**1) Create an Address class with the following attributes:- street, city, state, zip, country Create an Customer class with the following attributes:- customerld, customer Name, customerContact, customerAddress. object and print details of Customer.**

**Inject the Address bean into Customer bean using setter injection Create a Test class with main() method, get Customer bean from ApplicationContext**

**Also write the JUnit Test cases for above program.**

**Modify the above application and inject the bean using constructor injection Use XML based Configuraion.**

package io.first;

public class Address {

private Long id;

protected String street;

protected String city;

protected String state;

protected String zip;

protected String country;

public Address() {

this.id = id;

this.street = street;

this.city = city;

this.state = state;

this.zip = zip;

this.country = country;

}

public Long getId()

{

return id;

}

public void setId()

{

this.id = id;

}

public String getStreet()

{

return street;

}

public void setStreet()

{

this.street = street;

}

public String getCity()

{

return city;

}

public void setCity()

{

this.city = city;

}

public String getState()

{

return state;

}

public void setState()

{

this.state = state;

}

public String getZip()

{

return zip;

}

public void setZip()

{

this.zip = zip;

}

public String getCountry()

{

return country;

}

public void setCountry()

{

this.country = country;

}

public void displayAdd()

{

System.out.println("Addresss = " + this.street + this.city + this.state + this.zip + this.country);

}

}

**#Customer class**

package io.first;

public class Customer {

protected int customer\_id;

protected String customer\_name;

protected Address customer\_address;

protected String customer\_contact;

public Customer() {

this.customer\_id = customer\_id;

this.customer\_name = customer\_name;

this.customer\_address = customer\_address;

this.customer\_contact = customer\_contact;

}

public int getCustomerId()

{

return customer\_id;

}

public void setCustomerId()

{

this.customer\_id = customer\_id;

}

public String getCustomerName()

{

return customer\_name;

}

public void setCustomerName()

{

this.customer\_name = customer\_name;

}

public Address getCustomerAddress()

{

return customer\_address;

}

public void setCustomerAddress()

{

this.customer\_address = customer\_address;

}

public String getCustomerContact()

{

return customer\_contact;

}

public void setCustomerContact()

{

this.customer\_contact = customer\_contact;

}

public void display()

{

System.out.println("Customer Details - " + this.customer\_name + this.customer\_id + this.customer\_contact);

}

}

**#Main class**

package io.first;

public class Test {

public static void main(String[] args) {

ApplicationContext context=new ClassPathXmlApplicationContext("customerbean.xml");

Customer customer = (Customer) context.getBean("Customer");

customer.display();

}

}

**#XML file**

<bean id = "Customer" class="io.first.Customer">

<property name="customer\_id" value="1"/>

<property name="customer\_name" value="JOE "/>

<property name="customer\_contact" value="783893"/>

<property name="street" ref="customer\_address"/>

<property name="city" ref="customer\_address"/>

<property name="state" ref="customer\_address"/>

<property name="zip" ref="customer\_address"/>

<property name="country" ref="customer\_address"/>

</bean>

<bean id="customer\_address" class="io.first.Address">

<property name="street" value="8"/>

<property name="city" value="Bhubaneswar"/>

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

<property name="zip" value="751003"/>

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

</bean>

</beans>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**2) Example of Injecting collections (List, Set and Map)**

**Create a class Question with following attributes: questionid, question, answers. There are 3 cases for above program.**

**a. Write a program where answers is of type List<String> or String[] b. Write a program where answers is of type Set<String> C Write a program where answers is of type Map<Integer, String>**

**In case of Map, Integer value represents answer's sequence number.**

**d. Create a Test class with main() method, get Question bean from ApplicationContext object and print question and its answers. e. Also write the JUnit Test cases for above program.**

**Use XML based configuration.**

package com.io.second;

import java.util.Iterator;

import java.util.List;

public class Question {

private int id;

private String name;

private List<String> answers\_list;

private Set<String> answers\_set;

private Map<Integer, String> answers\_map;

public int getId()

