**Q:what is Spring Bean.**

A:Every classyou are writing in spring application can be called as Spring Bean.

**Q:what is Dependency.**

A:Any property available in a Bean class which needs to be initialized is called Dependency.

**Q:what is injection.**

A:inetialization of Bean class property is also called as injection.

**Q:what is Dependency injection(DI).**

A:the process of inetializing of Bean Dependencies is called Dependency injection.

**Q:what are types of Dependency Injection supported by Spring.**

A:Spring Dependency Injection uses 3 ways to inject the dependencies

1-Setter Injection

2-Constructor Injection

3-Field Injection(using Annotations)

**Q:what is Setter Injection.**

A:the process of inetializing Bean Dependencies with setter method is called

Setter injection.

**Q:How can I implement Setter Injection**.

A:Using <property> tag

**Q:what is Constructor Injection.**

A:the process of inetializing Bean Dependencies with Constructor is called as

Constructor injection.

**Q:How can I implement Constructor Injection.**

A:Using <constructor-arg> tag

**Q:when should i use <property> tag and <constructor-arg> tag ?**

A:refer notes

**Q:when should I use 'value' attribute and 'ref' attribute.**

A:use 'value' attribute for specifying values for simple type of variable like

1- Primitive variables

2- Wrapper Variables

3- String Variables

use 'ref' attribute for specifying bean references for references type variable

**Q:what are the GOF Patterns used in Spring IOC Container.**

A:-Single-ton pattern and Factory pattern

**Q:what is Spring Container.**

A:ApplicationContext object is called as Spring Container.

**Q:what is Spring Configuration file.**

A:Spring Configuration file is an XML file which contains various bean definitions.

**Q:Can I configure two beans with same Id.**

<beans>

<bean id="hello" class="Hello1"/>

<bean id="hello" class="Hello2"/>

</beans>

A:No,Bean Id must be unique.

there are multiple occurence of ID value 'hello'.

**Q:Can I configure two beans with same Bean Class.**

A:yes

<beans>

<bean id="hello1" class="Hello"/>

<bean id="hello2" class="Hello"/>

</beans>

Hello h1 = (Hello)ctx.getBean("hello1");

Hello h2 = (Hello)ctx.getBean("hello2");

Here h1!=h2

**Q:how many bean scope are there.**

A:there are 5 scope

1- singleton

2- prototype

3- request

4- session

5- global-session

**Q:what is the default bean scope.**

A:singleton is the default scope in the ApplicationContext container.

**Q:what is the diff b/w singleton and prototype**.

**Q:how many loading types are there.**

**Q:what is the defaul bean loading tpe.**

A:lazy loading lazy-init=true

**Q:what is wiring ? how many ways are available**

**Q:what is Auto wiring ? How to implement Auto Wiring?**

----------------------------------------------------------------------------------------------

Consider the Hello class with 4 constructors as follows in the same order

Hello()

Hello(A)

Hello(B)

Hello(A,B)

<bean id="h" class="Hello" autowire="constructor"/>

**Q:which Constructor will be called with the following availability of the beans ?**

A type- 0 beans

B type- 0 beans

A:Hello()

**Q:which Constructor will be called with the following availability of the beans ?**

**A type- 1 beans**

**B type- 0 beans**

A:Hello(A)

**Q:which Constructor will be called with the following availability of the beans ?**

**A type- 0 beans**

**B type- 1 beans**

A:Hello(B)

**Q:which Constructor will be called with the following availability of the beans ?**

**A type- 1 beans**

**B type- 1 beans**

A:Hello(A,B)

**Q:which Constructor will be called with the following availability of the beans ?**

**A type- 2 beans**

**B type- 1 beans**

A:Hello(A,B) if A bean is resolved with byName

Hello(B) if A bean is not resolved with byName

**Q:which Constructor will be called with the following availability of the beans ?**

**A type- 1 beans**

**B type- 2 beans**

A:Hello(A,B) if B bean is resolved with byName

Hello(A) if B bean is not resolved with byName

**Q:which Constructor will be called with the following availability of the beans ?**

**A type- 2 beans**

**B type- 2 beans**

A:Hello(A,B) if A and B bean is resolved with byName

Hello(A) if A bean is resolved with byName

Hello(B) if B bean is resolved with byName

Hello() if A and B bean is not resolved with byName

**Q:Consider the Hello class with 3 constructors as follows in the same order**

**Hello()**

**Hello(A)**

**Hello(B)**

**which Constructor will be called with the following availability of the beans ?**

**A type- 1 bean**

**B type- 1 bean**

A:Hello(A)-Present first in class.

