**\*\*\*\*\*\*\*VaishaliSpring3.0AnnoXMLProject\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*Address.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.cg.bean;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.factory.annotation.Qualifier;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.stereotype.Component;

@Component("offAdd")

public class Address

{ @Value("MS")

private String state;

@Value("India")

private String country;

@Value("67877")

private String zipcode;

public Address() { super(); }

@Autowired

@Qualifier("city1")

private City city;

@Override

public String toString() {

return "Address [state=" + state + ", country=" + country

+ ", zipcode=" + zipcode + ", myCity=" + city + "]";

}

public Address(String state, String country, String zipcode, City city) {

super();

this.state = state;

this.country = country;

this.zipcode = zipcode;

this.city = city;

}

public String getState() { return state; }

public void setState(String state) { this.state = state; }

public String getCountry() { return country; }

public void setCountry(String country) { this.country = country;}

public String getZipcode() { return zipcode; }

public void setZipcode(String zipcode) { this.zipcode = zipcode; }

public City getCity() { return city; }

public void setCity(City city) { this.city = city; }

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Igreet.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.cg.bean;

public interface IGreet

{

public String greetMe();

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*NewYearGreetWishImpl.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.cg.bean;

import org.springframework.beans.BeansException;

import org.springframework.beans.factory.BeanFactory;

import org.springframework.beans.factory.BeanFactoryAware;

import org.springframework.beans.factory.BeanNameAware;

import org.springframework.beans.factory.DisposableBean;

import org.springframework.beans.factory.InitializingBean;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.stereotype.Component;

public class NewYearWishGreetImpl implements IGreet,BeanNameAware ,BeanFactoryAware ,DisposableBean

,InitializingBean

{

@Value("Vaishali") String name;

@Value("1999") String year;

public String getYear() { return year; }

public void setYear(String year) {

this.year = year;

System.out.println("\*\*\*\*\*\*\*\*setYear() called and year initialized\*\*\*"); }

public String getName() { return name; }

public void setName(String name) {

this.name = name;

System.out.println("\*\*\*\*\*\*\*\*setName() called and name initialized\*\*\*"); }

public NewYearWishGreetImpl()

{ System.out.println("\*\*\*\*NewYearWishGreetImpl Bean Instantiated\*\*\* "); }

@Override

public String greetMe(){

return name +" Happy New Year "+year;

}

@Override

public void setBeanFactory(BeanFactory bf) throws BeansException {

System.out.println("setBeanFactory callled \*\*\*\*\*");

}

@Override

public void setBeanName(String bn) {

System.out.println("setBeanName callled \*\*\*\*\*"+bn);

}

public void initGreetAfterCreation()

{

System.out.println(" \*\*After bean instantiated "+

"some initialization can be performed to get it to a usable state "+

" that time initGreetAfterCreation is called\*\*\*\*\*\*\*\*\*\*\*\*");

}

public void destroyGreetBeforeDeletion()

{

System.out.println(" \*\*When the bean is removed from"+

" the container, some cleanup may be required"+

" that time destroyGreetBeforeDeletion is called\*\*\*\*\*\*\*\*\*\*\*\*");

}

@Override

public void destroy() throws Exception {

System.out.println("bean is destroied from container\*\*\*\*\*\*");

}

@Override

public void afterPropertiesSet() throws Exception {

System.out.println(" This Is called After All PropertiesSet ");

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*BirthDayWishImpl.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.cg.bean;

public class BirdayWishImpl

implements IGreet

{

@Override

public String greetMe()

{

return " Happy Birth day ";

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*NewYearWishPostProcessor.java\*\*\*\*\*\*\*\***

package com.cg.bean;

import org.springframework.beans.BeansException;

import org.springframework.beans.factory.config.BeanPostProcessor;

public class NewYearWishPostProcessor implements BeanPostProcessor {

@Override

public Object postProcessAfterInitialization(Object bean,

String beanName)

throws BeansException {

System.out.println("AfterInitialization : " + beanName);

Class cls=bean.getClass();

String type=cls.getName();

System.out.println(type);

return bean; // you can return any other object as well

}

@Override

public Object postProcessBeforeInitialization(Object bean, String beanName)

throws BeansException {

System.out.println("BeforeInitialization : " + beanName);

return bean; // you can return any other object as well

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*City.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.cg.bean;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.stereotype.Component;

