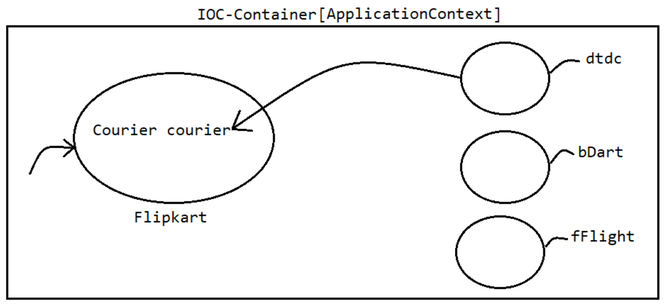
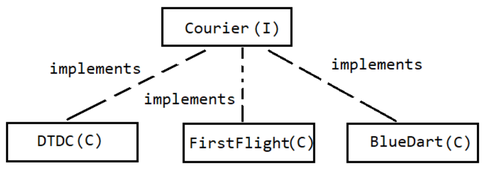
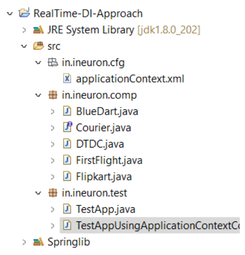
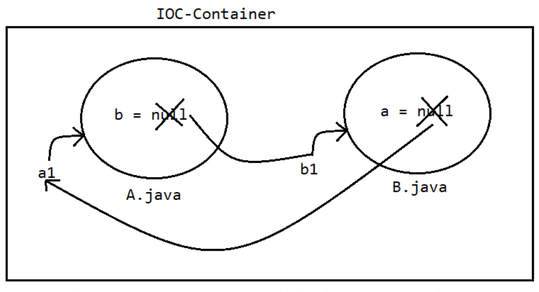
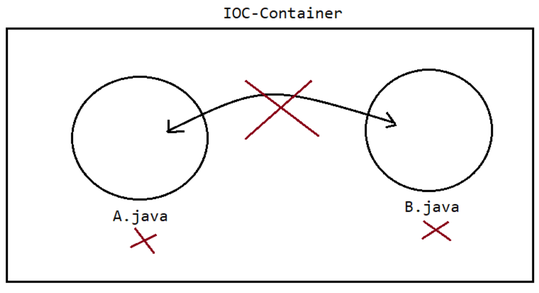




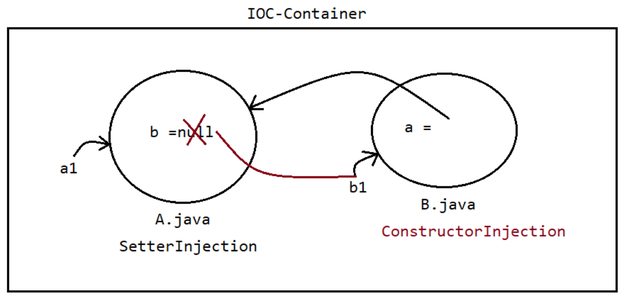


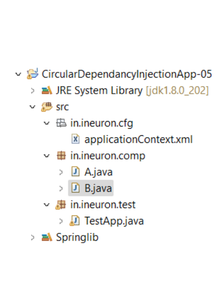


Cyclic Dependancy is possible in Setter injection

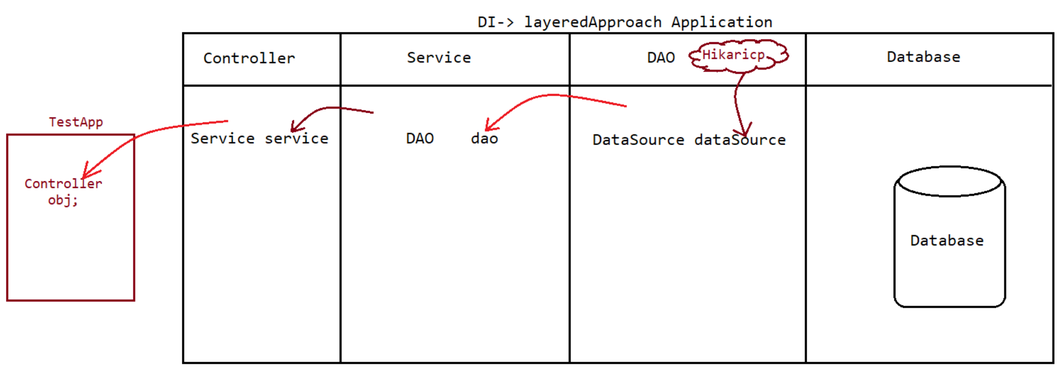


Cyclic Dependancy is not possible in "Constructor Injection"





Cyclic Dependancy is possible if one is constructor and the other one is Setter Injection



Constructor Injection

a. Hikaricp ---> DataSource(DAO)
b. DAO ---> Service
c. Service ---> Controller
d. Test ---> Get the Controller object

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 type/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]

Note:What happens if we specify name,type,index attribute in one <constructor-arg>
tag at a time?
      => If all these are pointing to same param of consturctor.. then give value
will be injected to the param otherwise it would result in
            "org.springframework.beans.factory.UnsatisfiedDependencyException".

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


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.


Advantages of DefaultListableBeanFactory
========================================
1. It directly recoginizes the beans,without passing this work to another class.
2. Allows to take multiple xml files at a time becoz loadDefnition(String... args)
argument is of var-args.
3. No need of taking Resource object to hold the name and location of spring bean
configuration file.
4. The performance towards reading and processing xml file is good.

One Project using SpringCore[DI technique to inject many dependant
objects[implementation classes of Courier] to Target class]
-------------------------------------------------------------------------------
--------
1. Courier(I)
        |=> DTDC,BlueDart,FirstFlight
2. Flipkart(C)
        |=> Courier courier
Perform dependancy injection through setter approach and also demonstrate the usage
of "Scope of bean".


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.

```
public class Student
{
	private Integer sid;
	private String sname;
	private Integer sage;
	private String cityName;
	private String countryName;
	private String stateName;

	// 1 Zeroparam constructor
	// 6 setXXXX methods
	// toString()
}

<bean id="student" class ="in.ineuron.comp.Student">
	<property name ="" value=""/>
	<property name ="" value=""/>
	<property name ="" value=""/>
</bean>

public class Student
{
	private Integer sid;
	private String sname;
	private Integer sage;
	private String cityName;
	private String countryName;
	private String stateName;

	// n! -> no of consturctors with all combinations
	// 1 6-param constructor
```

```
	// 2 3-param constructor
	// 3 4-param constructor

	// 6 setXXXX methods
	// toString()
}
<bean id="student" class ="in.ineuron.comp.Student">
	<constructor-arg name="cityName" value=""/>
	<constructor-arg name="sname" value=""/>
	<constructor-arg name="sid" value=""/>
	<constructor-arg name="sage" value=""/>
</bean>
```