**Q:Consider the Hello class with 3 constructors as follows in the same order**

**Hello()**

**Hello(A)**

**Hello(B)**

**which Constructor will be called with the following availability of the beans ?**

**A type- 1 bean**

**B type- 1 bean**

A:Hello(B)-Present first in class.

**Q:Consider the Hello class with 3 constructors as follows in the same order**

**Hello()**

**Hello(A,B)**

**Hello(B,A)**

**which Constructor will be called with the following availability of the beans ?**

**A type- 1 bean**

**B type- 1 bean**

A:Hello(A,B)-Present first in class.

**Q:Consider the Hello class with 3 constructors as follows in the same order**

**Hello()**

**Hello(B,A)**

**Hello(A,B)**

**which Constructor will be called with the following availability of the beans ?**

**A type- 1 bean**

**B type- 1 bean**

A:Hello(B,A)-Present first in class.

class A{}

class B extends A{}

class Hello

{

A aobj;

......

.....

}

<bean id="ao" class="...A"/>

<bean id="bo" class="...B"/>

<bean id="hello" class="Hello" autowire="byType"/>

Ans: it throws UnsatisfiedDependencyException

Two Beans found with the A type-one is ao and second is bo(because B inherit A)

**Q:How can i implement XML Based Autowiring?**

A:Using autwire attribute of bean tag.

**Q:what will happen when I specify the “byName “ autowire process?**

**Q:what will happen when I specify the byType autowire process?**

**Q:what will happen when I specify the constructor autowire process?**

**Q:what will happen with the following code**.

public class Hai

{

int a;

String str;

public Hai(int a)

{

this.a=a;

}

public Hai(String str)

{

this.str=str;

}

}

<beans>

<bean id="hai" class="Hai">

<constructor-arg value="1234"/>

</bean>

</beans>

A:creates the Hai class instance by using Matching Constructor[Hai(String)]

**Q:what will happen with the following code.**

public class Hai

{

int a;

String str;

public Hai(int a)

{

this.a=a;

}

}

<beans>

<bean id="hai" class="Hai">

<constructor-arg value="1234"/>

</bean>

</beans>

A:creates the Hai class instance by using Matching Constructor[Hai(int)]

**Q:what will happen with the following code.**

public class Hai

{

int a;

long str;

public Hai(long a)

{

this.a=a;

System.out.println("long");

}

public Hai(int a)

{

this.a=a;

System.out.println("int");

}

}

<beans>

<bean id="hai" class="Hai">

<constructor-arg value="1234"/>

</bean>

</beans>

A:creates the Hai class instance by using Constructor which is found first in

the class [Hai(long)]

**Q:what will happen with the following code.**

public class Hai

{

int a;

long str;

public Hai(int a)

{

this.a=a;

System.out.println("int");

}

public Hai(long a)

{

this.a=a;

System.out.println("long");

}

}

<beans>

<bean id="hai" class="Hai">

<constructor-arg value="1234"/>

</bean>

</beans>

A:creates the Hai class instance by using Constructor which is found first in

the class [Hai(int)]

**Q:what will happen with the following code.**

public class Hai

{

int a;

long str;

public Hai(int a)

{

this.a=a;

System.out.println("int");

}

public Hai(long a)

{

this.a=a;

System.out.println("long");

}

}

<beans>

<bean id="hai" class="Hai">

<constructor-arg value="Subhag"/>

</bean>

</beans>

A:Unsatisfied dependency expressed through construtor argument with index 0 of type

[int]:Could not convert constructor argument value of type.

so excepton is java.lang.NumberFormatException:for input string:"jlc"

**Q:can i use cyclic Dependency Injection?**

A:Depending on the case.

when you are injecting the resources with Setter Injection then circular reference

is allowed(Refer Q\*->next question)

when you are injecting the resources with C Injection then circular reference

may be allowed or may not(Refer Q\*\*-second next question)

**Q\*:what will happen with the following code?**

class Hai

{

Hello hello;

pubic void setHello(Hello hello)

{

this.hello=hello;

}

public void show()

{

System.out.println(hello);

}

}

public class Hello

{

Hai hai;

public void setHai(Hai hai)

{

this.hai=hai;

}

public void show()

{

System.out.println(hai);

}

}

<beans>

<bean id="hai" class="Hai" autowire="byType"/>

<bean id="hello" class="Hello" autowire="byType"/>

</beans>

A:it will run successfully.

injects Hai into Hello first and then Injects Hello into Hai.

