1- IOC ve DI ne anlama gelmektedir?

Inversion of Control (IoC) bir yazılım tasarım prensibidir ve basit tabiriyle nesnelerin uygulama boyuncaki yaşam d öngüsünden sorumludur diyebiliriz.
Uygulama içerisinde kullanılan objelerin instance'larının yönetimi sağlar ve bağımlılığı en aza indirgemeyi amaçlar.

Dependency Injection kısaca "bağımlılıkların loose coupled yani gevşek bağlı bir şekilde dışarıdan enjecte edilmesi " şeklinde tanımlayabiliriz.
Inversion of Control (IoC) ise bir yazılım tasarım prensibidir ve basit tabiriyle nesnelerin uygulama boyunca ki yaşa m döngüsünden sorumludur diyebiliriz.
Uygulama içerisinde kullanılan objelerin instance'larının yönetimi sağlar ve bağımlılığı en aza indirgemeyi amaçlar.

Container program içerisinde request edilen nesneleri abstraction'lara bağlı tutarak otomatik olarak oluşturan ve bağı mlılıklarını inject eden bir framework diyebiliriz.
Oluşturmuş olduğu bu nesneleri kendi içerisinde yönetimini yaparak tekrardan ihtiyaç duyulduğunda yeni bir instanc e oluşturmak yerine mevcut olan nesneyi atar.

IoC daha kolay test edilebilir loose coupling dediğimiz gevşek bağlı ve reusable bir yazılım desene oluşturmamızı sa ğlar.

IoC ilk başlarda implementasyonundan dolayı zor ve karmaşık gibi görünsede geliştirme yaptıkça ve sağladığı kolay lıkları fark ettikçe hayran kalınacak bir yazılım tasarım prensibidir.
Basitçe işleyişini anlatmak gerekirse; soyut tiplerin hangi somut tipler tarafından register edildiği bilgisini tutar. Uyg ulama içerisinde container'dan abstract bir nesne talebinde bulursunuz
ve size register bilgisinde tanımlı olan concrete type'ın instance'ını oluşturup verir. Bir tür object factory olarak düşü nebilirsiniz.

2- Spring bean Scopes

Beanlerimizin bir yaşam döngsü vardır. Bu yaşam döngüsü çerçevesinde istediğimiz işlemleri yapması için Beanimi zin kapsamını yani scope'unu belirlememiz gerekmektedir.
Spring Beanlerimizdeki scopeleri Spring IoC container tarafından yönetilir ve beanlerimizdeki nesnelerin ne zaman ve nasıl oluşturulacağını belirler.

Spring'e oluşturduğumuz beanlerin Scope'lerini belirterek yönetmemiz ve Spring'in bu belirtmemize göre oluşturma sını sağlamaktayız. Bu scope göre Beanimizin kullanım alanını bir bakıma belirtmiş olmaktayız.
Scope Çeşitleri
singleton
Varsayılan olarak her bean Singleton'dur. Bu Bean'den sadece bir tane üretilir.

prototype
Bean'e istek geldiğinde oluşturulur. Her istekte farklı bir instance oluşturulur.

request
Web uygulamaları için kullanılır. Her HTTP isteği geldiğinde instance oluşturulur.

session
Web uygulamaları için kullanılır. Her HTTP session oluştuğunda instance oluşturulur.

globalSession
Web uygulamaları için kullanılır. Her HTTP isteği geldiğinde sadece bir tane instance oluşturulur.

Kullanımı
Anotasyon olarak @Scope(“——“) olarak Bean’in en başına konulur.

```
@Scope("request")
public class Uye{
...
}
```

Xml konfigürasyonu olarak ise

```
<bean class="com.burakkutbay" id="uyeid" scope="request"></bean>
```

şeklinde Scope tanımlanmaktadır.

3- @SpringBootApplication ne yapar?

