**@Bean, @Configuration, @ComponentScan and @Import**

**@Bean**

• **org.springframework.context.annotation.Bean** is an annotation used for methods.

• **@Bean** makes a method a factory to produce a bean to be managed by

the **Spring** container.

• No need to annotate the classes whose instances will be created in

**@Bean-annotated** methods with **@Component.**

• **@Bean** provides the same semantics as **</bean>** in XML file.

Xml’de bean elementi yerine burada @Bean annotation sahip bir method oluyor, objeyi olusturup geri dönderiyor.

<https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/context/annotation/Bean.html>

@Bean

public MyBean myBean() {

// instantiate and configure MyBean obj

return obj;

}

Böyle obje olusturdugumuz zaman birer bean ama artık **@Component** ile annotation etmeye gerek yok. Çünkü o objeleri nasıl olusturacağımıza java kodunda biz karar veriyoruz. O yüzden **java-Based Configuration** diyoruz.

• **@Bean-annotated** methods can be declared in any Spring **@Component** in which case a bean object produces another bean object.

• However, they are most often used with **@Configuration** beans.

• Because it is best to put together factory methods into separate configuration classes annotated with **@Configuration**.

Herhangi bir spring @Component ‘da tanımlanabilir. @Component’lerin icine koyabiliriz.

@Component ile annotate edilmis class’ı factory yapıyoruz onun icindeki method da factory method oluyor.

Genellikle @**Component** icine konulmaz cünkü çok fazla @**Component** icinde beanler yapmak yerine @**Beanleri** tek bir classa toplarız, ona da @**Configuration** deriz ve bununla annotate ederiz. O zaman bütün factoryler bir arada toplansın. Yani bütün factoryler @**Configuration** annotate sahip sınıflar olsun.

**@Configuration**

• **org.springframework.context.annotation.Configuration**

is an annotation used for classes.

• **@Configuration** indicates that a class declares **@Bean** methods.

• It is processed by the container to generate bean definitions and

service requests for those beans at runtime.

• **@Configuration** classes are typically bootstrapped using

either **AnnotationConfigApplicationContext** or its web-capable

variant, **AnnotationConfigWebApplicationContext**.

Bean definitionları üretir, hangi beanleri ver gibisinden isteklere de cevap verir. A beani ya da B beani hangisi istersen ona.

**AnnotationConfigApplicationContext** bununla ayağa kaldırabiliriz.

• **Configuration** has two attributes:

• **value** is a **String** and represents **Configuration** name.

• **proxyBeanMethods** is a **boolean** which is **true** in default.

İki tane attribute vardır.

Value’da configuration’un ismi verilir.

**proxyBeanMethods** temelde @Beanli methodları override edip onun yerine proxyler kullanılabilir.

**Bootstraping @Configuration Classes**

• There are several ways to register configuration classes:

• **AnnotationConfigApplicationContext** is used to bootstrap a stand-alone context that uses annotations.



• The XML configuration file can be used to register configuration classes.



• Through component scan.

@Configurationı biz configurationun parçası olarak o da bir component oldugu icin ona componentscan ile ulasabiliriz.

Xml icinde ulasabiliriz.

Bean icinde register edip ulasabiliriz.

**AnnotationConfigApplicationContext**

• **org.springframework.context.annotation.AnnotationConfigApplicationContext** is a class that implements both **BeanFactory** and **ApplicationContext** interfaces.

• It accepts component classes such as **@Component** beans and others

produced by **@Configuration-annotated** classes.

• **AnnotationConfigApplicationContext** also accepts plain types

and JSR-330 compliant classes using **javax.inject** annotations.

**ApplicationContext**  daha yetkin hali.

1. Component isliyor
2. @Configuration isliyor
3. @Component typelarını isliyor.
4. and JSR-330 compliant olan sınıfları da isliyor. İnject ile alakalı.

• **AnnotationConfigApplicationContext** can register component classes and beans.

• It can also scan packages.

• It has constrcutors and methods to do these.

Bütün bunları yapmak icin constructors ve methodlara da sahiptir.

**@ComponentScan**

• **org.springframework.context.annotation.ComponentScan** is an annotation that provides component scanning directive for use with all **@Component** classes.

• **@Configuration** is a kind of **@Component** and therefore is subject to classpath scanning.

• **@ComponentScan** has an attributes for packages to scan.

• In default it starts scanning from the package the **@ComponentScan** resides.

Bu xml’den kurtulmamızı sağlayan bir yapı. @Configuration annotate olan class’ları component scanning (classpath scanning)s ile bulmamıza yarayan annotation.

**@Import**

• **org.springframework.context.annotation.Import** is an annotation to import one or more component classes.

Bir ya da daha fazla component classını dolayısıyla configuration classını scan edebiliriz.

**@Bean and @Configuration**

• The **@Bean** annotation doesn’t have any attribute for profile, scope,

lazy, depends-on or primary.

• @**Scope, @Lazy, @DependsOn, @Primary and @Qualifier**

annotations should be used with @Bean to get the necessary effect.

• If **@Lazy** is used with **@Configuration** then all beans produced with

**@Bean** methods will be initialized lazily.

@Bean’in attirubute yok.

Dolayısıyla @Bean annotate olusturdugu bean üreten methodların ürettikleri beanlerle alakalı farklı annotation kullanarak davranıs ve özelliklerini degistirebiliriz.

Örnek olarak Bean’in scope’nu degistereceksek @**Scope,** lazy loading yapmasını sağlamak icin **@Lazy** bir baska beane olan DI yönetmek icin **@DependsOn, veya @Primary and @Qualifier** gibi annotation’ları @Bean ile kullanabiliriz.

**Bunlar @Configuration ile kullanılmaz, tek tek @Bean’lerle kullanılması gerekiyor.**

**Ama @Configuration ile ayrı kullandığımız @Lazy vardır. Bunu kullandığımız zaman bütün beanleri lazy’li initalize eder.**

**@Profile**

• A class that is annotated with **@Configuration** can have a

**@Profile** annotation too.

• In this case all of the **@Bean** methods and **@Import** are associated with

specified profiles.

@Configurationun kendisini profile verebiliriz.

Dolayısıyla Profile icindeki bütün bean metotları, import ettiği diğer configuration icin de bu profile gecerli olur.