@Component("city1")

//@Component //autowire byType

public class City

{

public City() { super(); }

@Value("Pune")

private String name;

@Value("3000000")

private int population;

public City(String name, int population) {

super();

this.name = name;

this.population = population;

}

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public int getPopulation() { return population; }

public void setPopulation(int population) { this.population = population; }

@Override

public String toString() {

return "City [name=" + name + ", population=" + population + "]";

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CityList.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.cg.bean;

import java.util.ArrayList;

public class CityList

{

private ArrayList<City> cityList;

@Override

public String toString() { return "CityList [cityList=" + cityList + "]"; }

public ArrayList<City> getCityList() { return cityList; }

public void setCityList(ArrayList<City> cityList) { this.cityList = cityList; }

}

**\*\*\*\*\*\*\*\* CurrencyConvertorImpl.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.cg.bean;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.stereotype.Component;

@Component("currencyConObj1")

public class CurrencyConvertorImpl implements ICurrencyConvertor

{

private int exchangeRate;

private String countryName;

@Override

public String toString() {

return "CurrencyConvertorImpl [exchangeRate=" + exchangeRate

+ ", countryName=" + countryName + "]";

}

@Override

public int yenToRupees(int yen)

{

return (exchangeRate\*yen);

}

@Override

public double dollerToRupees(double dollers) { return (exchangeRate\*dollers); }

public CurrencyConvertorImpl(int exchangeRate)

{

super();

this.exchangeRate = exchangeRate;

}

public CurrencyConvertorImpl() { super(); }

@Autowired

public CurrencyConvertorImpl(@Value("68")int exchangeRate,

@Value("India")String countryName)

{

super();

this.exchangeRate = exchangeRate;

this.countryName = countryName;

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ICurrencyConvertor .java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.cg.bean;

public interface ICurrencyConvertor

{

public double dollerToRupees(double dollers);

public int yenToRupees(int yen);

}

**\*\*\*\*\*\*\*\*\*\*\*\* CustomDateEditorRegistrar \*\*\*\*\*\*\*\*\*\*\*\*\***

package com.cg.bean;

import java.text.SimpleDateFormat;

import java.util.Date;

import org.springframework.beans.PropertyEditorRegistrar;

import org.springframework.beans.PropertyEditorRegistry;

import org.springframework.beans.propertyeditors.CustomDateEditor;

public class CustomDateEditorRegistrar implements PropertyEditorRegistrar

{

@Override

public void registerCustomEditors(

PropertyEditorRegistry registry)

{

registry.registerCustomEditor(

Date.class, new CustomDateEditor(

new SimpleDateFormat("dd-MM-yyyy"),

false));

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\* Employee .java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.cg.bean;

import java.sql.Date;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.factory.annotation.Qualifier;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.stereotype.Component;

@Component("empObj")

public class **Employee**

{

@Value("#{T(java.lang.Math).random()\*100}")

private int eid;

@Value("#{per.firstName}")

private String enm;

@Value("#{per.getAge()}") private int eage;

@Autowired @Qualifier("offAdd") private Address empAddress;

@Value("getDoj()") Date doj;

@Value("03-02-2013") java.util.Date dob;

public Date getDoj() { return doj; }

public void setDoj(Date doj) { this.doj = doj; }

public java.util.Date getDob() { return dob; }

public void setDob(java.util.Date dob) { this.dob = dob; }

public Address getEmpAddress() { return empAddress; }

public void setEmpAddress(Address empAddress) { this.empAddress = empAddress; }

@Value("100000") private double esl;

@Override

public String toString() {

return "Employee [eid=" + eid + ", enm=" + enm + ", eage=" + eage

+ ", empAddress=" + empAddress + ", doj=" + doj + ", dob="

+ dob + ", esl=" + esl + "]";

}

public int getEid() { return eid; }

public void setEid(int eid) { this.eid = eid; }

public String getEnm() { return enm; }

public void setEnm(String enm) { this.enm = enm; }

public int getEage() { return eage; }

public void setEage(int eage) { this.eage = eage; }

public double getEsl() { return esl; }

public void setEsl(double esl) { this.esl = esl; }

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Person.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.cg.bean;

import java.util.ArrayList;

import javax.annotation.Resource;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.stereotype.Component;