@SpringBootApplication anotasyonu uygulamanın giriş metodunu belirtir. Yani main fonksiyondur. Uygulama bu metod ile başlar.
@SpringBootApplication, sınıfın bir yapılandırma sınıfı olduğunu ve @EnableAutoConfiguration ve bileşende @Co mponentScan açıklaması aracılığıyla tarama yoluyla otomatik yapılandırmayı tetiklediğini belirtir.
Spring Uygulama Bağlamının otomatik yapılandırmasını sağlar.

4- Spring Boot neden Spring yerine tercih edilir?

Spring; bir web uygulaması geliştirmek için en yaygın olarak kullanılan Java EE (Java Enterprise Edition) kütüphan elerinden biridir. Java platformu için ayrıntılı bir programlama ve yapılandırma modeli sunar.
Java EE kapsamında uygulama geliştirmeyi basitleştirmeyi amaçlar ve biz geliştiricilerin daha üretken olmasına yard ımcı olur.

Diğer kütüphanelerin aksine Spring uygulamanın birkaç alanına odaklanır ve kapsamda birçok özellik sunar.

Bilindiği üzere Spring kütüphanesinin en önemli özelliği Dependency Injection dır.

Spring kütüphanesi bize esneklik uygulamaya odaklanırken, Spring Boot kod uzunluğunu kısaltmayı ve bir web uyg ulaması geliştirmenin en kolay yolunu bize sunmayı amaçlamaktadır.
Spring Boot, uygulama geliştirme için gerekli olan süreyi bir hayli kısaltır. Neredeyse hiçbir konfigürasyon yapmada n tek başına bir uygulama oluşturulmasına yardımcı olur.

Oto konfigürasyon Spring Boot için özel bir özelliktir.

Spring kütüphanesinin yararları
Spring kütüphanesi bir web uygulamasının tüm katmanlarına uygulanabilir.
Gevşek bağlılık (Loose Coupling) ve kolay test edilebilirlik sağlar.
XML ve Annotation konfigürasyonlarını destekler.
Singleton ve Factory sınıflarının ortadan kaldırılması için gerekli yeteneğe sahiptir.
Bildirimsel (Declarative) programlamayı destekler.
Spring Boot’un yararları
Bağımsız (stand-alone) uygulamalar oluşturur.
Gömülü olarak Tomcat, Jetty veya Undertow birlikte gelir.
XML konfigürasyonuna ihtiyaç duymaz.
LOC (Lines of Code) ‘u azaltmayı hedefler.
Başlatması kolaydır.
Özelleştirme ve yönetim basittir.
Bu nedenle Spring Boot bir kütüphane olmayıp, Spring tabanlı hazır bir proje başlatıcıdır. Otomatik yapılandırma gi bi özelliklerle sizi uzun kod yazmaktan kurtarır ve gereksiz yapılandırmalardan kurtulmanızı sağlar.

5- Singleton nedir? Nerede kullanılır?

Singleton yazılım patterni Creational Design Patterns(Yaratıcı Tasarım Kalıpları) kategorisinde yer alır. Oluşturulan sınıfın sadece bir adet instance'ı olmasını garantiler.
Genel olarak bir sınıftan kaç tane obje üretilidğini kontrol etmek istendiğinde singleton yapısı kullanılmış olur.Tek bir nesneye ihtiyaç duyulan durumlarda (veri tabanı bağlantıları, port bağlantıları, dosya işlemleri, loglama işlemleri,bildirimlerde, iş katmanı servislerimizde) kullanılır.. Bu kalıpta istenilen tek bir nesnenin yaratılm ası olduğundan ilgili sınıfın içinde nesnenin oluşturulması gerekir.Bu statik bir yöntem ile de mevcuttur.
Nesne statik metodun içinde oluşturulup sınıfın gizli olan elemanına atanır. Tek üretilen nesneye geri dönmek için r eferansına dönmek yeterli olacaktır ama bu yöntem çok verimli bir yöntem değildir .
Sebebi ise program belleğe yüklendiği anda bellekte bu tip sınıflar oluşturulur. Bu sebeple singleton design pattern kullanılarak tüm uygulama üzerinden bir kere oluşturulup defalarca kullanılan bir nesne tanımlanabilir.

