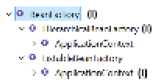
What is the difference between BeanFactory and Applicationcontext container? (only for interview)

BeanFactory container	ApplicationContext container
a) It is the object of a class that implements BeanFactory(I) directly or indirectly eg: XmlBeanFactory	(a) It is the object of a java class that implements ApplicationContext(I) directly or indirectly eg: FileSystemXmlApplicationContext, AnnotationConfigApplicationContext
b) It is subset of ApplicationContext Container	(b) It is super set of BeanFactory Container
c) Does not support pre-instantiation of singleton scope spring beans (only Lazy instatiation)	(c) supports (Supports eager instantiation for singleton scope spring bean)
(d) There is no support for properties file and place holders ${<key>} directly (some additional configurations are required)	(d) There is direct support for properties file and place hodlers ${<key>}
(e) No support for Internationalization(I18n)	(e) supports the Internationalization (I18n)
(f) Does not support the event handling	(f) supports
(g) Does not give direct support for annotation driven cfgs and 100% code driven cfgs	(g) supports
(h) allows only xml file as the spring bean cfg file	(h) Allows xml file as the spring bean cfg file and @Configuration class (java class annotated with @Configuration) as the configuration class (required in 100% code driven cfgs and spring boot programming)
(i) can not think about using in Spring boot Programming	(i) Required in spring boot Programming
(j) Suitable for Memory and Resource SEnsitive applications (If less memory is there for app to execute)	(j) Mostly used spring container in most of the spring applications and in spring boot apps
(k) This container can not be stopped or closed	(k) This container can be stopped or closed or refreshed or ....
(l) Resource obj is required to pass spring bean cfg file (xml file)	(l) we can pass spring bean cfg file (xml file) directly while creating this IOC container No need of Resource obj)
(m) Lightweight container	(n) Bit Heavy weight container compare to BeanFactory


BeanFactory (I)
HierarchicalBeanFactory (I)
ApplicationContext
ListableBeanFactory
ApplicationContext (I)

note:: BeanFactory Container means it is the obj of a class that implements BeanFactory(I) directly or indirectly

ApplicationContext container means it is the object of a class that implements ApplicationContext(I) directly or indirectly

Application context Container:: Bean Factory Container++

note:: Using BeanFactory Container we can not keep track of when IOC container is started , stoped/closed.. (No support for event handling) Using ApplicationContext Container such event handling is possible

note:: For Bean factory container , we can give inputs only through xml file.. where as in ApplicationContext container we can give inputs to IOC container using only xml file or using xml +annotations or using only java code + annotations

note:: Both Spring/Spring boot frameworks use same IOC containers .. In spring programming we can use both BeanFactory and Application Containers ..where as in Spring boot programming we can use only ApplicationContext container

note: Compare to ApplicationContext IOC container the BeanFActory IOC container is light weight IOC container (Overall both the IOC containers are Light weight containers. In which BeanFactory container is more light weight)

Where should I use BeanFactory IOC container and where should I use ApplicationContext container?

Ans) => If spring is used in Memory Sensitive apps like mobile Apps , embedded System apps IOT apps and etc.. where 1 or 2 few extra kilo bytes also matters then prefer using BeanFactory IOC container ..

=> If spring is used in web applications, distributed Apps , enterprise apps and etc.. where Memory utilization does not matter then prefer using ApplicationContext container

note: spring is not popular in mobile , Embedded systme apps.. spring,spring boot frameworks known for developing enterprise apps .. so prefer using ApplicationContext container as u r first choice container

web applications distributed App

note:: In latest Spring, Spring Boot Projects always prefere working with ApplicationContext Container , if u r not using that container u must have proper reason to convience u r self,,

Q) what is the difference between IOC and Dependency Injection?

Ans) IOC is the software specification that gives set of rules and guidelines to manage the dependency among the target and dependent classes.. Dependency Injection and Dependency Lookup are the two implemetation models of IOC specification

IOC is like blue print/theory / plan / for managing the dependency where as Dependency lookup(DL) and Dependency Injection (DI) are the two implementation models/praticals of IOC specification

Q) For IOC container of 100% Code driven cfgs , If we can not give certain inputs or instructions through Java code and annotations then how to pass such special insturctions to IOC container ?

Ans) we can give these special instructions to IOC container through xml file (spring bean cfg file --generally applicationContext.xml) by linking the file with @Configuration class using @ImportResource annotation

applicationContext.xml (com/nt/cfgs) (spring bean cfg file)

```
------------------------------
 ... ....   ....     ....
....  ..... ...      ....
```

AppConfig.java (Configuration class)

```
@Configuration
@ImportResource("com/nt/cfgs/applicationContext.xml")
 public class AppConfig{
   ...
   ...
    ...
   ...
 }
```

To create the IOC container

```
=========================
AnnotationConfigApplicationContext ctx=
      new AnnotationConfigApplicationContext(AppConfig.class);
```

conclusion :: First prefer giving all inputs and instructions to IOC container using java code + annotations.. if they are not sufficient then think about going for spring bean cfg file (xml file) by linking with @Configuration class using @ImportResource annotation..