@Component("per")

public class Person

{

@Value("Vaishali")

private String firstName;

@Value("Srivastava")

private String lastName;

@Value("80") private int age;

@Resource(name="getAddList")

private ArrayList addList;

public int getAge() { return age; }

public ArrayList getAddList() { return addList; }

public void setAddList(ArrayList addList) { this.addList = addList; }

@Override

public String toString() {

return "Person [firstName=" + firstName + ", lastName=" + lastName

+ ", age=" + age + ", addList=" + addList + "]";

}

public void setAge(int age) { this.age = age; }

public String getFirstName() { return firstName; }

public void setFirstName(String firstName) { this.firstName = firstName; }

public String getLastName() { return lastName; }

public void setLastName(String lastName) { this.lastName = lastName; }

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* SpringConfig .java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.cg.bean;

import java.util.ArrayList;

import java.util.List;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@Configuration

public class SpringConfig

{

@Bean

public List<Address> getAddList()

{

City hometown=new City("Nagpur",40000);

Address homeA=new Address("MS","INDIA","600087",hometown);

City edutown=new City("Ramtek",20000);

Address eduA=new Address("MS","INDIA","678",edutown);

List<Address> addList=new ArrayList<Address>();

addList.add(homeA);

addList.add(eduA);

return addList;

}}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* SQLDateEditor.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.cg.bean;

import java.beans.PropertyEditorSupport;

import java.sql.Date;

public class SQLDateEditor extends PropertyEditorSupport

{

public SQLDateEditor() { }

public void setAsText(String text) {

if(text.equalsIgnoreCase("getDoj()"))

{

Date date = new Date(System.currentTimeMillis());

setValue(date);

}

else

{

setValue(Date.valueOf(text));

}

}

public String getAsText() { return getValue().toString(); }

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* User .java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.cg.bean;

import java.util.ArrayList;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.context.annotation.PropertySource;

import org.springframework.context.annotation.PropertySources;

import org.springframework.stereotype.Component;

@Component("urObj")

public class User

{

@Override

public String toString() {

return "User [userName=" + userName + ", password=" + password

+ ", userLocs=" + userLocs + "]";

}

@Value("${username}")

private String userName;

@Value("${password}")

private String password;

private ArrayList<String> userLocs;

public ArrayList getUserLocs() {

return userLocs;

}

public void setUserLocs(ArrayList userLocs) { this.userLocs = userLocs; }

public String getUserName() { return userName; }

public void setUserName(String userName) { this.userName = userName; }

public String getPassword() { return password; }

public void setPassword(String password) { this.password = password; }

}

**\*\*\*\*\*\*\*\*\*\*\* TestAOPClient.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.cg.ui;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.cg.aop.bean.Instrument;

import com.cg.aop.bean.Performer;

public class TestAOPClient {

public static void main(String[] args) {

ApplicationContext ctx=new

ClassPathXmlApplicationContext("aopanno.xml");

Performer obj=(Performer)ctx.getBean("zakirHusein");

obj.perform();

}}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* TestSpelClientDemo.java\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.cg.ui;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.cg.bean.Address;

import com.cg.bean.Employee;

import com.cg.bean.Person;

import com.cg.bean.User;

public class TestSpelClientDemo {

public static void main(String[] args)

{

ApplicationContext ctx=

new ClassPathXmlApplicationContext("cg.xml");

Address a1=(Address)ctx.getBean("offAdd");

System.out.println(a1);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\* Person Info\*\*\*\*\*\*\*\*\*\*\*\*");

Person tempPer=(Person)ctx.getBean("per");

System.out.println(tempPer);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\* Emp Info\*\*\*\*\*\*\*\*\*\*\*\*");

Employee tempEmp=(Employee)ctx.getBean("empObj");

System.out.println(tempEmp);

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\* TestSpringCurrClient.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.cg.ui;

import org.springframework.context.\*;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.cg.bean.ICurrencyConvertor;

import com.cg.bean.IGreet;

import com.cg.bean.NewYearWishGreetImpl;

public class TestSpringCurrClient

{

public static void main(String[] args)

