Note:
While developing layered application,Instead of configuring all DAO classes,ServiceClass,Controller Class
as a springbean in single configuration file(.cfg.xml),It is recomended to take multiple xml files and link
them to single xml file.

```
      <beans ......>
            <import resource="location of .xml file"/>
      </beans>
```

applicationContext.xml
======================

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:context="http://www.springframework.org/schema/context"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
        https://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        https://www.springframework.org/schema/context/spring-context.xsd">


      <import resource="controller-beans.xml" />
      <import resource="service-beans.xml" />
      <import resource="persistence-beans.xml" />

</beans>
```

Collection Injection

=> It is all about injecting values to array,collection type bean properties through Dependancy injection.

Property type	tag/attribute
simple/primitive	<value>
object	<ref>
array	<array>/<list>
List	<list>
Set	<set>
Map	<map>
Properties	<props>

applicationContext.xml
======================

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
        https://www.springframework.org/schema/beans/spring-beans.xsd">


      <bean id="sysDate" class="java.util.Date" />
      <bean id="dob" class="java.util.Date">
            <property name="year" value="93" />
            <property name="month" value="0" />
            <property name="date" value="03" />
      </bean>
```

```
<!--Array Injection-->
<bean id="marks" class="in.ineuron.comp.MarksInfo">
	<property name="marksInfo">
		<array>
			<value>100</value>
			<value>97</value>
			<value>98</value>
			<value>35</value>
		</array>
	</property>
</bean>

<!--List Injection-->
<bean id="clg" class="in.ineuron.comp.College">
	<property name="studNames">
		<list>
			<value>sachin</value>
			<value>saurav</value>
			<value>dravid</value>
			<value>laxman</value>
		</list>
	</property>
	<property name="dateList">
		<list>
			<ref bean="sysDate" />
			<ref bean="dob" />
		</list>
	</property>
</bean>

<!--Set Injection-->
<bean id="contact" class="in.ineuron.comp.ContactsInfo">
	<property name="phoneNumbers">
		<set value-type="java.lang.Long">
			<value>9999999999</value>
			<value>8888888888</value>
			<value>7777777777</value>
			<value>7777777777</value>
		</set>
	</property>
	<property name="dates">
		<set>
			<ref bean="sysDate" />
			<ref bean="dob" />
		</set>
	</property>
</bean>

<!--Map Injection-->
<bean id="uInfo" class="in.ineuron.comp.UniversityInfo">
	<property name="facultyDetails">
		<map key-type="java.lang.Integer" value-type="java.lang.String">
			<entry>
				<key>
					<value>10</value>
				</key>
				<value>sachin</value>
			</entry>
```

```
                        <entry key="7" value="dhoni" />
                        <entry key="18" value="kohli" />
                  </map>
            </property>
            <property name="datesInfo">
                  <map key-type="java.lang.String" value-type="java.util.Date">
                        <entry>
                              <key>
                                    <value>toDay</value>
                              </key>
                              <ref bean="sysDate" />
                        </entry>
                        <entry key="dob" value-ref="dob"/>
                  </map>
            </property>

            <!-- Property Injection -->
            <property name="iplInfo">
                  <props>
                        <prop key="RCB">FAF</prop>
                        <prop key="MI">Rohith</prop>
                        <prop key="CSK">Dhoni</prop>
                        <prop key="LSG">Rahul</prop>
                        <prop key="GT">Pandya</prop>
                        <prop key="KKR">Iyer</prop>
                  </props>
            </property>
      </bean>
</beans>

Collection injection in realtime
================================
DriverManagerDataSource(C)
      |-> driverClassName
      |-> url
      |-> connectionProperties(java.util.Properties object with fixed key called
"user" and "password")
                  |-> user
                  |-> password

applicationContext.xml
======================
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:context="http://www.springframework.org/schema/context"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
        https://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        https://www.springframework.org/schema/context/spring-context.xsd">

      <!-- Setter Injection to get DataSource Object -->
      <bean id="mysqlDAO"
class="org.springframework.jdbc.datasource.DriverManagerDataSource"
scope="singleton">
            <property name="driverClassName" value="${jdbc.driverClassName}" />
            <property name="url" value="${jdbc.url}" />

            <!--Properties injection to inject user-name and password of database
-->
```

```
            <property name="connectionProperties">
                  <props>
                        <prop key="user">${jdbc.user}</prop>
                        <prop key="password">${jdbc.password}</prop>
                  </props>
            </property>
      </bean>
      <context:property-placeholder
location="in/ineuron/commons/application.properties" />
</beans>
```