**Q\*\*:what will happen with the following code?**

public class Hai

{

static

{

System.out.println("Hai-S.B");

}

Hello hello;

public Hai()

{

System.out.println("Hai-D.c");

}

pubic Hai(Hello hello)

{

this.hello=hello;

System.out.println("Hai-1.arg");

}

public void show()

{

System.out.println(hello);

}

}

public class Hello

{

static

{

System.out.println("Hello-S.B");

}

Hai hai;

public Hello()

{

System.out.println("Hello-D.c");

}

pubic Hello(Hai hai)

{

this.hello=hello;

System.out.println("Hello-1.arg");

}

public void show()

{

System.out.println(hai);

}

}

<beans>

<bean id="hai" class="Hai" autowire="constructor"/>

<bean id="hello" class="Hello" autowire="constructor"/>

</beans>

A:it will run successfully.

only Injects Hello into Hai

**Q:what is P-NameSpace**

A:

Q:How can i implement Annotation based Auto wiring.

A:Using @Autowired and @Resource

**Q:what is the diff b/w @Autowired and @Resource**

A:

**Q:can i use Annotations directly in Spring Application**.

A:No,

you must enable annotation processing by writing the <context:annotation-config/> tag

in spring configuration document.

.

**Q:How to decide to the type of Dependency Injection.**

A:

Setter Injection or Field Injection ->Make the beans loosely coupled

Constructor Injection ->Make the beans tightly coupled

Ex:

--

class A{}

class Hai

{

A ao;

...

Hai(){}

public void setAo(A ao)

{

this.ao=ao;

}

}

Without A bean instance,

Container will create Hai bean instance

Container can not inject A bean into Hai.

i.e Without A bean,Hai bean is instantiated.So these two beans are loosely coupled.

Ex:

--

class Hello

{

A ao;

Hello(A ao)

{

this.ao=ao;

}

}

Without A bean instance,

Container can not create Hello bean instance

i.e Without A bean,Hello bean will not be instantiated.So these two beans are

tightly coupled.

**Q:can i use P-Namespace with constructor.**

A:No

**Q:what is the bean name.**

A: bean id is also called as bean name.

you can also specify the name for the bean in the bean definition

Ex:

<bean id="ao" class="A"/>

ctx.getBean("ao");

<bean name="bo" class="B"/>

ctx.getBean("bo");

<bean id="co" name="cobj" class="C"/>

ctx.getBean("co");

ctx.getBean("cobj");

**Q:How can I implement Annotation Based Autowiring**

A:using the following 3 Annotations

@Autowired

@Resource

@Inject

**Q: what is the difference among @Autowired, @Resource,@Inject**

**Q:Annotations are replacement for XML. but still we are writing Beans in XML body.**

**can i use only annotations without xml.**

A:yes,you can configure your Beans with java based Configuration without XML configuration.

use the following Annotations for java Based Configuration

@Configuration

@Bean

refer-Lab\_32

**Q:what is the use of @Import Annotation.**

**Q:what is the use of @ImportResource Annotation.**

**Q:what is the use of @Scope Annotation.**

**Q:what is the use of @Lazy Annotation.**

**Q:what are the Initialization callbacks provided with BeanFactory container**

**Q:what are the Initialization callbacks provided with ApplicationContext container**

**Q:you can Inject the required resources with DI.what is the use of Intialization callbacks.**

A:mainly to check whether resources are initialized by the Spring Container or not.

class Hello

{

@Autowired

Hai hai;

@PostConstruct

public void init()

{

if(hai==null)

{

hai=....;

}

else

{

throw Some Exception;

}

}

void show()

{

hai.m1();

}

}

**Q:what are the Disposable callbacks provided with BeanFactory container**

**Q:what are the Disposable callbacks provided with ApplicationContext container**

**Q:How can i get the reference of BeanFactory container into the bean.**

A:there are two ways to initialize your bean with BeanFactory reference.

1-By implementing BeanFactoryAware interface and overriding

setBeanFactory() method

2-By using @Autowired(use this)

**Q:How can i get the reference of ApplicationContext container into the bean.**

**A:there are two ways to initialize your bean with BeanFactory reference.**

1-By implementing ApplicationContextAware interface and overriding

setApplicationContext() method

2-By using @Autowired(use this)

**Q:what is the use of BeanPostProcessor**

A:BeanPostProcessor allows to extend the functionality of ApplicationContextContainer

**Consider the following case**

**you want to make subhag() method as lifecycle method for every bean**

-------------------------------------------------------------------

subhag() method has to be called when bean class

--> is implementing subhag() interface.

