**Spring IoC Container**

Spring frameworkte kendi containeri var.

Java EE’nin temelde iki tane container vardır. Servlet Container ( web container), EJB container bunlar esittir Application server.(Servlet + EJB = Application server)

Springde objeleri olusturma, lifecyclerı yönetmek, dependencyleri yönetmek gibi islemleri Spring’in IoC containeri yapıyor.

\*\*\*Container icin bir kaba benzetebiliriz, bu kap ayağa kalkarken bu kap icinde yasayacak olan objelere hizmet verecek, ne hizmeti dersek , objeleri olusturma, birbirini inject etme.

**Main Objects** (**örnekler IoC projesi icinde beanFactoryExample paketi icinde. Hepsi)**

**Bean**

Spring’in IoC containeri tarafından yönetilen objelere **Bean** deriz.

Bu anlamda Spring IoC container bean’in objelerini olusturur onları assembled yani bir araya getirir ve bir sekilde onları yönetir (**lifecyclerını**).

Spring’in yöneteceği beanleri dependencyleri yönetme, cok fazla event ya da loglarla alakalı karmasık bilgileri **configuration metadata ile containere sağlanır**.

**Container da configuration metadatayı kullanarak, bizim isimizi bizim icin yapar.**

\*\*\*\*Spring’in bir applicationdaki bütün beanleri spring tarafından yönetilmesine gerek yoktur çünkü spring tarafından olusturulmayan beanler spring tarafından yönetilmezler.

**JavaBean ile Bean farklı kavramlardır.**

**Su sekilde JavaBean dedigimiz zaman default constructora sahip olan, varsa bütün fieldları private olan, set-get metotlara sahip olan beani kast ediyoruz.**

**Ama buradaki Bean öyle degil illa default constructora ihtiyac yok, constructor inject yaptığımız zaman constructor olmayabilir.**

**Bean Factory**

Spring Ioc’nin temel interface **org.springframework.beans.BeanFactory** paketi altındaki BeanFactory’dir.

Spring’in DI yapısı tamamen beanFactory ve onun altındaki subtypler tarafından olusturuluyor.



Bean’leri olustururken bizim o beani getir derken eger bulamazsa böyle bir bean bulunamadı exceptionı fırlatır yani ( **NoSuchBeanDefinitionException**)

Eger eslesecek bir bean bekleniyorsa ama birden fazla eslesen bean adayı geliyorsa

**NoUniqueBeanDefinitionException** fırlatılır.

Bütün application icerisinde spring’in olusturdugu bütün beanleri yönetmek icin olusturdugu merkezi kasadır.(**BeanFactory**) ama beanFactory cok da yetkin değil. Birden fazla bean getiremiyor.

**ListableBeanFactory**

Listenebilen beanFactory. BeanFactory altındaki subclasslardan bir tanesi.

Enumeration yetkinliği verir.

Yani arrayler olarak bize geriye döndürür.

Az önce beanFactory’nin yapamadıgı birden fazla beani getirme kısmını halledebiliriz.

**Bir bir vermek yerine bu tipten bütün beanleri ver diyebiliriz.**

**ApplicationContext**

Temelde containerı beanFactory’den daha yetkin sekilde temsil eder.

**org.springframework.context.ApplicationContext** is a sub interface of **BeanFactory** and adds more application-specific functionalities such as loading the configuration for beans because it is a **ResourceLoader**.

• It is the parent of application-layer specific contexts such as the **org.springframework.web.context.WebApplicationContext** for web applications.

Farklı farklı application türlerinde farklı implementleri vardır. Yukarıdaki webAppcontext gibi.

ListableBeanFactory vs. lerde resource yükleme gibi bir islev yoktu, örneklerdeki gibi biz kendimiz veriyorduk xml pathini vererek. Bu beanFactoryler bu isi yapmıyor.

**ApplicationContext objeleri bu isi yapıyor.**

****

•**ApplicationContext** is used more frequently than BeanFactory.

• The terms **application** **context** or **context** refers to the configuration of Spring IoC that ApplicationContext represents.

• It also provides event publication and message resource handling for internationalization.

ApplicationContext , BeanFactoryden daha fazla kullanılır.

Eventler, handler ile alakalı birçok yetkinliği var.