6- Spring Boot'taki @RestController anotasyonunu açıklayınız.

RestController aynı zamanda Controller içeren bir interface 'dir. Herhangi bir template engine ihtiyacı olmaz. Spring Web ile birlikte default dönüş tipi bir Json Nesnesidir.
Spring Boot web kullandığımız için Jackson bağımlılığı örtüktür ve açıkça tanımlamamız gerekmez.Ve @RestContr oller ile açıklama eklediğiniz için açık json dönüşümü yapmanıza gerek yoktur.
Sadece bir POJO döndürün ve jackson serializer json'a dönüştürme işlemini halledecektir. Bu, @Controller ile kulla nıldığında @ResponseBody kullanmaya eşdeğerdir.
Her controller metoduna @ResponseBody yerleştirmek yerine, vanilla @Controller yerine @RestController yerleştir iriz ve @ResponseBody varsayılan olarak bu controller'daki tüm kaynaklara uygulanır.

7- Spring ve Spring boot arasındaki temel fark nedir?

Spring boot size hiç bir ayar dosyasıyla uğraşmadan, çoğu ayarı otomatik yaparak size anahtar teslimi çalışma ortam ı sağlar. Spring MVC, web uygulamaları için geliştirilmiş, çok gelişmiş ayarlar yapabileceğiniz bir çalışma çatısıdır. Spring MVC, Spring Data, Hibernate ve Tomcat kullanarak bir web uygulaması geliştiriyorsanız, tüm bu bileşenlerl e tek tek uğraşmanız gerekir. Spring boot ile bir proje hazırladığınızda, sonuçta projenizi bir jar dosyasıyla yayınlaya bilirsiniz.
Yani içinde Spring, Hibernate,Tomcat ve diğer tüm bileşenleri barındırır.

8- VCS neden kullanırız?

Kaynak kod yönetimi olarak da adlandırılan versiyon kontrolü, zaman içinde dosyalarda yapılan değişiklikleri yönet menize ve bu değişiklikleri bir veritabanında depolamanıza olanak tanır.
Versiyon kontrol sistemi, yazılım ekiplerinin zaman içinde kaynak kodda yapılan değişiklikleri yönetmesine yardım cı olan yazılım araçtır.
Versiyon kontrol yazılımını kullanmak, yüksek performanslı yazılımlar geliştirmek ve DevOps süreçlerini doğru şek ilde çalıştırmak için büyük faydalar sağlar.

Versiyon kontrolü ayrıca geliştiricilerin daha hızlı hareket etmesine yardımcı olur ve ekip daha fazla geliştiriciyi içer ecek şekilde ölçeklenirken yazılım ekiplerinin verimliliği ve çevikliği korumasına olanak tanır.

Versiyon kontrol sistemlerinin sağladığı bazı faydalar aşağıdaki şekildedir.

Her dosyanın tam bir uzun vadeli değişiklik geçmişi tutulur. Buda, dosya üzerinde yıllar içinde birçok kişi tarafında n yapılan her değişikliğin tutulması anlamına gelir.
Ekip üyelerinin eşzamanlı olarak aynı kod üzerinde çalışmasına imkan verir. Alt sürümler oluşturarak yazılım üzeri nde farklı çalışmaları yürütüp, sonrasında ana yazılıma bunu entegre etmek mümkündür.
Yazılımda yapılan her değişikliği takip edip proje yönetimine bağlayabilme imkanı sağlar. Yazılım üzerindeki sorunl arın, sürümler ile ilişkilendirilebilmesine ve takip edilebilmesine olanak sağlar.
Versiyon kontrolü, değişiklikleri takip etmek ve her ekip üyesinin en son sürümle çalışmasını sağlamak için önemlid ir.