--> is containing the method which is marked with @subhag

**1-Bean class Implemeting subhag interface**

-----------------------------------------

class Hello imp subhag,InitializingBean

{

public void subhag()

{

// do something

}

}

public class MyBeanPostProcessor implements BeanPostProcessor

{

public Object postProcessBeforeInitialization(Object object,String bname)

{

// 1- get Object of the Bean

// 2- get the List of interfaces implemented by the Bean class using Reflection

// 3- Check the subhag interface is in the list

// 4- if subhag interface is in the List

then call subhag() method

}

}

**2-Method in the Bean class marked with @subhag**

--------------------------------------------------------------

class Hello imp subhag,InitializingBean

{

@jlc

public void subhag()

{

// do something

}

}

public class MyBeanPostProcessor implements BeanPostProcessor

{

public Object postProcessBeforeInitialization(Object object,String bname)

{

// 1- get Object of the Bean

// 2- get the List of methods available in Bean class using Reflection

// 3- Check whether any method is marked with @jlc

// 4- if @jlc is found for any method

then call that method

}

}

**Q:what are the Spring Containers supported**

**Q:what are the diff b/w BeanFactory Container and ApplicationContext Container**

**Q:We are creating Spring container object in main method.after scope over still container objects exits.How to justify(Every object should be inside scope)**

A:if container is shutdown then it has to invoke disposable callbacks.

**Q:is there any relationship b/w BeanFactory and ApplicationContext container**

A:ApplicationContext is the decorator of BeanFactory

ApplicationContext = BeanFactory+ Extra code

ApplicationContext creates the instance of BeanFactory internally.

**Q:why we are using AbstractApplicationContext.**

A:only to invoke the method registerShutdownHook()

**Q:why we are using AnnotationConfigurationApplicationContext**.

A:to use java Based Configuration

**Q:can I write multiple Spring Configuration files.**

A:Spring Application can have multiple spring Configuration files

Assume that you have written subhag.xml,subhag1.xml,subhag2.xml,

1- in subhag.xml

<import resource="subhag1.xml"/>

<import resource="subhag2.xml"/>

ctx = new ClassPathXmlApplicationContext("subhag.xml");

2-String str[]={"subhag.xml","subhag1.xml","subhag2.xml"};

ctx = new ClassPathXmlApplicationContext(str);

**Q:can i write multiple java based Configuration classes.**

A:yes you can.(Refer Lab\_34)

**Q:whether Spring container instance creation is Singleton,multi-threaded.**

A:Depending on you

you can create one or more instance of Spring Container and should be multi-threaded

**Q:How can i decide to use type of Spring container**

A:use always ApplicationContext only

**Q:there are two classes A(lazy) and B(Agresive) and are in inheritence relationship.when Spring container is loading sub classes and what will happen to Super class.**

A:One instance of B will be created.loads both classes and allocates the memory for both the class variables and invokes the both the class constructor.

**Q:is there any way to extend the BeanFactory container functionality.**

A:NO

**Q:Can i clone Application Context object.**

A:NO

**Q:can i inject the Bean with One scope into another Bean with different scope**

A:yes

**Q:can i mark two methods with @PostConstruct annotations**

A:yes,you can

**Q:can i mark two methods with @PreDestroy annotations**

A:yes,you can

**Q:can i inject the bean defined one xml into another bean defined in another xml.**

A:yes.

**Q:what are Inner Beans.**

A:when you specify the Bean defination inside another defination then it is called Inner Bean.

**Q:while you are configuring the beans in xml,we are hard-coding the values?**

**can i make it dynamic.**

A:yes,you can use property files.(using Externalizing Bean properties)

**Q:what is the use of ApplicationEvents.**

**Q:how can I implement ApplicationEvents.**

**Q:how can I Publish ApplicationEvents.**

**Q:how can I implement ApplicationListeners.**

**Q:how can I Register ApplicationListeners.**

**Q:can i register multiple ApplicationListeners with the Spring Container.**

**Q:what is the use of Property Editors.**

**Q:how can i develop Custom Property Editors.**

**Q:how can i Registor Custom Property Editors.**

**Q:how can i Registor Message Bundels.**

**Q:how can i access the properties from Message Bundels.**

**Q:why i need to Inject ApplicationContext into the Bean**

A:you can use the following methods of ApplicationContext.

getMessage()

publishEvent()

etc.

**Q:what is the diff b/w ClassPathResource and FileSystemResource.**

**Q:what is the diff b/w ClassPathXmlApplicationContext and FileSystemXmlApplicationContext**

**Q:what is the sub classes of BeanFactory interface.**

**Q:what is the sub classes of ApplicationContext interface.**

**Q:what is the use of @Required Annotation**

**Q:what is the diff b/w @Autowired(required=true) and @Autowired(required=false)**

**Q:what is the use of @Qualifier Annotation.where can i use this**

**Q:what are the GOF Patterns discussed in Spring IOC.**

A: 1- Singleton Pattern

2- Factory Pattern

3- Decorator Pattern