**1 – IOC and DI means ?**

Inversion of control bir yazılım tasarım prensibidir. IOC ile uygulama içerisindeki obje instance’larının yönetimi sağlanarak, bağımlılıklarını en aza indirgemek amaçlanmaktadır. Projemizdeki bağımlılıkların oluşturulmasını ve yönetilmesini geliştiricinin yerine, framework’ün yapması olarak da açıklanabilir.

Bağımlılık oluşturacak nesneleri direkt olarak kullanmak yerine, bu nesnelerin dışardan verilmesiyle sistem içerisindeki bağımlılığı minimize etmek amaçlanır. Böylece bağımlılık bulunan sınıf üzerindeki değişikliklerden korunmuş oluruz. Dependency injection bir sınıfın/nesnenin bağımlılıklardan kurtulmasını amaçlayan ve o nesneyi olabildiğince bağımsızlaştıran bir programlama tekniği/prensibidir.

**2 – Spring Bean Scopes ?**

Bean Nedir ?

Spring Framework uygulamamızın omurgasını oluşturan ve Spring IOC container tarafından yönetilen nesnelere BEAN denir. Yeniden kullanılabilir objeler olarak kabul edebiliriz.

Scope Nedir ?

Scope kelimesi kapsam, alan, faaliyet alanı gibi anlamlar taşımakta biz de Bean objelerimizin birer yaşam alanının var olduğunu düşünebiliriz.

Scope Türleri

Singleton : Her Bean varsayılan olarak singleton olmakla birlikte sadece bir kez üretilir. Singleton Design Pattern’deki gibi düşünebiliriz. Bir kere üret tekrar tekrar kullan.

Prototype : Söz konusu olan bean için her bir istek yapıldığında oluşturulur. Her oluşturmada farklı bir instance üretilir.

Request : Request bean’i adından da yola çıkarak HTTP isteği geldiğinde oluşturulur. HTTP request düzeyinde etkin bir şekil kapsama alınır.

Session : Session Scope Web Uygulamalarında HTTP isteği geldiğinde oluşturulur. Request Scope’a benzer bir yaklaşım.

Global Session : Bir HTTP’nin yaşam döngüsünde tek bir Bean’ın tanımını kapsamaktadır. Yanlızca WEB’e duyarlı bir Spring’te geçerlidir.

**3 – What does @SpringBootApplication do ?**

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

**4 – Why Spring Boot over Spring?**

Spring Boot Spring Framework’e hızlı uygulama geliştirme özelliği sağlayan bir spring modülüdür. Basit ve web tabanlı uygulamaları kurmak, yapılandırmak ve çalıştırmak için daha kolay ve hızlı bir yol sağlar. XML yapılandırması için gereklilik yoktur. Gömülü http sunucuları yardımıyla web uygulamalarını kolayca test eder.

**5 – What is Singleton and where to use it ?**

Singleton(tek nesne) tasarım kalıbı, bir sınıfın tek bir örneğini almak için kullanılır. Veritabanı bağlantısı, ön-bellek , yapılandırma dosyası gibi uygulamalarda kullanılır.

**6 – Explain @RestController annotation in Sprint boot?**

@RestController URL ile istenen request’lere ait dönüş biçiminde herhangi bir template engine'ne (şablon motoru) ihtiyaç duymaz. Datanın kendisini JSON veya XML formatı ile direkt olarak sunar.

**7 - What is the primary difference between Spring and Spring Boot ?**

Spring Boot Spring framework’e hızlı uygulama geliştirme (RAD) özelliği sağlayan bir spring modülüdür.Spring boot içerisinde gömülü http serverları barındırır. Xml konfigürasyonunu otomatikleştirir.

**8 – Why to use VCS ?**

Version Control System bir döküman (yazılım projesi, ofis belgesi…) üzerinde yaptığımız degişiklikleri adım adım kaydeden ve bunu internet üzerinde depoda (respository) saklamamızı ve yönetmemizi sağlayan bir sistemdir. Örn Git.

Bir dosyanın değişik sürümlerini korumak istiyorsak, VCS kullanırız. VCS, dosyaların ya da bütün projenin geçmişteki belirli bir sürümüne erişmemizi, zaman içinde yapılan değişiklikleri karşılaştırmamızı, soruna neden olan şeyde en son kimin değişiklik yaptığını, belirli bir hatayı kimin, ne zaman sisteme dahil ettiğini ve bir çok şeyi görebilmemizi sağlar. Öte yandan, VCS kullanmak, bir hata yaptığımızda ya da bazı dosyaları yanlışlıkla sildiğimizde durumu kolayca telâfi etmemize yardımcı olur.

**9 – What are SOLID Principles ? Give sample usages in Java ?**

SOLID prensipleri ; geliştirilen herhangi bir yazılımın esnek, yeniden kullanılabilir, sürdürülebilir ve anlaşılır olmasını sağlayan, kod tekrarını önleyen prensiptir. Kodun esnek, sürdürülebilir ve geliştirilebilir tasarlanmaması kodu kırılganlaştırır ve yazılım ürününün gelişmesini etkiler. SOLID 5 farklı prensipten oluşur ve her birini baş harfini alır.

