we can develop the spring apps in 3 approaches

a) using xml driven cfgs (all inputs and instructions to the IOC container will be given using xml file)
b) using xml + annotation driven cfgs (all inputs and instructions to the IOC container will be given using annotations +xml file)
c) using 100% Java code driven cfgs (all inputs and instructions to the IOC container will be given using annotations + Java code)
(best)

In xml driven cfgs, we get support for two basic dependency injections
a) setter injection (we make IOC container calling setter method of target spring bean class to assign dependent spring bean class obj to target spring bean class object)
(best)
=> For this we need to use <property> under <bean> tag in spring bean cfg file

b) constructor injection (we make IOC container using parameterized constructor of target spring bean class to create target spring bean class obj and also to assign to dependent spring bean class obj to it)
(later)
=> For this we need to use <constructor-arg> under <bean> tag in spring bean cfg file

In Annotation driven cfgs (java code driven cfgs) we can perform all the 4 basic injections using different annotations

a) setter injection (place @Autowired on the top of the setter method for setter injection)

b) constructor injection (place @Autowired on the top of the parameterized constructor for constructor injection)
(latest)

c) field injection (place @Autowired on the top of fields (HAS-A properties) in target spring bean class)
(best)
field is like member variable or property in a class

d) Arbitrary method injection (place @Autowired on the top of Arbitrary method of target spring bean class)
=> setter method is having fixed method name like setName(), setCourse(), setAddrs() ...
=> arbitrary method is similar setter method but we can take any name for it like putAge(), assignAge(), injectData() , fieldData() and etc ..

=> Dependency injection is called as wiring (linking target and dependent spring beans)
=> @Autowired makes the IOC container to detect the dependent spring bean dynamically either by name (bean id) or by type (bean class name) and to assign to target spring bean class obj. So it is named as @Autowired

In target Spring bean (Sample code)

@Autowired //Field Injection
private DTDC courier; //HAS-A property or filed

w.r.t industry we deal with 3 types of Projects
20% a) New Projects (scratch level development)
70% b) Maintenance Projects (Already released projects for which we are giving support)
10% c) Migration Projects (Redeveloping the project using latest technologies or latest versions of the existing technologies)

=> Spring maintenance projects are there in xml driven cfgs or xml+annotation driven cfgs
=> Spring New or Migration projects are there in 100% Java code driven cfgs

Spring App Designing for setter injection in xml driven cfgs

```
Target class                                    Dependent class
                (user-defined class)
com.nt.beans.WishMessageGenerator               java.time.LocalTime     (WishMessageGenerator uses
                (HAS-A property)                (pre-defined class)       uses LocalTime obj given current hour
   |--> private LocalTime time;                                          of the day to generate the wish message
                                                                          like "Good Morning", "Good Evening",
                //b.method                                                "Good Afternoon", "Good Night")
   |--> public String showWishMessage(String user){
            // get current hour of the day (time)
              ...
            //generate the wish message
              ...
        }
```

WishMessageGenerator.java

```
package com.nt.beans;

                    package import statements

public class WishMessageGenerator{
   //HAS-A property
   private LocalTime time;


   //setter method for setter injection
   public String showWishMessage(String user){
     //get current hour of the day

     int hour=time.getHour(); // 24 hours format

     //generate the wish message
     if(hour<12)
         return "Good Morning"+user;
   else if(hour<16)
         return "Good Afternoon"+user;
   else if(hour<20)
         return "Good Evening "+user;
   else
         return "Good Night"+user;
```

=> we can develop the spring apps in 3 approaches

a) using xml driven cfgs (all inputs and instructions to the IOC container will be given using xml file) b) using xml+ annotation driven cfgs (all inputs and instructions to the IOC container will be given using

annotations +xml file)

c) using 100% java code driven cfgs (all inputs and instructions to the IOC container will be given (Best) using annotations + java code)

In xml driven cfgs, we get support for two basic dependency injections

a) setter Injection (we make IOC container calling setter method of target spring bean class to assign dependent spring bean class obj to target spring bean class object)

(Best)

=> For this we need to use <property> under <bean> tag in spring bean cfg file

b) constructor Injection (we make IOC container using parameterized constructor of target spring Bean class to create target spring bean class obj and also

(fatest)

to assign to dependent spring bean class obj to it)

=> For this we need to use <constructor-arg> under <bean> tag in spring bean cfg file

