=> Spring supports both Dependency lookup and Dependency Injection
=> Mostly used Dependency management technique in spring is
Dependency Injection (DI)

Dependency Management in spring

1. Dependency Lookup
=> call lookup( ) on IOC container to perform this DL operation

2.Dependency Injection
=> This can be implemented in multiple ways
a) using Field Injection (best)
b) using setter Injection
c) using Constructor Injection (fastest)
d) using arbitrary method injection
e) Interface Injection/Aware Injection
f) Lookup Method Injection
g) Method Replacer /Method Injection

(a–d: Most regularly used injection spring)

Creating IOC containers
==================

=> Spring f/w offers two IOC containers
a) BeanFactory IOC container
b) ApplicationContext IOC container

Both these Containers are Light weight containers compare to the heavy weight Servlet Container and Jsp Container

note: To work Servlet container and Jsp Container we need to install the heavy weight web SErver like Tomcat or Application server (weblogic) s/w

How Can u say Spring containers /IOC containers are Light weight Containers?

ans1) By creating objects for spring api supplied pre defined classes we can keep IOC containers ready in any kind of java apps

ans2) To create and run IOC containers , we do not need heavy weight web server or Application server s/w installations

ans3) We can create and run IOC containers just like that by having JRE/JVM as the Base software

BeanFactory IOC container creation
=========================

=> To create container , take a pre defined class of spring api implementing BeanFactory(I) and create object for it, then object as the BeanFactory IOC container

org.springframework.beans.factory.xml
Class XmlBeanFactory

java.lang.Object (c)
 └ org.springframework.core.SimpleAliasRegistry (c)
  └ org.springframework.beans.factory.support.DefaultSingletonBeanRegistry (c)
   └ org.springframework.beans.factory.support.DefaultListableBeanFactory (c)
    └ org.springframework.beans.factory.xml.XmlBeanFactory (c)

BeanFactory(I)
ListableBeanFactory(I)
ConfigurableListableBeanFactory(I)
implements

=> Regularly used class to create IOC container is XmlBeanFactory class

eg:: FileSystemResource res=new FileSystemResource("..../applicationContext.xml");
Resource obj holding the name and location of spring bean cfg file — path — Spring Bean configuration file

XmlBeanFactory factory =new XmlBeanFactory(res);

This object represents BeanFactory IOC container by taking given xml file as the spring bean cfg file

To give inputs and instructions to IOC container
|--> spring bean cfgs
|--> Dependency Injection cfgs
|--> life cycle cfgs and etc..

ApplicationContext Container =BeanFactory Container ++

BeanFactory
ApplicationContext

What is basic reason for BeanFactory becoming outdated IOC container ?

Ans) The recent /latest trend of providing inputs and instructions to IOC container is using annotation and java code driven configurations (inputs and instructions) But the BeanFactory IOC container supports only XML driven cfgs which is fading out So BeanFactory IOC container is also fading out

ApplicationContext IOC container creation
===============================

=> This Container takes xml file, annotations and java code driven cfgs
=> This Container is Industry standard IOC container
=> To create this IOC container take a class implementing ApplicationContext(I) directly or indirectly

Package org.springframework.context

Interface ApplicationContext

note: ApplicationContext(I) is sub interface of BeanFactory(I)

=> Spring supports both Dependency Lookup and Dependency Injection => Mostly used Dependency management technique in spring is

Dependency Injection (DI)

Dependency Management in spring

1. Dependency Lookup

=>call lookup(-) on IOC container to perform this DL operation

Creating IOC containers

2.Dependency Injection

=> This can be implemented in multiple a) using Field Injection (best) b) using setter Injection

=====

c) using Constructor Injection (fastest) d) using arbitrary method Injection

e) Interface Injection/Aware Injection f) Lookup Method Injection

g) Method Replacer /Method Injection

=> Spring f/w offers two IOC containers

a) BeanFactory IOC container

b) ApplicationContext IOC container

ways

Most regularly used Injection spring

Both these Containers are

Light weight containers compare to the heavy weight

SErvlet Container and Jsp Container

note:: To work Servlet container and Jsp Container we need to install

the heavy weight web Server (like Tomcat) or Application server (weblogic) s/ws

How Can u say Spring containers /IOC containers are Light weight Containers?

ans1) By creating objects for spring api supplied pre-defined classes we can keep IOC containers ready in any kind of java apps

1

ans2) To create and run IOC containers we do not need heavy weight web server or Application server s/w installations

ans3) WE can create and run IOC containers just like that by having JRE/JVM as the Base software

BeanFactory IOC container creation

=======

====

=> To create container take a pre-defined class of spring api implementing BeanFactory(I) and create object for it. then object as the BeanFactory IOC container

org.springframework.beans.factory.xml

Class XmlBeanFactory

java.lang.Object (c)

Lorg.springframework.beans.factory.support.AbstractBeanFactory. (c)