{

return id;

}

public void setId()

{

this.id = id;

}

public String getName()

{

return name;

}

public void setName()

{

this.name = name;

}

public ArrayList<Object>getList()

{

return answers\_list;

}

public void setList(ArrayList<Object> answers\_list)

{

this.answers\_list = answers\_list;

}

public Set<Object>getSet()

{

return answers\_set;

}

public void setSet(Set<Object> answers\_set)

{

this.answers\_set = answers\_set;

}

public Map<Integer, String>getMap()

{

return answers\_map;

}

public void setList(Map<Integer, String> answer\_map)

{

this.answers\_map = answers\_map;

}

public void displayList(){

System.out.println(id+" "+name);

System.out.println("answers are:");

Iterator<String> itr=answers\_list.iterator();

while(itr.hasNext()){

System.out.println(itr.next());

}

}

public void displaySet(){

System.out.println(id+" "+name);

System.out.println("answers are:");

Iterator<String> iter = answers\_set.iterator();

while(iter.hasNext()){

System.out.println(iter.next());

}

}

public void displayMap()

{

answers\_map.forEach((key, value) -> System.out.println(id + ":" + answers\_map));

}

}

**#Main Class**

package io.second;

import org.springframework.beans.factory.BeanFactory;

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

import org.springframework.core.io.ClassPathResource;

import org.springframework.core.io.Resource;

public class Test {

public static void main(String[] args) {

Resource r=new ClassPathResource("applicationContext.xml");

BeanFactory factory=new XmlBeanFactory(r);

Question q=(Question)factory.getBean("q");

q1.displayList();

Question q1=(Question)factory.getBean("q1");

q1.displaySet();

Question q2=(Question)factory.getBean("q2);

q1.displayMap();

}

}

**#XML File**

<?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:p="http://www.springframework.org/schema/p"

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

http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

<bean id="q" class="io.second.Question">

<property name="id" value="1"></property>

<property name="name" value="What is Java?"></property>

<property name="answers">

<list>

<value>Java is a programming language</value>

<value>Java is a platform</value>

<value>Java is an Island</value>

</list>

</property>

</bean>

<bean id="q1" class="io.second.Question">

<property name="id" value="2"></property>

<property name="name" value="Java is a platform"></property>

<property name="answers">

<list>

<value>Java is a programming language</value>

<value>Java is a platform</value>

<value>Java is an Island</value>

</list>

</property>

</bean>

<bean id="q2" class="io.second.Question">

<property name="id" value="3"></property>

<property name="name" value="Java is an Island"></property>

<property name="answers">

<list>

<value>Java is a programming language</value>

<value>Java is a platform</value>

<value>Java is an Island</value>

</list>

</property>

</bean>

</beans>

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**3) Example on autowiring**

**Design and Develop a Banking Application as follows: a Create a BankAccount class with following attributes: accountid,**

**accountHolderName, accountType, account Balance**

**b Create an interface BankAccountRepository with following methods:**

**public double getBalance(long accountid) public double updateBalance(long accountld, double newBalance):**

**Note: Above method returns updated balance.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**4) Example on @Controller, @Service, @Repository, @Autowired, @Configuration and @Bean**

**Modify the above application, use annotations and java based configuration.**

* @repository

public interface BankCardDAO {

public BankCardDTO insertBankCard(String bankName, String cardNumber, String createDate);

}

// Annotated this object as a DAO function bean.

@Repository("bcDao")

public class BankCardDAOImpl implements BankCardDAO {

@Override

public BankCardDTO insertBankCard(String bankName, String cardNumber, String createDate) {

BankCardDTO ret = new BankCardDTO();

ret.setBankName(bankName);

ret.setCardNumber(cardNumber);

ret.setCreateDate(createDate);

System.out.println("Bank card has been inserted by BankCardDAOImpl. