

1-) IOC (Inversion of control) ; Projedeki bağımlılıkların oluşturulmasını ve yönetilmesini geliştirici yerine kendisi yapar.(Lifecycling)

Yazılan kod bloğu çalışacağı zaman framework, yazılı kod bloğunu çağırır ve çalıştırır.Daha sonra kontrol yeniden framework'e geçer. Bu olaylar IoC nin çalışma prensibidir.

Özellikleri;

- *Bir metodun implementasyonundan izole bir şekilde çalıştırılabilmesini sağlar.
- *Farklı implementasyonlar arasında kolayca geçiş yapmayı sağlar.
- *Program modülerliğini artırır.
- *Test etmeyi ve yazmayı kolaylaştırır.

DI (Dependency Injection) ; Türkçeye bağımlılıkları zerketmek olarakda çevrilebilir. Bir class'ın bağımlı olduğu diğer classları sınıf içersine dışarıdan enjekte etmekte denilebilir.

Kodu sadece bir amaç uğruna değil, ileride müşterinin isteyeceği değişiklikleri eklenebilecek türden yazmak gerekir.Aksi halde varolan kodlardan bazıları silinecekse bu değişikliğin test edilmesi ve ileride sorunlar oluşturabileceği göz önünde bulundurulmalı.

2-) Java Bean Scopes; Bean oluşturmak ve Lifecycle'ı yönetmek için için kullanılan yaklaşımlar.

*Singleton scope ; Bean eğer singleton scope ile tanımlanırsa o bean'den yalnızca tek bir adet initialize edilebilir. Singleton design pattern'la benzerdir. Bu bean in üzerinde yapılan değişiklik, bean in kullanıldığı her yere etki edecektir. Varsayılan(default) scope dur.

*Prototype Scope ; IoC container içersinde çağrıldığı her request te yeniden

oluşturulacaktır. Singleton'ın karşıtı gibi düşünölebilir.

* Request Scope ; Web uygulamalarında HTTP request geçerli olduđu sürece, geçerli olacak bir nesne oluşturulabilmesi için kullanılır. Kullanılması için ApplicationContext.xml gerekir.

*Session Scope ; Web uygulamalarında oturum, her bir HTTP session'ı için bir bean oluşturur.ApplicationContext.xml gerekir.

*Global Session Scope ; Standart HTTP session la aynıdır. Portlet tabanlı web uygulamalarında kullanılır. Bu tüm portletlerin global olarak saklanması sağlar. Standart servlet tabanlı bir uygulamada birden fazla kullanılabilirden hata oluşturmaz.

*Custom Scope ; Geliştirici tarafından tasarlanılabilir bir scope türüdür. Spring 2.0 ile birlikte gelmiştir.

* Application Scope ; Servlet context'in lifecycle' ı için bean örneđi oluşturur. Çoklu servlet tabanlı uygulamalar ile de paylaşılabilir.

*WebSocket Scope ; Tipik olarak singleton scope'a benzerdir. Herhangi bir websocket oturumundan daha uzun yaşar. Singleton davranışı sergiler. Yalnızca bir websocket session'ıyla sınırlıdır. ProxyMode Tanımı gerekir.

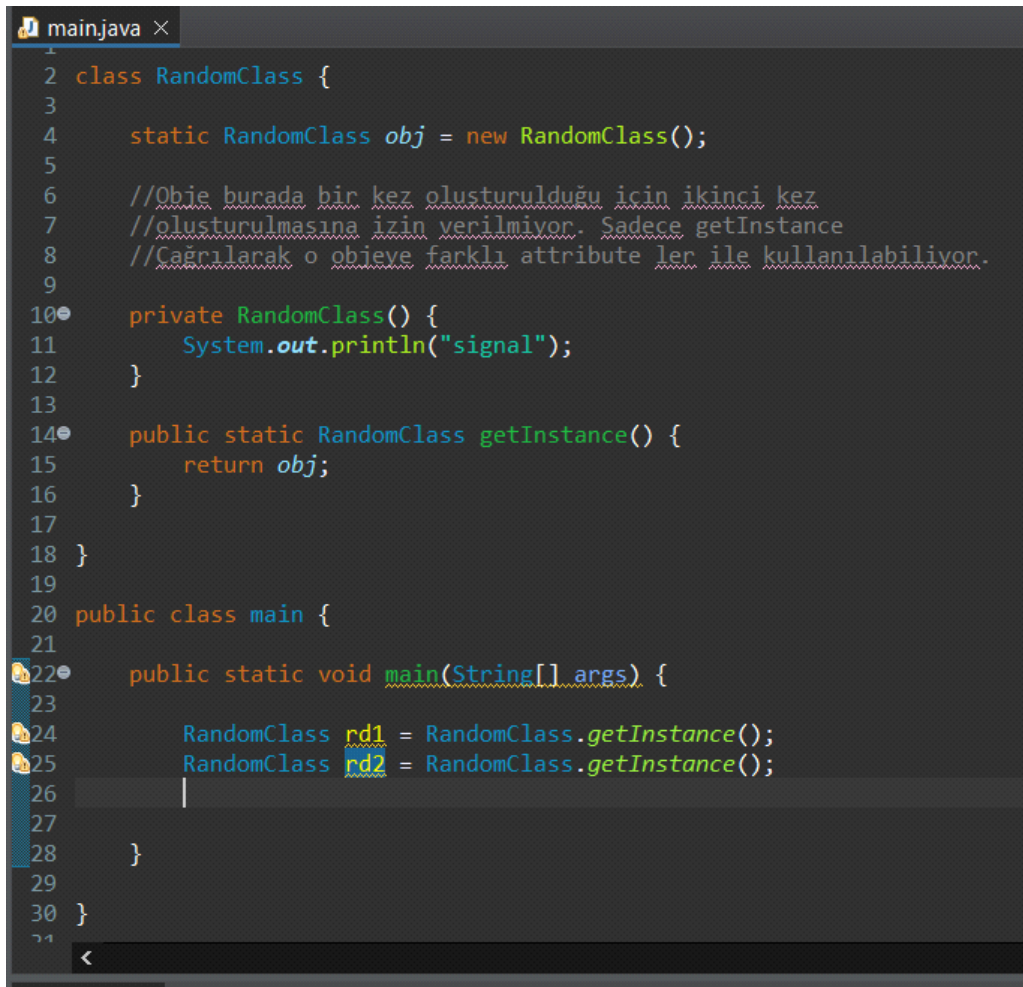
3-) @SpringBootApplication ; Spring uygulamasının giriş metodudur.@Configuration, @EnableAutoConfiguration, @ComponentScan anotasyonlarının üçünüde içerir.

4-) Why Spring boot over Spring? ; Spring boot, embedded container desteđi sağlar. Java'nın jar komutlarını kullanarak , bağımsız çalışmasını sağlar. Harici containerlar ile çalışıldığında olası jar çakışmalarını önleyebilmek için dependency'leri hariç tutar.

Deploying esnasında aktif profilleri belirtme seçeneđi vardır. Integration test için rastgele bağlantı noktası oluşturur.

Otomatik yapılandırma özelliđi sayesinde uzun kod yazmamızı sağlar. Gereksiz yapılandırmalara gerek bırakmaz.

5-) Singleton(Design pattern) ; Bir sınıf tipinden yalnız tek bir nesnenin oluşturulmasına izin vermek olan tasarım kalıbıdır. Mesela bir sunucu yapısını ele alalım. Bu yapıdan sadece bir tane olsun ve birden fazla create edilemesin veya bu create edilmeye bir limit belirleyebiliriz. Bir classtan kaç tane obje üretebilme konusunu yönetebilmeye de yarar.



```
1  main.java x
2  class RandomClass {
3
4      static RandomClass obj = new RandomClass();
5
6      //Objeye burada bir kez oluşturulduğu için ikinci kez
7      //oluşturulmasına izin verilmiyor. Sadece getInstance
8      //Çağrılarak o objeye farklı attribute ler ile kullanılabilir.
9
10     private RandomClass() {
11         System.out.println("signal");
12     }
13
14     public static RandomClass getInstance() {
15         return obj;
16     }
17
18 }
19
20 public class main {
21
22     public static void main(String[] args) {
23
24         RandomClass rd1 = RandomClass.getInstance();
25         RandomClass rd2 = RandomClass.getInstance();
26
27     }
28
29 }
30 }
```

6-) @RestController anotasyonu ; @Controller ve @ResponseBody sınıflarının oluşmasını sağlayan anotasyondur.RESTful hizmetlerinin oluşturulmasını sağlar. Spring 4.0'da tanıtılmıştır.

7-) Biggest Differences; Spring framework'ün en önemli özelliği dependency injection. Spring boot'un ise AutoConfiguration.

Spring, kurumsal uygulama geliřtirmeye ynelik, open source bir frameworktr. Boot ise REST API'leri geliřtirmek iin yaygın olarak kullanılan frameworktr.

Ana hedefi Web uygulamalarıdır.

8-) VCS ; Birden fazla developer'ın designer'ın ve team member'ın aynı proje stnde birlikte alıřmasına olanak tanır. Herkesin gncel koda ve deęiřikliklerin izlenebilmesine imkan saęlar.