100% code driven cfgs approach = java config approach

note:: To link spring bean configuration file (xml file) with @Configuration class use @ImportResource annotation
To link one @Configuration class with another @Configuration class use @Import annotation

What is difference among the setter injection , constructor and field Injection?

Setter Injection	Constructor Injection	Field Injection
a)uses setter method of target class to assign given dependent value/object to target spring bean class obj	a)uses parameterized constructor for the same	a) IOC container directly access the filed /property (uses the reflection api) of target spring bean for injection activity
b) we need to use <property> tag in xml driven cfgs for setter injection	b) we need to use <constructor-arg> tag in xml driven cfgs for constructor injection	b) Not possible in xml driven cfgs
c) needs to use setter method with @Autowired to perform setter injection in annotation driven cfgs	c) Needs to use param constructor with @Autowired to perform constructor injection in annotation driven cfgs	c)needs to place @Autowired on the top of the property/filed in the target spring bean class
d) Injects bit lately compare to constructor Injection/ Filed Injection (order is 3)	d) Injects value/object first compare to setter Injection / Filed Injection (order is 1)	d) Injects bit lately compare to constructor Injection /bit early compare to setter Injection (order is 2)
e) this injection makes the IOC container to create spring bean class obj using 0-param constructor	e) this injection makes the IOC container to create spring bean class obj using param constructor	e) this injection makes the IOC container to create spring bean class obj using 0-param constructor
f) makes the IOC container to create target spring bean first then creates and Injects the dependent spring bean	f) makes the IOC container to create dependent spring bean first then creates the the target spring bean using that dependent spring bean	f) makes the IOC container to create target spring bean first then creates and Injects the dependent spring bean
g) supports Circular/Cyclic Dependency Injection (A is dependent to B and B is dependent to A)	g) does not Support	g) supports
(h) if the spring bean properties are optional participate in dependency injection then prefer setter injection	(h) if all the properties of spring bean are mandatory to participate in the injections then prefere working with Constructor Injection	(h) if the properties of the spring bean are optional to participate in the injections then prefer working with Field Injection

What is the difference between BeanFactory and Applicationcontext container?

BeanFactory container

a) It is the object of a class that implements BeanFactory(1) directly or indirectly eg: XmlBeanFactory

b) It is subset of ApplicationContext Container

(only for interview)

ApplicationContext container

(a) It is the object of a java class that implements ApplicationContext(1) directly or indirectly

eg:: FileSystemXmlApplicationContext,

AnnotationConfigApplicationContext

(b) It is super set of BeanFactory Container

(c) supports

(Supports eager instantiation for singleton scope spring bean)

c) Does not support pre-instantiation of singleton scope spring beans (only Lazy instatiation)

(d) There is no support for properties file and place holders ${<key>} directly (some additional configurations are required) (e) No support for Internationalization(I18n)

(d) There is direct support for properties file and place hodlers ${<key>}

(e) supports the Internationalization (118n)

BeanFactory (1)

✓ HierarchicalBeanFactory (1) ApplicationContext

>

✓ ListableBeanFactory

> ApplicationContext (1)

note:: BeanFactory Container means it is the obj of a class that implements BeanFactory(1) directly or indirectly

ApplicationContext container means it is the object of a class that implements ApplicationContext(I) directly or indirectly

Application context Container= Bean Factory Container++

(f) Does not support the event handling

(f) supports

(g) supports

(g) Does not give direct support for code annotation driven cfgs and 100% driven cfgs

(h) allows only xml file as the spring bean cfg file

(j)

in

(I) Resource obj is required to pass spring bean cfg file (xml file)

(h) Allows xml file as the spring bean cfg file and

(i)

@Configuration class (java class annotated with @Configuration) as the configuuration class

note:: Using BeanFactory Container we can not keep track of when IOC container is started, stoped/closed.. (No support for event handling) Using ApplicationContext Container such event handling is possible

(required in 100% code driven cfgs and spring boot programming) Required; in spring boot Programming

(xml file) No need of Resource obj)

note:: For Bean factory container, we can give inputs only through xml file.. where as in ApplicationContext container we can give inputs to IOC container using only xml file or using xml +annotations or using only java code + annotations

note:: Both Spring/Spring boot frameworks use same IOC containers.. In spring programming we can use both BeanFactory and Application Containers ..where as in Spring boot programming we can use only ApplicationContext container

note: Compare to ApplicationContext IOC container the BeanFactory IOC container is light weight IOC container (Overall both the IOC containers are Light weight containers. In which BeanFactory container is more light weight)

(i) can not think about using Spring boot Programming

Suitable Memory and Resource SEnşitive applications

(if less memory is there for app to execute)

(k) This container can not be stopped or closed

(m) Ligight weigth container

(j) Mostly used spring container in most of the spring applications and in spring boot apps

(k) This container can be stopped or closed or refreshed or ....

