w Xml driven cfgs , we can perform the following basic injections on spring beans

a) Setter injection
b) constructor injection (Mine)

=> In annotation driven , 100% Code driven cfgs we can perform the following 4 basic injections

a) setter injection b) constructor injection c) field injection d) Arbitrary Method Injection
(Mine) (Mine)

Q) Where should I use setter injection and where Should I use constructor injection?

Ans) If there is a mandatory rule that all the properties of spring bean must participate in the injection activity then we can prefer using constructor injection bcoz by taking 1 parameterized constructor in spring bean class we can complete the injection.

If there is possibility of involving on choice no.of spring bean properties in the injection activity then we can prefer using Setter injection bcoz by taking "n" setter methods maximum for "n" properties we can make our choice no.of properties in the Dependency Injection process.

Example scenario
================

=> If spring bean class is having 10 spring bean properties and u want make all properties participating in the injection then we need to take only one 10 param constructor to complete that process.

=> if spring bean class is having 10 spring bean properties and u want to choose no.of properties participating in the injection through constructor injection then we need to take 1023 (1+10+45+120+210+252+210+120+45+10+1= 1,024,000) no.of constructors . So always prefer constructor injection here...but prefer working with setter injection here we can take max 10 setter methods for setter injection to involve our choice no.of properties in the injection process.

What is difference b/w setter injection and constructor injection?

setter injection	constructor injection
a)IOC container uses the setter method to inject dependent value or dependent spring bean class obj to target spring bean class obj	a)IOC container uses the parameterized constructor to create target spring bean class obj and also to inject dependent value or dependent spring bean class obj to target spring bean class obj
b) Here setter method is used only for injection activity	b) Here parameterized constructor is used to create target spring bean class obj and also to inject dependent values/objects to it
c) Injects the dependent values/objects only after creating the target spring bean class obj	c) Injects the dependent values/objects while creating target spring bean class obj itself
d) Relation in the injection process compare to constructor injection	d) Relation in the injection process compare to setter injection
e) use <property> under <bean> tag for setter injection in xml driven cfgs	e) use <constructor-arg> under <bean> tag for constructor injection in annotation driven cfgs and 100% Code driven cfgs
f) To involve our choice no.of spring bean properties in the injection we just "n" setter methods for "n" spring bean properties	f) To involve our choice no.of spring bean properties in the constructor injection we need "n" no.of parameterized constructors for constructor injection
g) To involve all bean properties of spring bean immediately in the injection process we need to make "n" setter methods for "n" properties	g) To involve all bean properties of a spring bean immediately in the injection process we need to make "1" parameterized constructor for "n" properties
h) prefer this setter injection to involve our choice no.of spring bean properties in the injection process	h) prefer this constructor injection to involve all spring bean properties in the injection process
i) In annotation driven cfgs, 100% code driven cfgs we need to place @Autowired on top of setter methods for setter injection	i) In annotation driven cfgs, 100% code driven cfgs, we need to place @Autowired on top of parameterized constructor for constructor injection
j) In setter injection, the IOC container first creates target spring bean class and then creates dependent spring bean class obj then assigns dependent spring bean to target spring bean by calling setter method Target.setX(a)(dependent);	j) In constructor injection, the IOC container first creates Dependent spring bean class obj and uses that object as the arg value of parameterized constructor while creating target spring bean class obj. Target class obj=new Target class)(dependent);
k) Supports cyclic or circular dependency injection	k) Does not Support cyclic or circular dependency injection
l) if the properties of spring bean class are configured for setter injection or for no injection then IOC container creates spring bean class obj using 0 param constructor	l) if none or all spring bean properties are configured for constructor injection then the IOC container creates spring bean class obj using parameterized constructor

Q) What is the difference b/w creating the IOC container using BeanFactory and ApplicationContext

=> Xml driven cfgs, we can perform the following basic injections on spring beans

a) Setter Injection

b) constructor Injection (fatest)

=> In annotation driven, 100% Code driven cfgs we can perform the following 4 basic Injections a) setter injection b) constructor injection c) Field Injection d) Arbitrary Method Injection (best)

(fatest)

Q) Where should i use setter Injection and where Should i use constructor Injection?

Ans) if there is a mandatory rule that all the properties of spring bean must participate in the injection activity then we can prefer using constructor Injection becoz by taking 1 parameterized constructor in spring bean class we can complete this injection.

if there is possibility of involving ur choice no.of spring bean properties in the

injection activity then we can prefer using Setter Injection becoz by taking "n" setter methods maximum for "n" properties we can make our choice no.of properties in the Dependency Injection process.