9-) S.O.L.I.D. ;

***S (Single Responsibility Principle) ;** Her sınıfın veya metodun tek bir sorumluluęu olmalı. Yapması gereken sadece bir iři olması.

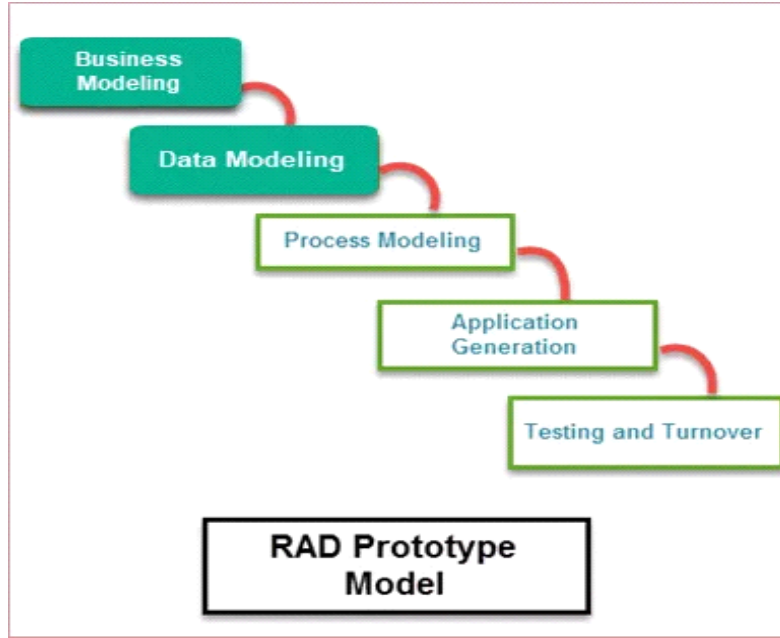
***O (Open-Closed Principle) ;** Sınıflar deęiřime kapalı ancak geliřime aık olmalıdır. Sınıflar varolan zelliklerini korumalı ve yeni zellikler kazanabiliyor olmalı.

***L (Liskov's Substitution Principle) ;** Kodda herhangi bir deęiřiklik yapmaya gerek duymadan, alt sınıfları tredikleri st sınıfların yerine kullanabilmeliyiz. Yani base class'ın attribute'unu kullanan metodlar bu sınıftan tremiř olan sınıfları ekstra bilgiye ihtiya duymaksızın kullanabilmeli.

***I (Interface Segregation Principle) ;** Sorumlulukların hepsini tek bir arayze toplamak yerine daha zelleřtirilmiř birden fazla Interface yazmak. Dummy code oluřmasını engellemek.

***D (Dependency Inversion Principle) ;** Sınıflararası baęımlılıklar olabildięince az olmalıdır. zellikle st seviye sınıflar alttakilerle baęımlı olmalıdır. Baęımlılıklar sadece abstract kavramlar olmalıdır.

10-) R.A.D. Model (Rapid Application Development); Hızlı uygulama geliřtirme modeli. Prototiplemeyi esas alan bir yazılım geliřtirme metodolojisidir. Planlamaya daha az dikkat edilir. Mřterinin gereksinimlerinin toplanması, prototiplenmesi ve erken test edilmesini hedefler.



11-) Spring Boot starter; Spring boot, uygulamaları içinde barındırdığı pom.xml dosyasındaki dependency'leri okuyarak ayağa kalkar.pom.xml' eklenen dependency'lere bağlı olacak şekilde farklı şekilde de çalışmaya başlayabilir.

özellikleri;

*Yapılandırma süresini azaltabilir.

*Eklenecek dependency'ler azaldığı için pom.xml'i yönetmek daha kolay olacaktır.

*Dependency'lerin adı ve sürümünü hatırlamaya gerek yoktur.pom.xml sırasıyla okur.

12-)What is spring boot Annotations ?;

@Bean ; Bir metodun spring tarafından yönetilen bir bean ürettiğini belirtir.

@Service; Belirtilen sınıfın bir servis sınıfı olduğunu belirtir. Projenin Bussiness Logic kısmında kullanılır.

@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 eşleşmesini yapar.

@Autowired ; Constructor, variable yada setter metodlar içi DI işlemi gerçekleştirir. Bu ifadeyi kullanmak için her iki sınıfta bean sınıfı olmalı.

@SpringBootApplication; Spring boot AutoConfiguration ve component taramasını aktif eder.

@Qualifier ; Eğer bir interface birden fazla sınıf tarafından implement edildiyse bu sınıflardan hangisine eşit fonksiyonun kullanılacağını bilmek için @autowired tanımlanmasına ek olarak @Qualifier("SınıfAdı") adında bir anotasyon daha eklendiği takdirde yine new ile o sınıftan özel bir nesne türetmeye gerek kalmadan, o sınıfın fonksiyonunu kullanabiliriz.

