**Extensions to ApplicationContext**

• IoC has several interfaces that can be used to extend

**ApplicationContext** object.

• They are called **processor**.

**• Two main processors are:**

• **BeanPostProcessor**

• **BeanFactoryPostProcessor**

• They are in **org.springframework.beans.factory.config** package.

AC’ extend etmeye ona farklı yetkinlikler kazandırmaya yarayan bir yapı.

IOC’nin bir özelliği olarak hook pointler var bunlar tipik olarka interfacelerdir. Bunları implement ediyoruz ve IOC ne zaman cagıracağını biliyor. IOC zaten budur.

Çok fazla processor vardır ama en sık kullanılan iki tanesi ilgilenecek.

Sonraki islemci, beani ve beanin factorysini islemek icin.

**Processors**

• **BeanFactoryPostProcessors** run before **BeanPostProcessors**.

• Processors are autodetected by **ApplicationContext** if they are in classpath.

• If they are created by a **@Configuration** class, **@Bean** methods that

return their instances should be **static** to avoid the initialization of

other beans in other **@Bean** methods of the same class.

**BeanFactoryPostProcessors, BeanPostProcessors’dan önce calısıyor.**

Factory isleyen processorlar, bean isleyenlerden önce calısıyor.

Çünkü factory islemek icin beani etkilemek icin olusturma sürecini vs gerekiyor. Bean önce calıssaydı factory islesek bile bean zaten olustugu icin etki olmayacak.

Bunlar AC tarafından bulunuyorlar. Classpath kullanıyorsak.

@Configuration classı ile de olusturabiliriz, @Bean ile yapabiliriz. Böyle olusturduğumuz zaman bean metotları static yapmalıyız çünkü bunların olusturulması aynı @Configuration classındaki instanceların, diğer instance metodu olan @bean metotların da cagırılmasını engellemek icin.

**Static** yapınca configurationdaki beanlar daha önce olussunlar diğer beanlerin olusmasını tetiklemeyelim diye.Çünkü ihtiyaca göre bunları olusturuyor. **Bu yüzden static yapmak gerekiyor.**

**BeanPostProcessor**

• **org.springframework.beans.factory.config.BeanPostProcessor** allows custom modification for beans.

• It has **two** **default** **methods** that can be implemented by choice:

**• default Object postProcessAfterInitialization(Object bean, String beanName)**

**• default Object postProcessBeforeInitialization(Objectbean, String beanName)**

Yani beanleri istediğimiz gibi custom yapmak icin kullanırız.

• **postProcessBeforeInitialization()** and

**postProcessAfterInitialization()** are called just before and

after any bean initialization callbacks such as a method with

**@PostConstruct** annotation or **InitializingBean’s**

**afterPropertiesSet()** method.

• When these methods are called the beans had already been created

and populated with dependencies.

Bunlar hemen öncesinde hemen sonrasında initialization callbacks de cagırılır.

Hemen öncesinde **postProcessBeforeInitialization hemen sonrasında postProcessAfterInitialization** cagırılıyor.

• Any custom logic such as programmatically setting some dependencies

right after the beans instance are created and just before any

initialization callbacks are called can be implemented in

**postProcessBeforeInitialization**() method.

• Custom logic such as checking if all bean dependencies are met or

making some changes to bean or its dependencies after the

initialization can be implemented in

**postProcessAfterInitialization**() method.

**BeanFactoryPostProcessor**

• **org.springframework.beans.factory.config.BeanFactoryPostProcessor** allows custom modification for bean factory.

• It has one abstract method:

• **void** **postProcessBeanFactory(ConfigurableListableBeanFact**

**ory beanFactory)**

• It can be used to get information for example about all registered beans

or override some of the properties of bean factory.

Bean factoryi özelliklerini degistirmekte kullanırız.

• Notice that when **postProcessBeanFactory**() is called no

registered bean has not been initialized yet.

• So **postProcessBeanFactory**() should be used to modify bean

definitions not bean instances because of the fact that it is called before

any bean initialization and trying to initialize a bean inside it may result

in premature bean instances.

Burada bean definitionlarla, configurationla ilgileniliyor, beanlerle ilgilenilmiyor.

**Çünkü bu çağırıldığı zaman beanler henüz initialize edilmis değil, çünkü bean olusmadan önce configuration degistirelim ki beanleri etkileyebilelim, ve beanler buna göre olussun ama bean bunlardan önce olusursa beani etkileyemez cünkü bean zaten olusmustur.**