak kullanmak yerine, bu nesneleri dışarıdan verilmesiyle sistem içerisindeki bağımlılığı minimize etmek amaçlanır. Böylece bağımlılık bulunan sınıf üzerindeki değişikliklerden korunmuş olunur.
2. Unit testlerin yazımını kolaylaştırırken doğruluğunu da arttırır. Yazılım geliştirmedeki en önemli konulardan biri de yazılım içerisinde bulunan componentlerin "loosely coupled" (gevşek bağlı) olmasıdır. Dependency Injection'da bunu sağlayabilen önemli tekniklerden birisidir. Böylece bağımsızlığı sağlanan sınıflar tek başına test edilebilir.

2. Spring Bean Scopes?

Spring IOC container tarafından yönetilen nesnelere “Bean” denir. Ayrıca Bean anotasyonu method düzeyinde kullanılır. Çalışma mantığı @Component gibidir. Tanımladığı şeyi Spring’in Application Context dediğimiz kısma atar. Eğer bir metod @Bean ile tanımlıysa, Spring tarafından yönetilen bir Bean ürettiğini belirtmiş olur. Spring framework çatısı altında 6 adet Bean Scope bulunmaktadır;

1. Singleton
2. Prototype
3. Request
4. Session
5. Application
6. Websocket

2.1. Singleton Scope

Bir bean default olarak singleton scope’a sahiptir. Bean singleton scope ile tanımlandığı zaman mevcut application context’imiz içerisinde o bean’den yalnızca tek bir adet initialize edileceğini garanti ederiz. Bu bean ile yapılacak olan tüm request’ler cache’lenmiş olan aynı nesne üzerinden yapılır. Bean nesnemiz üzerinde yapılan bir değişiklik bean’i kullanan tüm yerlerde etkilenecektir.**Kullanımı;**

```
@Bean
@Scope("singleton")
//@Scope(value = ConfigurableBeanFactory.SCOPE_SINGLETON)
public MyBean myBean() {
   return new MyBean();
}
```

2.2. Prototype Scope

Prototype ile belirlenmiş bir bean, container içerisinde çağırıldığında her request’te yeniden oluşturulacakır. Diyelim ki setter methoduna sahipt bir sınıf var. Bu sınıf için bir bean oluşturduğunuzda, her zaman sınıfın yeni bir instance’sını verecek ve nesne niteliklerini değiştirmek için setter’ı özgürce kullanıp çalıştıracaktır. Çok thread’li uygulamalar için uygun.**Kullanımı;**

```
@Bean
@Scope("prototype")
//@Scope(value = ConfigurableBeanFactory.SCOPE_PROTOTYPE)
public MyBean myBean() {
   return new MyBean();
}
```

2.3. Request Scope

Request scope değeri her bir http request'i bir adet bean oluşturur. Örneğin, bir "/products" API'si var. Controller bu isteği aldığında ve service methodunu çağırdığında, Request Scope ilei bir Bean'iniz olacak ve bu API isteği yanıtını geri gönderene kadar her zaman nesnein aynı instance2ını alırsınız, ancak yeni bir request geldiğinde, yeni bir instance gönderecektir. **Kullanımı;**

```
@Bean
@Scope(value = WebApplicationContext.SCOPE_REQUEST, proxyMode =
ScopedProxyMode.TARGET_CLASS)
//@RequestScope
public MyBean myBean() {
   return new MyBean();
}
```

2.4. Session Scope

Session scope değeri her bir http session'ı için bir adet bean oluşturur. Mesela Spring boot uygulamasında kullanıcı sessionlarını sürdürdüğünde, bu scope yardımcı olabilir. Session scope kullandığımızda, tüm session için (kullanıcı düzeyindeki oturumda)her zaman nesnenin aynı instance'ını return eder. Ancak kullanıcı oturumu kapandığında, yeni bir kullanıcı oturumu için nesnenin yeni bir instance'ını alacakır. **Kullanımı;**

```
@Bean
@Scope(value = WebApplicationContext.SCOPE_SESSION, proxyMode =
ScopedProxyMode.TARGET_CLASS)
//@SessionScope
public MyBean myBean() {
   return new MyBean();
}
```

2.5. Application Scope

Bir application scope, ServletContext'in yaşam döngüsü için bir bean örneğini oluşturur. Bu singleton scope'a benzer ancak aralarında farklılıklar mevcuttur. Bir bean application scope değerine sahipken bu bean çoklu servlet tabanlı uygulamar ile de paylaşılabilirken, singleton scope değerine bir bean yalnızca mevcut application context'i içerisinde tanımlıdır. **Kullanımı;**

```
@Bean
@Scope(value = WebApplicationContext.SCOPE_APPLICATION, proxyMode =
ScopedProxyMode.TARGET_CLASS)
//@ApplicationScope
public MyBean myBean() {
return new MyBean();
}
```