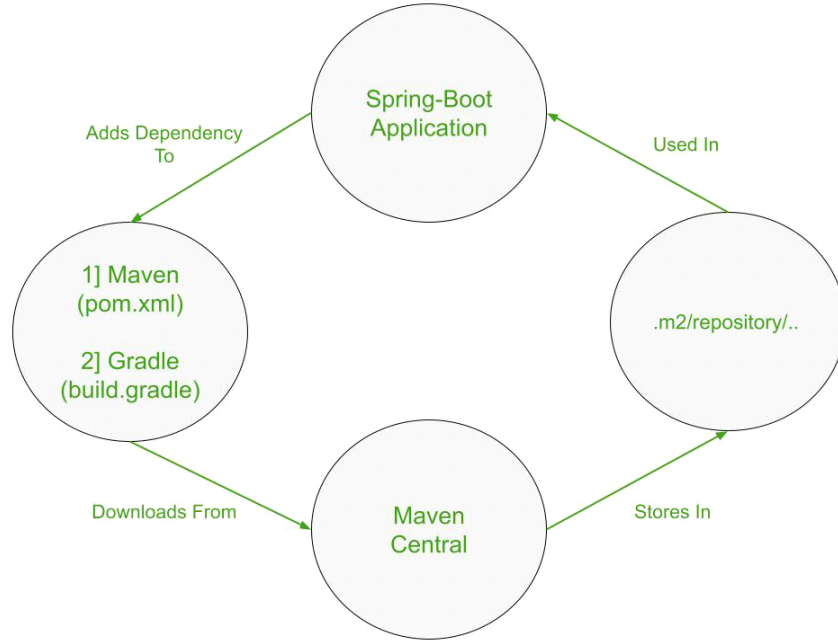
@Component ; Bir sınıfı bean olarak işaretler. Bu sayede Spring'in component tarayıcısı Bu sınıfı ApplicationContext'e ekler. Projedeki tüm component ifadelerini tarar. @Repository, @Service, @Configuration, @controller anotasyonlarını barındırır. Class seviyesinde bir anotasyondur.

@EnableAutoConfiguration ; Spring'in classpath'inde bulunan jar dosyalarına dayalı yapılandırmayı tahmin etmesini sağlar. Hangi kütüphaneyi kullandığınızı anlar. Bileşenleri önceden yapılandırabilir. Tüm spring-boot-starter kütüphaneleri böyle çalışır.

@ComponentScan; Component için bir veya daha fazla paketi / alt paketi tarar.

@Required ; Setter metodunun yapılandırma zamanında bir değerle DI şeklinde yapılandırması gerektiği söylenir. XML aracılığıyla doldurulan dependency'leri işaretlemek için kullanılmaz ise "BeanInitializationException" exception'ını fırlatır.

13-) What is Spring boot dependency management? ; Autoconfiguration sayesinde yapar. Çalıştığımız uygulamada kullandığımız kütüphaneleri yönetmeyi amaçlar. Bunun için kullanılacak kütüphaneleri pom.xml'e ekleriz pom' a eklenen dependency ler Maven Central' dan indirilecektir. İndirilen dependency'ler .m2 klasöründe depolanır. Spring boot, bu dependency'lere .m2 den erişir.



14-) What is Spring Boot Actuator? ; Spring boot framework'ün bir sub-projesidir. Spring boot uygulamamızı işlememize ve yönetmemize yardımcı olan bir dizi özellik içerir. Actuator, endpointsler içerir. Bu endpointer sayesinde uygulamamız hakkında bilgi edinebiliriz.

Id	Açıklama	Başlangıç Değeri
autoconfig	Tüm autoconfig tanımlamalarını gösterir	True
beans	Spring tarafından yönetilen tüm beanleri gösterir	True
dump	Thread dump almayı sağlar.	True
env	Spring's ConfigurableEnvironment değerlerini gösterir.	True
health	Uygulama sağlığını gösterir.	False
info	Uygulama bilgilerini gösterir.	False
loggers	Uygulamada kullanılan log bilgileri gösterir.	True
mappings	@RequestMapping tanımlaması yapılan değerleri gösterir.	True
trace	En son kullanılan 100 HTTP isteklerini listeler.	True

Author ; Birkan Yaşar

Date ; 16.07.2022

mail | birkanyasar.366@gmail.com

References;

<https://baeldung.com>

<https://bilgisayarkavramlari.com>

<https://docs.spring.io>

<https://geeksforgeeks.org>

<https://medium.com>

<https://javapoint.com>