{

//annotation only supported by application contecxt bea nn factory

ApplicationContext ctx=new

ClassPathXmlApplicationContext("cg.xml");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

IGreet greet1=(IGreet)ctx.getBean("obj1");

System.out.println(greet1.greetMe());

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

ICurrencyConvertor currC1= (ICurrencyConvertor)ctx.

getBean("currencyConObj1");

System.out.println(" 50 Doller Is ? "+

currC1.dollerToRupees(50) + " RS. ");

/\*ICurrencyConvertor currC2= (ICurrencyConvertor)ctx.

getBean("currencyConObj2");

System.out.println(" 30 Yen Is ? "+

currC2.yenToRupees(30) + " RS. ");\*/

}

}

**\*\*\*\*\*\*\*\*\*\*\* TestSpringLifeCycleClient .java\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.cg.ui;

import org.springframework.beans.factory.BeanFactory;

import org.springframework.beans.factory.xml.XmlBeanFactory;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.core.io.ClassPathResource;

import org.springframework.core.io.Resource;

import com.cg.bean.ICurrencyConvertor;

import com.cg.bean.IGreet;

import com.cg.bean.NewYearWishGreetImpl;

public class TestSpringLifeCycleClient

{

public static void main(String[] args)

{

ApplicationContext ctx=

new ClassPathXmlApplicationContext("basic.xml");

IGreet greet1=(IGreet)ctx.getBean("obj1");

System.out.println(greet1.greetMe());

IGreet greet2=(IGreet)ctx.getBean("obj1");

System.out.println(greet2.greetMe());

System.out.println(greet1.hashCode());

System.out.println(greet2.hashCode());

try {

((NewYearWishGreetImpl)greet1).destroy();

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}}

**\*\*\*\*\*\*\*\*\*\*\*\*\* TestUserClient .java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.cg.ui;

//import org.springframework.boot.SpringApplication;

import org.springframework.context.ApplicationContext;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.cg.bean.User;

public class TestUserClient {

public static void main(String[] args)

{

ApplicationContext ctx=new

ClassPathXmlApplicationContext("cg.xml");

User u=(User)ctx.getBean("urObj");

System.out.println(" My User Id Id : "+

u.getUserName());

System.out.println(" My User Password Is : "+

u.getPassword());

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*TestAopClient.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.cg.ui;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.cg.aop.bean.Instrument;

import com.cg.aop.bean.Performer;

public class TestAOPClient {

public static void main(String[] args) {

ApplicationContext ctx=new

ClassPathXmlApplicationContext("aopanno.xml");

Performer obj=(Performer)ctx.getBean("zakirHusein");

obj.perform();

}

}

**\*\*\*\*\*\*\*\*\*\*\*aopanno.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:aop="http://www.springframework.org/schema/aop"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/aop

http://www.springframework.org/schema/aop/spring-aop.xsd">

<aop:aspectj-autoproxy />

<bean id="tabala" class="com.cg.aop.bean.Tabla" />

<bean id="zakirHusein" class="com.cg.aop.bean.Instrumentalist">

<property name="instrument" ref="tabala" />

<property name="song" value="Ja Ja Re Apane " />

</bean>

<!-- Aspect -->

<bean id="audienceAspect" class="com.cg.advices.AudienceAspects" />

</beans>

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*AudienceAspect.java\*\*\*\*\*\*\*\*\*\*\*\***

package com.cg.advices;

import java.util.Arrays;

import org.aspectj.lang.JoinPoint;

import org.aspectj.lang.ProceedingJoinPoint;

import org.aspectj.lang.annotation.After;

import org.aspectj.lang.annotation.AfterThrowing;

import org.aspectj.lang.annotation.Around;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Before;

@Aspect

public class AudienceAspects

{

//method will be executed before the execution

@Before("execution(\* com.cg.aop.bean.Performer.perform())")

public void performBefore(JoinPoint joinPoint) {

System.out.println("performBefore() is running!");

System.out.println("Audience is taking their seates.");

System.out.println("Audience is turnning of their cell phones.");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\* : " + joinPoint.getSignature().getName());

System.out.println("\*\*\*\*\*\*");

}

//method will be executed after the execution of

@After("execution(\* com.cg.aop.bean.Performer.perform())")

public void performAfter(JoinPoint joinPoint) {

System.out.println("performAfter() is running!");

System.out.println("Clap Clap Clap Clap");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\* : " + joinPoint.getSignature().getName());

System.out.println("\*\*\*\*\*\*");

}

//method will be executed if the customerBo interface,

//addCustomerThrowException() method is throwing an exception.