2.6. WebSocket Scope

WebSocket scope'a sahip bean'ler tipik olarak singleton scope yapısındadır ve herhangi bir WebSocket oturumundan daha uzun yaşar. Bu nedenle, WebSocket scope'una sahip bean'ler için bir proxyMode tanımı gerekir. Singleton davranış sergilediğini, ancak yalnızca bir WebSocket session'ı ile sınırlı olduğunu da söyleyebiliriz. **Kullanımı;**

```
@Bean
Scope(scopeName = "websocket", proxyMode = ScopedProxyMode.TARGET_CLASS)
public MyBean myBean() {
return new MyBean();
}
```

3. What does @SpringBootApplicaton do?

@SpringBootApplication anotasyonu uygulamanın girişi metodunu belirtir. Yani halk arasındaki tabir ile main fonksiyonudur. Uygulama bu metod ile başlar. Bu anotasyon **@Configuration**, **@EnableAutoConfiguration**, **@ComponentScan** anotasyonlarının üçünü de içeren temel bir anotasyondur.

- **@Configuration:** Java tabanlı konfigürasyon işlemi yapan bir anotasyondur.
- **@ComponentScan:** Projeye dahil edilen komponentlerin otomatik taranmasını sağlar.
- **@EnableAutoConfiguration:** Varsayılan konfigürasyonların otomatik gerçekleşmesini sağlar.

4. Why Spring Boot over Spring?

Spring kütüphanesi bize esneklik uygulamaya odaklanırken, Spring Boot kod uzunluğunu kısaltmayı ve bir web uygulaması geliştirmenin een kolay yolunu bize sunmayı amaçlamaktadır. Spring boot, uygulama geliştirme için gerekli olan süreyi bir hayli kısaltır. Neredeyse hiçbir konfigürasyon yapmadan tek başına bir uygulama oluşturulmasına yardımcı olur. Oto konfigürasyon Spring Boot için özel bir özelliktir. **Spring Boot;**

- Bağımsız(stand-alone) uygulamalar oluşturur.
- Gömülü olarak Tomcat, Jetty veya Undertow ile birlikte gelir.
- XMl konfigürasyonuna ihtiyaç duymaz.
- LOC(Lines of Code)'u azaltmayı hedefler.
- Başlatması kolaydır.
- Özelleştirme ve yönetim basittir.

Bu nedenle Spring Boot, Spring'in önüne geçerek hazır bir proje başlatıcısı olarak otomatik yapılandırma gibi özellikleri ile uzun kod yazmaktan ve gereksiz yapılandırmalardan kurtulamızı sağlar.

5. What is Singleton and where to use it?

Singleton(tek nesne) tasarım kalıbı, bir sınıfın tek bir örneğini almak için kullanılır. Amaç oluşturulan nesneye global erişim noktası sağlamaktır. Sistem çalıştığı sürece ikinci bir örnek oluşturulmaz, böylelikle istenen nesnenin tek bir defa oluşturulması garanti altına alınacaktır. Singleton nesneler ilk çağırıldıklarında bir kere oluşturulurlar ve sonraki istekler bu nesne üzerinden karşılanır.

Çoklu istemcili, örneğin bir web projesindeki nesne için singleton uygulayacaksak oluşturulan ilk örneğin kilitlenmesi gerekmektedir. Eğer kilitleme yapılmazsa iki farklı thread'in ard arda yapacağı istek sonucu birinin sonucuna ulaşamadan yeni bir örneklendirme yapar. Kilitleme yapılırsa nesne örneği kilitli olacağından, oluşturulan ilk örneğin işleminin bitmesini bekler ve ikinci bir istek yapıldığında oluşturulan ilk örneği kullanır.

6. Explain @RestController annotation in Spring Boot?

@RestController, **@Controller** ve **@ResponseBody** anotasyonlarının birleşiminden oluşan bir streotype'dır. Dolaylı olarak da **@Controller** üzerinden **@Component** anotasyonunu da barındırıyor. **@RestController** kendisi ile çalışan sınıfın URL ile istenen request'lere ait dönüş biçimini belirler. **@RestController** anotasyonu datanın kendisini JSON veya XML formatı ile direkt sunabiliyor.

7. What is the primary difference between Spring and Spring Boot?

Spring; bir web uygulamasıgeliştirmek için en yaygın olarak kullanılan Java EE kütüphanelerinden birisidir. Spring Boot ise bir kütüphane olmayıp, Spring tabanlı hazır bir proje başlatıcıdır.

8. Why to use VCS?

Versiyon kontrol sistemi, değişiklikleri takıp etmek ve her ekip üyesinin en son sürümle çalışmasını sağlamak için önemlidir. Birden çok ekip üyesinin üzerinde işbirliği yapacağı tüm kodlar, dosyalar ve varlıklar için sürüm kontrol yazılımı kullanmak gerekir. Versiyon kontrol sistemi ürünlerin daha hızlı geliştirilmesini sağlar. **Ayriyetten;**

1. Görünürlüğü arttırır.
2. Ekiplerin dünya çağında işbirliği yapmasına yardımcı olur.
3. Ürün geliştirme sürecini hızlandırır.