1*) S : Single Responsibility Principle (SRP)*

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

2) O : Open Closed Principle (OSP)

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

[3) Liskov substitution principle](https://gokhana.medium.com/liskov-substitution-prensibi-nedir-kod-%C3%B6rne%C4%9Fiyle-soli%CC%87d-3cfc1cd63c1a)

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

[4) Interface segregation principle](https://gokhana.medium.com/interface-segregation-prensibi-nedir-kod-%C3%B6rne%C4%9Fiyle-soli%CC%87d-ac0fd6812ecf)

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

[5) Dependency Inversion Principle](https://gokhana.medium.com/dependency-inversion-prensibi-nedir-kod-%C3%B6rne%C4%9Fiyle-soli%CC%87d-b61296523565)

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

**10 - What is RAD model ?**

RAD(Rapid Application Development) yani Hızlı Uygulama Geliştirme bir yazılım geliştirme yöntemidir. Çok fazla detaya girilmeden, hızlı şekilde çalışan bir uygulama oluşturmayı amaçlar.

RAD modellemesinin aşağıdaki aşamaları vardır

İş modeli , Veri Modelleme , Süreç Modelleme , Uygulama Üretimi , Test ve Ciro

Bilginin girdi-çıktı kaynağına ve hedefine odaklanır. Projeleri küçük parçalar halinde teslim etmeye vurgu yapar; daha büyük projeler bir dizi küçük projeye bölünmüştür. RAD modellemenin temel özellikleri, şablonların, araçların, süreçlerin ve kodun yeniden kullanımına odaklanmasıdır.

*RAD Metodolojisi ne zaman kullanılır?*

\*Kısa sürede (2-3 ay) bir sistem üretilmesi gerektiğinde

\*Gereksinimler bilindiğinde

\*Kullanıcı tüm yaşam döngüsü boyunca dahil olduğunda

\*Teknik risk daha az olduğunda

\*2-3 ay gibi kısa bir sürede modüler hale getirilebilen bir sistem oluşturma zorunluluğu olduğunda

\*Bir bütçe, kod oluşturma için otomatik araçların maliyeti ile birlikte modelleme için tasarımcıları karşılayacak kadar yüksek olduğunda

**11 - What is Spring Boot starter ? How is it useful ?**

Spring Boot uygulama türüne göre uygulama ayarlarını varsayılan-otomatik ayar özelliği ile yaparak daha hızlı uygulama geliştirmeyi sağlar.

Spring Boot bağımlılık, maven ve diğer ayarları **parent pom** özelliğini kullanarak varsayılan ayarları otomatik olarak belirler.

Spring Boot geliştirilecek olan proje türüne göre ihtiyaç duyulan ve kullanılacak kütüphaneleri **spring-boot-starter** ön eki altında toplar.

* spring-boot-starter-web – web tabanlı uygulama
* spring-boot-starter-security – spring security

**12 – What are the Spring Boot Annotations?**

Bir öğenin tanımını yapar, ne yapması gerektiğini açıklar ve yazılım geliştirme sürecini hem hızlandırır hem de kolaylaştırır.

Örn : @Entity, @TestCase, @WebService

Anotasyonlar kodun içerisinde tanımlandıktan sonra, işlevsel hale gelebilmeleri için Spring tarafından 2 defa taranırlar.

İlk önce yalnızca anotasyonları (spring tarafından yönetilen bean) taranır ve yapılmas gereken görev eşleştirmeleri yapılır.

İkinci taramada ise anotasyon tanımlamasına göre  işlemini yapar.

Tüm Spring Bean’leri “App Context” yada "Spring Context" (IoC container) adı verilen bir container içinde yaşarlar

**13 – What is Spring Boot dependency management?**

Nesne yönelimli programlama metodolojisi ile yazılımın geliştirildiği ortamlarda ilerleyen süreçlerde nesneler arası bağ kurmak zor olabiliyor. Bir nesnede yapılan değişiklikler veya yerine başka nesneyi koymak, başka yerlerde problemlere yol açabiliyor. Bu problemleri en aza indirmek için de Dependency Inversion gibi prensiplere ihtiyaç duyuyoruz. “Dependency Inversion” prensibi, sınıfları arası geçişlerin “loosely coupled” yani “gevşek bağlı” olması anlamına geliyor. Yüksek seviye sınıfların, düşük seviye sınıflara direkt bağlı olmaması, her ikisinin de soyutlamalara bağlı olması beklenilen davranıştır. Buradaki soyutlamalar, detaylara bağlı olmamalıdır. Detaylar, soyutlamalara bağlı olmalıdır.

**14 -  What is Spring Boot Actuator?**

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