**@Component and @ComponentScan**

• **org.springframework.stereotype.Component** is an annotation

that indicates that the annotated class is a **component**.

• **Component** is placed before class declaration and makes its instances beans.

• Such classes are considered as candidates for auto-detection when

using annotation-based configuration and classpath scanning.

• Classpath scanning for components is possible by

**<context:component-scan/> in <beans/> in XML file.**

Component, XML’de ifade ettiğimiz beanlerin aynısı.

Spring beanini annotation ile göstermenin yolu @Component’dir.

Sınıflardan önce kullanır, o sınıfın instancelarını bean yapar.

Yani Spring’in o sınıfın objelerini olusturur, inject etmeye ya da inject edilmeye hazır hale getirir.

Otomatik olarak bulunması icin aday olarak gözükür, classpath scanning enable ve annotation-based configuration kullandığımız zaman beanleri bulup, objelerini olusturup, birbirlerine inject etme amacıyla spring bunları kullanır.

**Scanning Components**

• **<context:component-scan/>** element has an attribute called

**base-package** to specify where to start scanning beans.

• Using base-package is mandatory and it receives a comma/

semicolon/space/tab/linefeed-separated list of packages to scan for annotated components.

• Spring scans all of the packages and their sub-packages listed in

base-package for annotated components that will be auto-registered as beans.

Component scanning o componentleri bulması icin nereden baslaması gerektiğini de **base-package** vermemiz lazım. **Mandatory’dir**.

**Beanler** otomatik olarak kaydedilecek ve objeleri olusturulacak olan sınıfları **@Component** ile yaptığımız zaman spring gidip onları bulur.

• Component scan and use of Component annotation make XML

configuration files free of any bean definitions and keem them very short.

**• <context:component-scan/>** implicitly enables the functionality of

**<context:annotation-config>** which enables the use of Autowired annotation.

• So no need to include the **<context:annotation-config>** element

when **<context:component-scan/>** is used.

@Component veya @ComponentScan ile XML icerisinde bütün bean definitionsları cıkarır.

**<context:component-scan/> bunu** bunu configte kullandığımız zaman otomatik olarak Autowired de enable oluyor.

**Component Continue**

• Component has only one attribute called value and it takes a String

for the name of the bean.

• If called value is not used Spring uses as default name the simple

name of the class of the bean with its first letter converted to lower camel case.



Componenti kullandığımız zaman burası tek bir değer alıyor o da **value**, **value** da componentin adıdır. (3.görsel.)

Hic isim vermezsek 2.görseldeki gibi yazabiliriz.

Eğer isimlendirme vermezsek default olarak bağlı olduğun class’ın name’nin camel case halini alır. (1.görsel)

• **Component** must be used for classes so that their objects can be created.

• So using **Component** for interfaces or abstract classes does not make sense.

• In fact Spring provides many different components for different tasks:

• **Component, Repository, Service, Controller,**

**RestController, ControllerAdvice, and Configuration**

• These stereotypes are all annotations for Spring beans and they are

automatically detected using classpath scanning.

• We will discuss them in future.

Componentin bir çok sub typeları vardır yukarıda belirtilmistir.

Bütün bunlar classpath scanning ile bulunabilecek tipler. Objeleri olusturulup kullanmamızı sağlayacak alanlar.