9. What are SOLID Principles? Give sample usages in Java?

SOLID yazılım prensipleri; geliştirilen yazılımın esnek, yeniden kullanılabilir, sürdürülebilir ve anlaşır olmasını sağlayan, kod tekrarını önleyen prensipler bütünüdür. **SOLID'in amacı;**

1. Geliştirdiğimiz yazılımın gelecekte gereksinimlere kolayca adapte olması,
2. Yeni özellikleri kodda bir değişikliğe gerek kalmadan kolayca ekleyebileceğimiz,
3. Yeni gereksinimlere karşın kodun üzerinde en az değişimi sağlaması,
4. Kod üzerinde sürekli düzeltme hatta yeniden yazma gibi sorunların yol açtığı zaman kaybını da minimuma indirmektir.

9.1. S-Single Responsibility Principle

Bir sınıf(nesne) yalnızca bir amaç uğruna değiştirilebilir, o da o sınıfa yüklenen sorumluluktur, yani bir sınıfın(fonksiyona da indirgenebikir) yapması gereken yalnızca bir işi olması gerekir.

9.2. O-Open Closed Principle

Bir sınıf ya da fonksiyon halihazırda var olan özellikleri korumalı ve değişikliğe izin vermemelidir. Yani davranışını değiştirmiyor olmalı ve yeni özellikler kazanabiliyor olmalıdır.

9.3. L-Liskov Substitution Principle

Kodlarımızda herhangi bir değişiklik yapmaya gerek duymadan alt sınıfları, türedikleri(üst) sınıfların yerine kullanabilmeliyiz.

9.4. I-Interface Segregation Principle

Sorumlulukların hepsini tek bir arayüze toplamak yerine daha özelleştirilmiş birden fazla arayüz oluşturmalıyız.

9.5. D-Dependency Inversion Principle

Sınıflar arası bağımlılıklar olabildiğince az olmalıdır özellikle üst seviye sınıflar alt seviye sınıflara bağımlı olmalıdır.

10. What is RAD model?

RAD modeli veya Hızlı Uygulama Geliştirme modeli, belirli bir planlama olmaksızın prototip oluşturmaya dayalı bir yazılım geliştirme sürecidir. RAD modelinde planlamaya daha az önem verilir ve geliştirme görevlerine dadha fazla öncelik verilir. Kısa sürede yazılım geliştirmeyi hedefler. Aşamaları;

- İş Modelleme
- Veri Modelleme
- Süreç Modelleme
- Uygulama Üretimi
- Test ve Ciro

11. What is Spring Boot starter? How is it useful?

Starterlar uygulamanıza ekleyebileceğiniz bir dizi bağımlılık tanımlayıcısıdır. Sizi kullanmak istediğiniz teknolojilerin her biri için arama yapıp teker teker bağımlılık olarak ekleme zahmetinden kurtarır. Starterlar sayesinde ihtiyacınız olan Spring ve ilgili teknolojileri kolayca uygulamaya eklenebilir. Örnek olarak Spring ve JPA kullanmak istiyorsanız "*spring-boot-starter-data-jpa*" bağımlılığını projenize eklemeniz yeterli olacaktır.

Resmi starterlar spring-boot-starter-* kalıbını kullanırlar. Spring-boot ismini resmi Spring Boot starterkları için ayrılmıştır. Eğer kendi starterınız oluşturacaksanız spring-boot şeklinde başlamaması gerekiyor.

12. What are the Spring Boot Annotations?

- **@Bean** – Bir metodun Spring tarafından yönetilen bir Bean ürettiğini belirtir.
- **@Service** – Belirtilen sınıfın bir servis sınıfı olduğunu belirtir.
- **@Repository** – Veritabanı işlemlerini gerçekleştirme yeteneği olan yapıldığı repository sınıfını belirtir.
- **@Configuration** – Bean tanımlamaları gibi tanımlamalar için bir Bean sınıfı olduğunu belirtir.
- **@Controller** – Requestleri yakalayabilme yeteneği olan bir web controller sınıfını belirtir.
- **@RequestMapping** – Controller sınıfının handle ettiği HTTP Requestlerin path eşleştirmesini yapar.
- **@Autorwired** – Constructor, Değişken ya dasetter mtedoları için dependency injection işlemi gerçekleştirir.
- **@SpringBootApplication** – Spring Boot autoconfiguration ve component taramasını aktif eder.

13. What is Spring Boot dependency management?

Maven sadece bir tane parent pom kullanmayı sağlar. Farklı bir parent pom kullanıldığında veya kullanılmak istendiğinde dependencyManagement özelliği kullanılarak benzer sonuç elde edilir. Fakat bazı özellikler dependencyManagement tarafından desteklenmez

14. What is Spring Boot Actuator?

Spring Boot Actuator, uygulamaların production ortamına hazır özellikleri(health check, disk usage, heap dump vs.) otomatik aktifleştirir ve farklı http endpoint'ler ile etkileşimde bulunmayı sağlayan bir yapı sunar. Kısacası Spring Boot uygulaması hakkında bilgi almamızı sağlar.

NAİL BURAK KOLAY