Birden çok ekip üyesinin üzerinde işbirliği yapacağı tüm kodlar, dosyalar ve varlıklar için sürüm kontrol yazılımı ku llanmak gerekir.

Versiyon kontrol sistemi ürünleri daha hızlı geliştirmenize yardımcı olur.

Versiyon kontrol sistemi önemlidir., çünkü

Görünürlüğü arttırır.
Ekiplerin dünya çapında işbirliği yapmasına yardımcı olur.
Ürün geliştirme sürecini hızlandırır.

9- SOLID Prensipleri nedir? Java kullanımına örnek veriniz

SOLID yazılım prensipleri; geliştirilen yazılımın esnek, yeniden kullanılabilir, sürdürülebilir ve anlaşılır olmasını sa ğlayan, kod tekrarını önleyen ve Robert C. Martin tarafından öne sürülen prensipler bütünüdür.
Kısaltması Michael Feathers tarafından tanımlanan bu prensiplerin amacı;

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

S — Single-responsibility principle
ÖZET: Bir sınıf (nesne) yalnızca bir amaç uğruna değiştirilebilir, o da o sınıfa yüklenen sorumluluktur, yani bir sınıf ın(fonksiyona da indirgenebilir) yapması gereken yalnızca bir işi olması gerekir.

O — Open-closed principle
ÖZET: Bir sınıf ya da fonksiyon halihazırda var olan özellikleri korumalı ve değişikliğe izin vermemelidir. Yani dav ranışını değiştirmiyor olmalı ve yeni özellikler kazanabiliyor olmalıdır.

L — Liskov substitution principle
ÖZET: Kodlarımızda herhangi bir değişiklik yapmaya gerek duymadan alt sınıfları, türedikleri(üst) sınıfların yerine kullanabilmeliyiz.

I — Interface segregation principle
ÖZET: Sorumlulukların hepsini tek bir arayüze toplamak yerine daha özelleştirilmiş birden fazla arayüz oluşturmalı yız.

D — Dependency Inversion Principle
ÖZET: Sınıflar arası bağımlılıklar olabildiğince az olmalıdır özellikle üst seviye sınıflar alt seviye sınıflara bağımlı o lmamalıdır.

Örnek:
1. Tek Sorumluluk İlkesi (The single-responsibility principle)
Bu ilke, “ bir sınıfın değişmesi için tek bir neden olması gerektiğini ” ifade eder , bu da her sınıfın tek bir sorumluluğ u olması gerektiği anlamına gelir.

```
public class Vehicle {
   public void details() {}
   public double price() {}
   public void addNewVehicle() {}
}
```

10- RAD Model nedir?

RAD Modeli veya Hızlı Uygulama Geliştirme modeli, herhangi bir özel planlama olmaksızın prototiplemeye dayalı bir yazılım geliştirme sürecidir. RAD modelinde, planlamaya daha az dikkat edilir ve geliştirme görevlerine daha faz la öncelik verilir.
Kısa sürede yazılım geliştirmeyi hedefler.
SDLC RAD modellemesinin aşağıdaki aşamaları vardır;
İş modeli
Veri Modelleme
Süreç Modelleme
Uygulama Üretimi
Test ve Ciro

11- Spring boot starter nedir? Nasıl faydalıdır?

Starterlar kısaca uygulamanıza ekleyebileceğiniz bir dizi bağımlılık tanımlayıcısıdır. Sizi kullanmak istediğiniz tekn olojilerin her biri için arama yapıp teker teker bağımlılık olarak ekleme zahmetinden kurtarır. Starterlar sayesinde iht iyacınız
olan Spring ve ilgili teknolojileri kolayca uygulamanıza ekleyebilirsiniz. Örnek olarak Spring ve JPA kullanmak isti yorsanız spring-boot-starter-data-jpa bağımlılığını projemize eklemeniz yeterli olacaktır.
Resmi starterlar sprint-boot-starter-* kalıbını kullanırlar. spring-boot ismni resmi Spring Boot startırları için ayrılmış tır. Eğer kendi starterınızı oluşturacaksanız spring-boot şeklinde başlamaması gerekiyor.
Kendi oluşturacağınız starter örnek olarak benimprojem-spring-boot-starter şeklinde olabilir. Resmi spring boot start erları ise pom.xml e eklenebilir.