In Annotation driven cfgs /java code driven cfgs we can perform all the 4 basic injections using different annotations

a) setter Injection (place @Autowired on the top of the setter method for setter Injection)

b) constructor Injection (place @Autowired on the top of the parameterized constructor (fatest) for constructor Injection)

Field Injection (place @Autowired on the top of Fileds (HAS-A properties) (Best)

in target spring bean class)

Field = Member variable = property in a class

d) Arbitrary method Injection (place @Autowired on the top of Arbitrary method

of target spring bean class)

=>setter method is having fixed setXxx(-) name like setName(-), setCourier(-), setEngine(-),... =>arbitrary method is same setter method but we can take any name for it like

putXxx(-), assignYyy(-), injectData(-), linkData(-) and etc...

=>Dependency Injection is called as wiring (Linking target and dependent spring beans) =>@Autowired makes the IOC container to detect the dependent spring bean dynamically either by name (bean id) or by type (bean class name) and to assign to target spring bean class obj. So it is named as @Autowired

In taraget Spring bean (Sample code)

=====

@Autowired //Filed Injection

private DTDC courier; //HAS-A property or filed

s/w industry deals with 3 types of Projects

10% a) New Project (scratch level development)

80% b) Maintenance Projects (Already released projects for which we are giving support) 10%) Migration

Projects (Redeveloping the project using latest technologies or latest versions of the existing technologies)

=> Spring maintenance projects are there in xml driven cfgs or xml +annotation driven cfgs => Spring New or Migration projects are there in 100% java code driven cfgs

Spring App Designing for setter Injection in xml driven cfgs

=====:

Target class

==========

(user-defined class)

com.nt.sbeans.Wish MessageGenerator

(HAS-A property)

|---> private LocalTime time;

//b.method

======== ======

Dependent class

=============

java.time.LocalTime (pre-defined class)

-->public String showWishMessage(String user){

// get current hour of the day (time)

...

//generate the wish message

(WishMessageGenerator uses

uses LocalTime obj given current hour of the day to generate the wish message like "Good Morning", "Good Evening", "Good AfterNoon","Good night")

WishMessageGenerator.java

package com.nt.sbeans;

package import statements

public class WishMessageGenerator{

//HAS-A property

private LocalTime time;

//setter method for setter Injection

public Strng showWishMessage(String user){ //get current hour of the day

int hour-time.getHour(); // 24 hours format

//generate the wish message

if(hour<12)

return "Good Morning:"+user;

else if (hour<16)

else if(hour<20)

return "Good Afternoon:"+user;

return "Good Evening :"+user;

else

}

return "Good Night:"+user;

applicationContext.xml (spring bean cfg file)

====

<beans

bean id

<!--target spring bean class cfg -->

Spring bean class name

<bean id="wmg" class="com.nt.sbeans.Wish MessageGenerator">

(has-a property)

<property name="time" ref="ltime"/> Injects LocalTime class obj (ltime) to the "time" property of WishMessageGenerator class obj property name dependent spring bean id for injecting dependent spring bean class obj

</bean>

<!-- dependent spring bean -->

<bean id="ltime" class="java.time.LocalTime"/> </beans> bean id spring bean class name

=> Every Spring bean is identified with its bean id (object name of spring bean)

=> The bean ids will be used in multiple angels

a) To Inject one spring bean class obj with another spring bean class obj (ref="ltime")

b) To get specific spring bean class obj ref from the IOC container (ctx.getBean("wmg"))

and etc..

=> if we place any <bean> tag with out <property> or <constructor-arg> tag inside it (as the sub tags) then

we can say that spring bean is dependent spring bean or independent spring bean

=> The <bean> tag under which <property> or <constructor-arg> tag is placed is called target spring bean class

=> The IOC container uses given bean id as the object name while creating the object for the spring bean class

SetterInjectionTest.java (main class)

========

package com.nt.main;

======

public class SetterInjectionTest{

public static void main(String args[]){

//create IOC container

ClassPathXmlApplicationContext ctx= new

ClassPathXmlApplicationContext("...../applicationContext.xml");

```
//get Spring bean class obj ref
Object obj=ctx.getBean("wmg");
//type casting
WishMessageGenerator wmg=(WishMessageGenerator)obj;
//invoke the b.method
String msg=wmg.showWishMessage("raja");
System.out.println(msg);
// close the IOC Container
ctx.close(); // In the process of closing IOC container
destroys all the spring bean class objs
} //main
} //class
```