**Life-Cycle Events and Notifications**

**Post-Initialization & Pre-Destruction**

• Two main notifications for beans are:

• Post-initialization and

• pre-destruction (or pre-destroy)

• Initialization of a bean includes creation of its instance and injection of all of its dependencies so that the instance becomes ready to provide service.

• Destruction of a bean is the destruction of its instance so it can not serve anymore.

Springde temelde beanler icin iki tane notifications vardır.

Post-initialization and

Pre-destruction (or pre-destroy)

Initialization aslında Beanin instance’ın olusturulması ve bütün Dependencylerin inject edilmesi, nullPointer almayacak sekilde tüm dependencylerin inject edilmesi anlamına geliyor.

Dolayısıyla objenin hazır hale gelmesi demektir. Initialization springde beanler icin tipik olarak bu demektir.

Destruction ise instance destroy edilmesi bundan dolayı da artık serve edemeyecek olması.

**Post-initialization**

• The post-initialization notification that is called right after the bean is

initialized with all its dependencies can be used to provide custom

actions after the initialized bean instance is available.

• The custom actions can not be done in constructors due to the fact that

some of the dependencies might be instance variables and properties

and for full initialization they need to be injected too.

• The custom actions can be something like notifying some other beans

regarding its readiness or setting up environment for further activities,

etc.

post-initialization Bir bean ayağa kalktığı zaman haber ver. Bir bean bir thread poolunu dependencylerle birlikte ayağa kaldıracak ve ayağa kalkar kalkmaz notification gecilmesi lazım bu durumda kullanılır.

İnitialization bittiği zaman haber ver diyebiliriz.

**Pre-Destruction**

• The pre-destruction notification that is called right before the bean is destroyed by the container can be used to clean up resources the bean may hold.

• Beans may hold references to other objects or resources outside the JVM.

• It would be nice if a bean frees all the resources it holds especially if they are not in control of JVM such as files.

Obje destroy edilmeden önce tipik olarak toplama yapsın, etrafına dağıttığı seyleri toplasın, tipik olarak JVM’in kontrol edemediği resourceları toplamasından bahsediliyor.

**Post-initialization & Pre-Destruction**

• If beans are loaded eagerly which is the default case for singletons then even though the beans are not asked from the context they are initialized and post-initialization events are raised.

• But in case of lazy loading, the beans are only initialized when they are asked from the context and only that time post-initialization events are raised.

Eğer beanler eagerli load ediliyorsa beanler contexten alınmasa bile, initalize edilmese bile post initialization eventleri olusturulur.

Aama lazy loadingde lazy initalize edince bu eventler fırlatılır.

• While the post-initialization event is called regardless of the scope of the bean, the pre-destruction event is called only on singleton beans, container doesn’t call this event for prototypes.

• Prototype beans can receive the pre-destruction events via a custom post processor such as **DestructionAwareBeanPostProcessor**.

Beanin scope bağımsız calıstırlırken, pre-destruction ise sadece singleton beanler de cagırılır. Otomatik olarak. Prototypelarda container otomatik olarak bunları cagırmaz. Temelde prototype böyle seyleri otomatik cagırmak dayanılmaz durumlara yol acar.

**DestructionAwareBeanPostProcessor bununla implement eden prototype lar üzerinde otomatik olarak cagırılabilir.**

**Singleton da zaten otomatik olarak cagırılıyor.**