**@Profile**

• **org.springframework.context.annotation.Profile** is used

to select components that are eligible for registration when one or

more **specified profiles** are active.

• **Profile** is a logical group that has a name.

• **Profile** can be considered as a specific configuration of the beans for a specific purpose.

• For different database servers,

• For development, production or test environments etc.

Farklı sebeplerle bir interface implement eden birden fazla bean oldugu zaman parent Dependency durumunda birden fazla aday bean durumunda Qualifier tipleri vardır.

Test amaclı ya da farklı environment varsa, farklı db server varsa onlara yönelik farklı beanlerin olusturulması söz konusu ise **@Profile**  söz konusudur.

Yani Db olarak mysql, postgre, mongo dblerimiz var. Bunların her biri ayrı ayrı configurationları vardır. Burada her ortam icin ayrı profile olusturup ve her o db de kendi beanlerını **@Profile**  annotate ederek su profile calıssın gibisinden istenilen db calıstırılır.

Dolayısıyla sisteme hangi profile verirsek sistem onu kullanıp ona göre conf yapacaktır.

• **Profile** annotation has only one attribute called **value** which is of type **String** array.

• This is the same as using profile attribute of **</beans>** with profile

names in XML file such as **<beans profile=“p1, p2">.**

• Profile annotation can be used as both at class level with

**@Component and @Configuration** annotations and to create custom profiles.

• @Profile has only one attribute, value which is of type String array.

• So @Profile may have more than one profile name.

• @Profile may have simple name or more complicated expression that

include to express some profile logic:

• !: A logical “not” of the profile

• &: A logical “and” of the profiles

• |: A logical “or” of the profiles

• Default profile, @Profile(“default”) is the one that is enabled by default.

• If no profile is active, default profile is in effect.

• If any profile is enabled, the default profile does not apply.

**• Custom profiles can be created using @Profile annotation.**

**Activating Profiles**

• Profiles can be activated in several ways:

• **setActiveProfiles()** method of

**org.springframework.core.env.ConfigurableEnvironment**

**• setting spring.profiles.active property as JVM system property**

**• using org.springframework.test.context.ActiveProfiles** annotation

• **ActiveProfiles** is mainly used in integration testing.