Today Agenda

Spring Core
JDBC,Servlet,JSP(update part application)
Hibernate,Servlet,JSP(CRUD operation)

Constructor Injection

Here IOC container uses parameterized constructor to create target class object.
In this process it assigns/injectes dependant object to the Target class Object.
In Setter injection first target class object is created,next Dependant object will be created.
In case of Constructor injection,First Dependant class object should be created,next target
class object will be created and this dependant object will be passed as the argument to the
constructor.

syntax:

```
<bean id ='' class=''>
      <constructor-arg name='' ref=''/>
</bean>
```

if we place<constructor-arg> tag for 'n' time under bean tag, then IOC container uses
"n-param" constructor to create the Spring bean class Object.

Note: <ref> attribute to cfg bean id is based on Spring bean class object injection to target bean
class properites.
<value> to cfg is to inject "simple values" to Spring bean class properties.

Note:
What happens if we enable both setter injection and construcor injection to bean property?
Tell me which id will be injected as final values?
Ans> Since setter method is called after the constructor execution, we say setter injection overrides the value injected by the constructor injection.
Values/Object injected by the Setter injection will become final values.

Note:
Bean id should be unique w.r.t IOC container
We can configure 2 spring beans having same class names but we should take different bean id.

Usage of index/type/name attribute in the <constructor-arg>

If there are mulitple params in constructor and if they have same datatype,then to resolve the
parameter binding from the container we need to go either index/name(recomended)

```
private Integer eno;
private String ename;
private float esalary;
private String eaddress;
public Employee(Integer eno, String ename, float esalary, String eaddress) {
            this.eno = eno;
            this.ename = ename;
            this.esalary = esalary;
```

```
            this.eaddress = eaddress;
}

<constructor-arg value="MI" type='java.lang.String'/>
<constructor-arg value="sachin"  type='java.lang.String'/>
<constructor-arg value="3500.05F" type='float'/>
<constructor-arg value="10"  type='java.lang.Integer'/>

Injection from container
Employee [eno=10, ename=MI, esalary=3500.05, eaddress=sachin]

resolving the mismatch through index
-----------------------------------
private Integer eno;
private String ename;
private float esalary;
private String eaddress;
public Employee(Integer eno, String ename, float esalary, String eaddress) {
            this.eno = eno;
            this.ename = ename;
            this.esalary = esalary;
            this.eaddress = eaddress;
}

<constructor-arg value="MI" index='3'/>
<constructor-arg value="sachin"  index='1'/>
<constructor-arg value="3500.05F" index='2'/>
<constructor-arg value="10"  index='0'/>

Employee [eno=10, ename=sachin, esalary=3500.05, eaddress=MI]

Resolving through name attribute
-------------------------------
private Integer eno;
private String ename;
private float esalary;
private String eaddress;
public Employee(Integer eno, String ename, float esalary, String eaddress) {
            this.eno = eno;
            this.ename = ename;
            this.esalary = esalary;
            this.eaddress = eaddress;
}

<constructor-arg value="MI" name='eaddress'/>
<constructor-arg value="sachin"  name='ename'/>
<constructor-arg value="3500.05F" name='esalary'/>
<constructor-arg value="10"  name='eno'/>

Employee [eno=10, ename=sachin, esalary=3500.05, eaddress=MI]

Container work flow
-------------------
1. BeanFactory container  ========> XmlBeanFactory(Depercated from Spring3.1V)
2. BeanFactory container  ========> DefaultListableBeanFactory


Limitations of XmlBeanFactory
=============================
```

1. Need of Resource object to hold the name and location of Spring bean configuration file.
2. XmlBeanFactory uses XmlParser to read the bean defnition and to process the xml file which is
   not good in terms of performance.
3. Doesn't allow to take multiple xml files at a time as spring bean configuration file.

CircularDependancy Injection/Cyclic Dependancy Injection
==========
=> It is all about making 2 classes dependent on each other.
=> It is not at all industry practise.
=> Setter injection supports CyclicDependancy/but Constructor injection doesn't support
   CyclicDependancy.
=> One side Setter and another side Constructor injection would also support "CyclicDependancy".

Difference between Setter injection and Constructor injection
==========

Setter Injection
==========
1. use setter method to inject the dependant values/objects to target class object.
2. <property name='' value=''/> and <property name='' ref=''/>
3. supports cylic dependancy injection
4. bit slow becoz injection happens after creating the Target class object.
5. First Target object and later Dependant object will be created.
6. It is best suited when we want to involve our choice no of properties in dependancy
      injection.
7. If we configure spring bean in setter injection style,then container will create the bean
      using "zero-arg" constructor.

Constructor Injection
==========
1. use constructor to inject the dependant values/objects to target class object.
2. <constructor-arg name='' value=''/> and <constructor-arg name='' ref=''/>
3. Doesn't supports cylic dependancy injection
4. It is Fast becoz injection happens while instantiating the dependant class object.
5. First Dependant object and later Target object will be created.
6. It is not best suited when we want to involve our choice no of properties in dependancy
      injection,for this we need n! overloaded constructor.
7. If we configure spring bean in constructor injection style,then container will create the bean
      using "n-param" constructor.

One Project using SpringCore[DI strategy by configuring the container]

VO -> Value Object(it holds the data entered by the user, data would always be in
                                String format only)
BO -> Buisness Object(it holds the actual data which needs to persisted for future usage)
DTO-> Data transfer Object(it holds the data in the required data type for processing)