BeanFactory(1)

extends ListableBeanFactory(1)

extends

ConfigurableListableBeanFactory(I)

implements

Lorg.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory. (c)
Lorg.springframework.beans.factory.support.DefaultListableBeanFactory.
Lorg.springframework.beans.factory.xml.XmlBeanFactory (c)

=> Regularly used class to create IOC container is XmlBeanFactory class

eg1:: FileSystemResource_res=new FileSystemResource("..../applicationContext.xml");

Resource obj holding the

path

Spring Bean configuration file

name and location of

spring bean cfg file

XmlBeanFactory factory=new XmlBeanFactory(res);

This object represents

BeanFactory IOC container by taking

given xml file as the spring bean cfg file

Application Context Container =BeanFactory Container + +

BeanFactory ApplicationContext

To give inputs and instructions to IOC container |-->spring bean cfgs

|--> Dependency Injection cfgs |--> life cycle cfgs and etc..

What is basic reason for BeanFactory becoming outdated IOC container?

Ans) The recent /latest trend of providing inputs and instructions to IOC container is using annotation and java code driven configurations (inputs and instructions) But the BeanFactory IOC container supports only XML driven cfgs which is fading out So BeanFactory IOC container is also fading out

ApplicationContext IOC container creation

=======

=========

=> This Container takes xml files, annotations and java code driven cfgs

=> This Container is industry standard IOC container

=> To create this IOC container take a class implementing ApplicationContext(I) directly or indirectly

Package org.springframework.context

Interface ApplicationContext

note: ApplicationContext(I) is sub interface of BeanFactory(1)

Enter chat message here

All Superinterfaces:

ApplicationEventPublisher, BeanFactory, EnvironmentCapable, HierarchicalBeanFactory, ListableBeanFactory, MessageSource, ResourceLoader, ResourcePatternResolver

All Known Subinterfaces:

ConfigurableApplicationContext, ConfigurableWebApplicationContext, WebApplicationContext

All

Implementing Classes:

AbstractApplicationContext, Abstract RefreshableApplicationContext, Abstract RefreshableConfigApplicationContext, Abstract RefreshableWebApplicationContext, AbstractXm1ApplicationContext, AnnotationConfigApplicationContext, AnnotationConfigWebApplicationContext, ClassPathXmlApplicationContext, FileSystemXmlApplicationContext, GenericApplicationContext, GenericGroovyApplicationContext, GenericWebApplicationContext, GenericXmlApplicationContext, GroovyWebApplicationContext, StaticApplicationContext, StaticWebApplicationContext, XmlWebApplicationContext

BeanFactory(1)

extends

ListableBeanFactory(I)

extends

ApplicationContext(I)

Imp classes to create ApplicationContext Container

a) FileSystemXmlApplicationContext

b) ClassPathXmlApplicationContext

c) XmlWebApplicationContext

d) AnnotationConfigApplicationContext

e) AnnotationConfigWebApplicationContext

a) FileSystemXmlApplicationContext

=>Create the AC (ApplicationContext) container by taking given spring bean cfg file

from the Specified path of file system

FileSystemXmlApplicationContext_ctx=

-This obj represents AC IOC container

new FileSystemXmlApplicationContext("...../applicationContext.xml");

=>This is suitable in standalone apps (but not the best)

b) ClassPathXmlApplicationContext

path of the spring bean cfg file

=>creates the AC container by taking the given spring bean cfg file from the jar files and directories added to CLASSPATH env. variable

ClassPathXmlApplicationContext ctx=

new ClassPathXmlApplicationContext("applicationContext.xml");

=> This is best class to create "AC" IOC container taking xml driven cfgs in standalone apps (Compare to

FileSystemXmlApplicationContext) => It is suitable in stand alone apps (It is best for standalone apps)

c) XmlWebApplicationContext

=> creates "AC" IOC container in web applications by taking fixed notation name as the spring bean cfg file that is <servlet logical name>-servlet.xml as the spring bean cfg file from the WEB-INF folder

eg: XmlWebApplicationContext ctx=new XmlWebApplicationContext(); note:: if the servlet logical name is "xyz" then spring bean cfg file name is "xyz-servlet.xml".

=> In xml driven cfgs, this is only the option to create "AC" IOC container in web applications

d) AnnotationConfig ApplicationContext

=>create the "AC" IOC container in standalone apps by taking given java class as the Configuration class (java class with @Configuration annotation)

eg1:: AnnotationConfigApplicationContext ctx=

new AnnotationConfigApplicationContext(AppConfig.class);

@Configuration class

=> @Configuration class, we can give inputs and instructions to IOC container

note: In Annotation, java code cfgs, this is the only class to create "AC" IOC container in standalone apps.

=> Suitable only in standalone apps

e) AnnotationConfigWebApplicationContext

=>creates "AC" IOC container in web applications by taking given java class

as the Configuration class

=> Suitable in web applications, Distributed Apps

eg1: AnnotationConfig WebApplicationContext ctx=

AC container

new AnnotationConfig WebApplicationContext(AppConfig.class);