Example scenario

==========

=> if spring bean class is having 10 spring bean properties and u want make all properties participating in the injection then we need to take only one 10 param constructor

to complete that process.

=> if spring bean class is having 10 spring bean properties and u want ur choice no.of properties participating in the injection through constructor injection then we need to take 10! (10 factorial= 3,628,800) no.of constructor.. So do not prefer constructor injection here.. but prefer working with setter injection becoz we can take max 10 setter methods for setter injection to involve our choice no.of properties in the injection process.

What is difference b/w setter Injection and constructor Injection?

setter Injection

(a)IOC container uses the setter method to inject dependent value or dependent spring bean class obj to target spring bean class obj

(b) HEre setter method is used only for injection activity

(c) Injects the dependent values/objects only after creating the target spring bean class obj

(d) Bit slow in the injection process compare to constructor Injection

(e) use <property> under <bean> tag for setter Injection in xml driven cfgs

need

(f) To involve our choice no.of spring bean properties in th¶¶ction we just "n" setter methods for "n" spring bean properties

constructor Injection

(a)IOC container uses the parameterized constructor to

create target spring bean class obj and also to inject dependent value or dependent spring bean class obj to target spring bean class obj

b) Here parameterized constructor is used to create target spring bean class obj and also to inject dependent values/objects to it

(c) Injects the dependent values/objects while creating target spring bean class obj itself

(d) Bit faster in the injection process compare to setter Injection

(e) use <constructor-arg> under <bean> tag for constructor Injection in annotation driven cfgs and 100% Code driven cfgs

(f) To involve our choice no.of spring bean properties in the constructor injection we need "n!" no.of parameterized constructors for constructor injection

(g) To involve all bean properties of spring bean mandatorly in the injection process we need to max "n" setter methods for "n" proeprties

(h) prefer this setter injection to involve our choice no.of spring bean properties in the injection process

(i) In annotation driven cfgs, 100% code driven cfgs we need to place @Autowired on top of setter methods for setter injection

(j) In setter Injection, the IOC container first creates target spring bean class and next creates dependent spring bean class obj then assigns dependent spring bean to target spring bean by calling setter method

target.setXxx(dependent);

(k) Supports cyclic or

circular dependency Injection

(g) To involve all bean properties of a spring bean mandatorly in the injection process we need

to max of "1" parameterized constructor for "n" proeprties

(h) prefer this constructor injection to involve all spring bean properties in the injection process

(i) In annotation driven cfgs, 100% code driven cfgs, we need to place @Autowired on top of parameterized constructor for constructor injection

(j) In constructor Injection, the IOC container first creates Dependent spring bean class obj and uses that object as the arg value of parameterized constructor while creating target spring bean class obj.

(l) if the properties of spring bean class are configured for setter injection or for no injection

Target class obj=new Target class(dependent);

(k) Does not Support cyclic or circular dependency Injection

(l) if one or more or all spring bean properties are configured for constructor injection then the IOC container creates spring bean class obj using

then IOC container creates spring bean class obj parameterized constructor using O-param constructor

Q) What is the difference b/w creating the IOC container using FileSystemXmlApplicationContext and ClassPathXmlApplicationContext for spring app?

Ans) FileSystemXmlApplicationContext create the IOC container by locating the given spring bean cfg file from the specified path of the file system ..This path can be relative path(good) or absolute path

eg1::FileSystemXmlApplicationContext ctx=

new FileSystemXmlApplicationContext("src/com/nt/cfgs/applicationContext.xml"); relative path of spring bean cfg file

eg2:: FileSystemXmlApplicationContext ctx=

w.r.t Project folder

new FileSystemXmlApplicationContext("E:\\Worskpaces\\Spring\\NTSPBMS515\\IOCProj04-SeasonFinderApp-

DependencyInjection-xml+annotation driven cfgs\\src\\com\\nt\\cfgs\\applicationContext.xml");

Absolute path of spring bean cfg file

=>ClassPathXmlApplicationContext creates the IOC container by locating the spring bean cfg file from folders and jar files added to the CLASSPATH/BUILDPATH

=> "src" folder of every Java Project is kept in CLASSPATH by default

eg1:: ClassPathXmlApplicationContext ctx=

Best approach of

new ClassPathXmlApplicationContext("com/nt/cfgs/applicationContext.xml");

creating the IOC

container in xml driven cfgs

eg2:: => make com/nt/cfgs folder as the source folder (adding this folder to classpath) right click on com.nt.cfgs package --->buildpath-->use as the source folder

=> In main class

Assignment

=========

=> Inject LocalDate class obj to WeekDayFinder class using constructor injection of annotation driven cfgs and find out whether current week day is week end or not?