(1) we can pass spring bean cfg file directly while creating this IOC container

(n) Bit Heavy weight container compare to BeanFactory

Where should i use BeanFactory IOC container and where should i use ApplicationContext container? IOT apps Ans) => if spring is used in Memory Senstive apps like mobile Apps, embedded System apps and etc.. where 1 or 2 few extra kilo bytes also matters then prefer using BeanFactory IOC contianer ..

=>if spring is used in web applications, distributed Apps, enterprize apps and etc.. where Memory utlization does not matter then prefer using ApplicationContext container

note:: spring is not popular in mobile, Embedded systme apps.. sspring, spring boot frameworks

known developing enterprise apps .. so prefer using ApplicationContext container as u r first choice container

web applications distributed App

note:: In latest Spring, Spring Boot Projects always prefere working with ApplicationContext Container, if ur not using that container u must proper reason to convience ur self,,

have

Q) what is the difference between IOC and Dependency Injection?

Ans) IOC is the software specification that gives set of rules and guidelines to manage the dependency among the target and dependent classes.. Dependency Injection and Dependency Lookup are the two implemetation models of IOC specification

IOC is like blue print/theory / plan/ for managing the dependency

where as Dependency lookup(DL) and Dependency Injection are the two implementation models/praticals of

IOC specification

Q) For IOC container of 100% Code driven cfgs, if we can not give certain inputs or instructions through Java code and annotations then how to pass such special insturctions to IOC container?

Ans) we can give these special instructions to IOC container through xml file (spring bean cfg file --generally applicationContext.xml) by linking the file with @Configuration class using @ImportResource annotation

applicationContext.xml (com/nt/cfgs)

...

....

(spring bean cfg file)

AppConfig.java (Configuration class)

To create the IOC container

=====

AnnotationConfigApplicationContext ctx=

new AnnotationConfigApplicationContext(AppConfig.class);

@Configuration

@ImportResource("com/nt/cfgs/applicationContext.xml")

public class AppConfig{

}

with

conclusion :: First prefer giving all inputs and instructions to IOC container using java code + annotations.. if ‣ they are not sufficient then think about going for spring bean cfg file by linking @Configuration class using @ImportResource annotation..

(xml file)

100% code driven cfgs approach = java config approach

note:: To link spring bean configuration file (xml file) with @Configuration class use @ImportResource annotation To link one @Configuration class with another @Configuration class use @Import annotation

What is difference among the setter injection, constructor and field Injection?

Setter Injection

Constructor Injection

a)uses parameterized constructor for the

a) uses setter method of target class to assign given dependent value/object to target spring bean same class obj

b) we need to use <property> tag in xml driven cfgs for setter injection

c) needs to use setter method with @Autowired

to perform setter injection in annotation driven cfgs

d) Injects bit lately compare to constructor Injection/ Filed Injection (order is 3)

e) this injection makes the IOC container to create spring bean class obj using O-param

constructor

f) makes the IOC container to create target spring bean first then creates and Injects the dependent spring bean

b) we need to use <constructor-arg> tag in xml driven cfgs for constructor injection

c) Needs to use param constructor with @Autowired to perform constructor injection

in annotation driven cfgs

d) Injects value/object first compare to setter Injection / Filed Injection (order is 1)

e) this injection makes the IOC container

to create spring bean class obj using param constructor

f) makes the IOC container to create dependent spring bean first then creates the the target spring bean using that dependent spring bean

g) supports Circular/Cyclic Dependency Injection g) does not Support

(A is dependent to B

and B is dependent to A)

Field Injection

(uses the reflection api)

a) IOC container directly access the filed/property of target spring bean for injection activity

b) Not possible in xml driven cfgs

c) needs to place @Autowired on the top of the property/filed in the target spring bean class

d) Injects bit lately compare to constructor Injection /bit early compare to setter Injection (order is 2)

e) this injection makes the IOC container to create spring bean class obj using O-param constructor

f) makes the IOC container to create target spring bean first then creates and Injects the dependent spring bean

g) supports

(h) if the spring bean properties are optional participate

in dependency injection then prefer setter injection

(h) if all the properties of spring bean are mandatory to participate in the injections then prefere working with Constructor Injection

(h) if the properties of the spring bean are optional to participate in the injections then prefer working with Field Injection

Q) How to make our spring app as 100% Loosely coupled flexible application to change the dependent spring

bean with another dependent spring bean with out touching source code of the target spring bean class?

Ans) This can be done in two ways

a) Using @Qualifier(-) + properties file + spring bean cfg file (xml file) + alias tag and etc...

(or)

b) Using Spring profiles

(Best)

(yet to be discussed)