@AfterThrowing(

pointcut = "execution(\* com.cg.aop.bean.Performer.perform())",

throwing= "error")

public void performAfterThrowing(JoinPoint joinPoint, Throwable error) {

System.out.println("performAfterThrowing() is running!");

System.out.println("Bool We want our moany back");

System.out.println("\*\*\*\*\*\*\*\*\*\* : " + joinPoint.getSignature().getName());

System.out.println("Exception : " + error);

System.out.println("\*\*\*\*\*\*");

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Around("execution(\* com.cg.aop.bean.Performer.perform())")

public void performAround(ProceedingJoinPoint joinPoint) throws Throwable {

System.out.println("performAround() is running!");

System.out.println("hijacked method : " + joinPoint.getSignature().getName());

System.out.println("hijacked arguments : " + Arrays.toString(joinPoint.getArgs()));

System.out.println("performAround before is running!");

joinPoint.proceed(); //continue on the intercepted method

System.out.println("performAround after is running!");

System.out.println("\*\*\*\*\*\*");

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*basic.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:aop="http://www.springframework.org/schema/aop"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:util="http://www.springframework.org/schema/util"

xmlns:p="http://www.springframework.org/schema/p"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-4.0.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-3.0.xsd

http://www.springframework.org/schema/util

http://www.springframework.org/schema/util/spring-util-3.0.xsd

">

<context:component-scan base-package="com.cg"/>

<bean id="placeHolderConfig"

class=

"org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">

<property name="location" value="user.properties"/>

<bean id="obj1" class="com.cg.bean.NewYearWishGreetImpl"

init-method="initGreetAfterCreation"

destroy-method="destroyGreetBeforeDeletion">

<property name="name" value="Abhishek" />

<property name="year" value="2017" />

</bean>

<bean class="com.cg.bean.NewYearWishPostProcessor" />

</beans>

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*cg.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:aop="http://www.springframework.org/schema/aop"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:util="http://www.springframework.org/schema/util"

xmlns:p="http://www.springframework.org/schema/p"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-4.0.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-3.0.xsd

http://www.springframework.org/schema/util

http://www.springframework.org/schema/util/spring-util-3.0.xsd

">

<context:component-scan base-package="com.cg"/>

<context:annotation-config/>

<bean id="placeHolderConfig"

class=

"org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">

<property name="location" value="user.properties"/>

</bean>

<!-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-->

<bean id="customEditorConfigurer"

class=

"org.springframework.beans.factory.config.CustomEditorConfigurer">

<property name="customEditors">

<map>

<entry key="java.sql.Date"

value="com.cg.bean.SQLDateEditor" />

</map>

</property>

</bean>

<bean

class="org.springframework.beans.factory.config.CustomEditorConfigurer">

<property name="propertyEditorRegistrars">

<list>

<bean class="com.cg.bean.CustomDateEditorRegistrar" />

</list>

</property>

</bean>

<!-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* -->

<bean id="obj1" class="com.cg.bean.NewYearWishGreetImpl"/>

<!-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Bean Configuration\*\*\*\*\*\*\* -->

<bean id="city2" class="com.cg.bean.City">

<property name="name" value="Gaziabad" />

<property name="population" value="5000000" />

</bean>

<bean id="city3" class="com.cg.bean.City">

<property name="name" value="Kolkata" />

<property name="population" value="2000000" />

</bean>

<bean id="homeAdd" class="com.cg.bean.Address">

<property name="state" value="UP" />

<property name="country" value="India" />

<property name="zipcode" value="67877" />

<property name="city" value="#{city2}" />

</bean>

</beans>

**\*\*\*\*\*\*\*\*\*\*\*\*\*src/user.properties\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**username=CG**

**password=CG**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Thank You\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*