NullInjection

In constructor injection, all params must participate in injection process otherwise it would result in "Exception".
If constructor param type is object/reference type and we are not ready with value then we can go for null injection.
This is very handy(useful) when we are working with predeifned classes as a spring bean, that is a spring bean will have limited no of
Overloaded constructors and no setter injection support is available.
syntax: `<constructor-arg name=''><null/></constructor-arg>`

applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
        https://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="pinfo" class="in.ineuron.comp.PersonInfo">
            <constructor-arg name="pid" value="10"/>
            <constructor-arg name="pname" value="sachin"/>
            <constructor-arg name="paddress" value="MI"/>
            <constructor-arg name="dom"><null/></constructor-arg>
            <constructor-arg name="doj"><null/></constructor-arg>
            <constructor-arg name="dob" ref="dobObj"/>
</bean>

<bean id="dobObj" class="java.util.Date">
      <property name="year" value="93"/>
      <property name="month" value="0"/>
      <property name="date" value="03"/>
</bean>

</beans>
```

Person.java

```
public class PersonInfo {
      private Integer pid;
      private String pname;
      private String paddress;
      private Date dob;
      private Date dom;
      private Date doj;

      public PersonInfo(Integer pid, String pname, String paddress, Date dob, Date
```

```
dom, Date doj) {
            System.out.println("PersonInfo:: 6 param constructor...");
            this.pid = pid;
            this.pname = pname;
            this.paddress = paddress;
            this.dob = dob;
            this.dom = dom;
            this.doj = doj;
      }
      toString()
}

output
======
PersonInfo [pid=10, pname=sachin, paddress=MI, dob=Sun Jan 03 21:47:30 IST 1993,
dom=null, doj=null]

Can we do null injection while working with setter injection?
      Ans. yes, but not required becoz while working with setter injection there is
no mandatory to inject value/object to bean property.
             we can involve our choice properties in setter injection,more over the
reference type/object type bean properties by default holds
            null values.
                        <property name="dob"><null/></property>


Bean inheritance
--------------------
<bean id='baseCar' class="in.ineuron.bean.Car" abstract="true">
            <constructor-arg name="engineCC" value='1500' />
            <constructor-arg name="model" value='swift' />
            <constructor-arg name="company" value='suziki' />
            <constructor-arg name="fuelType" value='diesel' />
            <constructor-arg name="type" value='hatchback' />
</bean>

<bean id='car1' class='in.ineuron.bean.Car' parent="baseCar">
            <constructor-arg name="owner" value='sachin' />
            <constructor-arg name="regNo" value='KA4567' />
            <constructor-arg name="color" value='red' />
            <constructor-arg name="engineNo" value='12345' />
</bean>

<bean id='car2' class='in.ineuron.bean.Car' parent="baseCar">
            <constructor-arg name="owner" value='dhoni' />
            <constructor-arg name="regNo" value='JH5647' />
            <constructor-arg name="color" value='white' />
            <constructor-arg name="engineNo" value='56789' />
</bean>

Car car3 = factory.getBean("baseCar", Car.class);
System.out.println(car3);

Output
      org.springframework.beans.factory.BeanIsAbstractException(class is abstract
so)


Important points of inheritance in bean configuration file
```

---
1. This is not a class level inheritance, it is spring bean cfg file level bean properties inheritance across multiple spring bean cfgs.
2. <bean abstrat='true'> will neve make the class as abstract,but it makes spring bean cfg as abstract.
3. One spring bean can inherit and resuse the spring bean properties only from one spring bean..

Bean inheritance in realtime

```
      DataSource(I)===> javax.sql.*
            |
    DriverManagerDataSource(C)===> org.spring.**
            |
     HikariDatasource(C)  ======> com.zaxxer.**
```

persistence.xml

```
<bean id="mysqlDataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
            <property name="driverClassName" value='com.mysql.cj.jdbc.Driver' />
            <property name="username" value='root' />
            <property name="password" value='root123' />
</bean>

<bean id='hikariDataSource' class='com.zaxxer.hikari.HikariDataSource'
parent="mysqlDataSource">
      <property name="jdbcUrl" value='jdbc:mysql:///enterprisejavabatch' />
      <property name="minimumIdle" value='10' /><!-- min pool size -->
      <property name="maximumPoolSize" value='20' />
      <property name="connectionTimeout" value='2000' />
</bean>

<!-- DAO Configuration -->
<bean id='mysqlDaoImpl' class='in.ineuron.dao.CustomerMySQLDAOImp'>
            <constructor-arg name='dataSource' ref='hikariDataSource' />
</bean>
```

Design a layered approach application to perform CRUD opearation on database called Employee using SpringCore

- Concepts to be used are
  - a. Properties file injection
  - b. use BeanInheritane and work with hikaricp
  - c. use Collection injection
  - d. use VO,BO,DTO approach
- a. Employee => eid,ename,eage,eaddress
  - operation performed are
    - a. insert
    - b. update(by doing validation)
    - c. delete
    - d. read based on id
    - e. getAllRecords