**Ordering Beans**

**Order For Initialization and Injection**

• There are mainly two kinds of order among beans:

• Initialization order that is mainly determined by dependency relations.

• Injection order

• Initialization order can be controlled by **DependsOn** annotation.

• There are several different methods to control injection order among beans.

Beanlerin sıralanması dediğimiz zaman temelde iki kavram gelir.

**Initialization order**

**• Injection order**

**Initialization Order** yani elimizde bir sürü bean var bınları hangi sırayla ayağa kaldıracağız, spring temelde aralarındaki iliskilere bakarak kendisi o iliskileri gözeterek kendisi bir sıra belirler. **dependency** **relations**.deriz.

Birçok beani bir yere inject ediyorsak, hangi sırayla sırayla inject edilecek ?

Aaslında collection gibi aynı cinsten beanlere göre sıralar inject eder

**Initialization Order’da DependsOn** annotation kullanarak kontrol sağlayabiliriz. Bu buna bağımlıdır tarzında relation olusur.

Injection Order icin ise birden fazla yöntem var.s

**Initialization Order and @DependsOn**

**@DependsOn**

• **org.springframework.beans.factory.annotation.DependsOn** to declare that a bean depends on another bean.

• **@DependsOn** is used to influence the initialization order of the beans.

• **@DependsOn** takes an argument **value** of type **String** array which

are the names of the dependent beans.

• If a bean declares **@DependsOn** to depend on another bean, IoC

guaranties that another bean is initialized before the bean that declares **@DependsOn.**

Temelde IO spring tarafından belirleniyor.

Hangi bean’in hangi beanden önce initialization edeceğini anlar.

İstersek beanler arası iliskileri @ **DependsOn ile kontrol ediyoruz.**

**Spring’in fark edemeyeceği çok dolaylı dependency’ler** varsa mesela bir read bir de write islemi var read ile write arasında bir dependency varsa @ **DependsOn kullanarak** write, reade dependstir yani bağımlıdır diyebiliriz.

**Doğrudan dependency varsa zaten buna gerek yok spring bunu anlıyor.**

• **@DependsOn** also influences the destruction order of the singletonbeans.

• If a bean declares **@DependsOn** to depend on another bean then the

destruction of the declaring bean would happen before the depended bean.

• **@DependsOn** doesn’t have any effect at the class level if componentscanning isn’t used.

**• Depends on relationship** can also be declared in XML file using depends-on attribute of **</bean>.**

• If a bean is declared with depends on relationship both in XML and with **DependsOn** annotation then **DependsOn** annotation is ignored and XML declaration is in effect instead.

• Be careful not to create circular dependencies!

**Injection Order**

• Injection order can be controlled by several mechanisms in Spring:

**• @Order**

**• @Priority of JSR-250**

**• Ordered interface**

• Injection order does not influence initialization order which is mainly determined by dependency and **@DependsOn** relationships.

Injection Orderı ise yukarıdaki 3 annotation ile kontrol edebiliriz.

Injection Order’ın Initialize order ile alakası yoktur.

**Burada collectiona birden fazla objeyi inject ediyorsak onlar hangi sırayla inject edilmeli ona bakacağız.**

**@Order**

• **org.springframework.core.annotation.Order** is an annotation to control injection order of beans.

• **@Order** is used to sort beans for example in a list using a specific order.

• **@Order** has only one attribute **value** which is an int.

• Its default value is **org.springframework.core.Ordered.LOWEST\_PRECEDENCE**

• Beans with lower values in **@Order** are injected before than beans with higher values in **@Order**

• Beans with the same order are injected with an arbitrary order.

• **order.org.springframework.core.OrderComparator** is responsible for this.

Ne kadar düşük ordera sahipse öncelikle o inject edilir.

Bununla iki tane field vardır.

**org.springframework.core.Ordered.LOWEST\_PRECEDENCE**

**org.springframework.core.Ordered.HIGHEST\_PRECEDENCE**

diye ama default olarak **org.springframework.core.Ordered.LOWEST\_PRECEDENCE**

geçerlidir.

**HIGHEST\_PRECEDENCE** İnteger’ın en eksisisini alıyor. 2 üzeri -31 alıyor.

**LOWEST\_PRECEDENCE** En yüksek değer, dolayısıyla en düşük öncelik oluyor.

Order integerları aynı olanlar icin **OrderComparator karar veriyor.**

**Ordered Interface**

• **order.org.springframework.core.Ordered is** an interface to control the injection order of the beans.

• It has only one method **getOrder**() which returns the order as an

integer.

• Ordering of injection using this interface is the same as @**Order**.

• Beans that return lower values are injected before than beans that

return higher values.

**@Priority**

• **javax.annotation.Priority** is an annotation that is part of JSR-250.

• **@Priority** is used to sort beans for example in a list using a specific order.

• **@Priority** has only one attribute value which is an int.

• **value** is given generally a non-negative value.

• Beans with lower values are injected before than beans with higher values.