**ApplicationContext** has two main implementations for stand-alone applications,

**ClassPathXmlApplicationContext** and **FileSystemXmlApplicationContext**

• In other kinds of applications such as web different context objects and ways of configuration are available

\*\*\*Notes: BeanFactory ve Application kavramsal olarak aynı isi yapıyor ama biraz daha spesifik baktığımız zaman ApplicationContext daha yetkin ve resourceları da bildigi icin cogunlukla bundan bahsedilir.



They are both standalone XML application contexts that take bean definitions from classpath or file system, respectively.

• **ClassPathXMLApplicationContext** takes the context definition files in classpath using **packageName/resource.xml** style.

• **SystemFilePathXMLApplicationContext** takes the context definition files in file system using absolute path with **file:** prefix or relative path when used without any prefix.



File türünden prefix vermezsen hata verir.( **file:** prefix)

* Methods to get bean instances from ApplicationContext are the ones inherited from BeanFactory:



* To get all beans instances specific to a type when there is a possibility that many beans may have been defined use **ListebleBeanFactory** instead.



**Spring Definition Files**

• Beans can be defined either in XML files or properties files.

• **XmlBeanDefinitionReader** and

**PropertiesBeanDefinitionReader** can be used for this purpose.

• They are both in **org.springframework.beans.factory.support**

package and extends **AbstractBeanDefinitionReader**.

• They both load bean definition files via **loadBeanDefinitions**()

methods.

**XmlBeanDefinitionReader** ile XML file’ları yükleriz.

**PropertiesBeanDefinitionReader** properties file’larını yükleriz. Beanleri ve aralarında DI’ları ifade edebiliriz.

Bunların ikisi de **org.springframework.beans.factory.support** bu paketin altında. Ve **AbstractBeanDefinitionReader** extend ediyorlar. Üzerlerindeki **loadBeanDefinitions metotları ile de resource’ları yüklemeye izin veriyorlar.**

• They are typically used with **DefaultListableBeanFactory**, which is a **BeanDefinitionRegistry**.



Bunlar tipik olarak **DefaultListableBeanFactory ile** kullanılıyor, bu en temel implementation.

Spring’in IoC container altında kaynakları yükleyen ve fullfletch her türlü fonksiyonlara izin veren, ne kadar bean var, hangi beanden kac tane var, tek tek ya da array alabilir miyiz gibi bir cok seyi sorgulamaya yarayan en genis hali. Bu nihayetinde **BeanDefinitionRegistry’dir.**

**DefaultListableBeanFactory**

• **org.springframework.beans.factory.support.DefaultLis tableBeanFactory** is a sub interface of **BeanFactory** and **ListableBeanfactory**

• It is a full-fledged bean factory based on bean definition metadata that provides many functionalities on beans and dependencies.



* **BeanFactory** hieararchy is as shown.
* **org.springframework.beans.factory.xml.XMLBeanFactory is deprecated.**

**XMLBeanFactory deprecated edildiği icin kullanmıyoruz.**



**Configuration Metadata**

• The **Spring** IoC container gets its instructions to instantiate, configure, and assemble beans by reading **configuration** **metadata**.

• **Spring** configuration consists of at least one and typically more than one bean definition that the container manages.

****

**POJO’lar** var **bean** dediğimiz, bunları **Configuration** **Metadata** ile birlikte **Spring** **Containere** beslersek, elimizde **objeler** var onlar hakkında onların ötesinde **Configuration** **Metadata** verdiğimiz **zaman** **spring** bize kullanılmaya hazır **bean** **graph’ını** sunar.

**Bean** **olusturuyordu**, **DI** **yapıyordu**.

Bir ya da daha fazla **bean** **definition**, **bean** **tanımı** sahip olan **yapı** **Spring’in** **containerı** bunu **halledecek**, buna **configuration** **metadata** diyoruz.

**Configuration** **Metadata’yı** springte göstermenin 3 tane yolu var.

1. XML
2. Java Annotations ile
3. Java kodu ile yapmak.

• XML-based configuration metadata configures beans as <**bean**/> elements inside a top-level <**beans**/> element.

• @**Component** and other annotations are used to designate beans.

• Java configuration typically uses @**Bean**-**annotated** methods within a @**Configuration** class.

**XML-based configuration metadata configures beans isimli bir kök elementin root’ın icinde beanlerden ibarettir.**

**Annotationslarda @Component ve diger annotationslar var.**

Bunlar neticesinde Spring apisinde BeanDefinition icinde gözüküyor.