12- Spring boot anotasyonları nelerdir?

@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
@Autowired - Constructor, Değişken yada setter metodlar için dependency injection işlemi gerçekleştirir
@SpringBootApplication - Spring Boot autoconfiguration ve component taramasını aktif eder.
@Controller Spring MVC Controller olarak sınıfları işaretlemek için kullanılır. Bu anotasyon sayesinde rahatlıkla Js p ve Thymeleaf gibi şablonlar kullanılmaktadır.
@RestController rest api denemeleri yaparken kullanmak için bu anotasyon kullanılması gerekmektedir. Bu anotasy on ile ilgili Jsp ve Thymeleaf şablonları kullanılmamaktadır. @Controller den farklı olarak json ve xml formatları da talar
ile ilgili işlemler yapmak için kullanılmaktadır.
@RequestMapping() kullanılırken hangi linke tıklarsanız o link ile ilgili çalışacak method belirlenmek için bu anotas yon kullanmak gerekmektedir.
@ResponseBody kullanıldığı method içerisinde çıktıyı String bir ifade olarak göstermenizi sağlar.
@RequestParam() herhangibir link üzerinde örn. localhost:8080/kitap?id=1 üzerinde id parametresinin sahip olduğu değere ulaşmak için kullanılmaktadır.
@PathVariable() ile herhangibir link üzerinde örn. locahost/user/{id} ile user'ın detaylı bilgilerini çekmek istediğimi zde bu anotasyon ile id parametresinin sahip olduğu değere ulaşmak için kullanılmaktadır.
@Entity Tanımlanan sınıfın bir JPA varlığı olduğunu belirtir.
Uygulama çalıştıktan sonra veritabanı işlemlerinin buradaki verilere göre yapılacağı tanımlanır.
Eğer @Entity veya @Table içinde ek bir belirtim yapılmadıysa, sınıf adı tablo adı olur.
@Table belirtimi yalnızca veritabanı ile ilgili yapmak istediğimiz özel belirtimler için kullanılır.

13- Spring Dependency'lerinin yonetimi nedir?

Spring Boot, bağımlılıkları ve yapılandırmayı otomatik olarak yönetir. Spring Boot'un her sürümü, desteklediği bağı mlılıkların bir listesini sağlar. Bağımlılıkların listesi, Maven ile birlikte kullanılabilecek Malzeme Listeleri'nin

(spring-boot-dependencies) bir parçası olarak mevcuttur.Bu nedenle, yapılandırmamızdaki bağımlılıkların sürümünü belirtmemize gerek yoktur. Spring Boot kendini yönetir. Spring Boot, Spring Boot sürümünü güncellediğimizde tüm bağımlılıkları
otomatik olarak tutarlı bir şekilde yükseltir.

14- Spring boot actuator nedir?

Yaptığımız ya da yapacağımız Spring Boot uygulamalarımızın endpoint yardımı ile çalışan Spring Boot uygulaması hakkında bilgi almamızı sağlamaktadır.
Spring Boot uygulamanızın sonuna actuator/beans , actuator/health gibi yukarıda belirlediğimiz parametreleri yazarak endpointlere ulaşıp bilgi almamız mümkün olmaktadır.

/actuator/health endpoitine ulaşırsanız şu ekran sizi karşılayacaktır.

```
{
   "status" : "UP"
}
```
Bazı endpointleri kapatmak isterseniz application properties dosyasından değerlerini false yapabilmek mümkün.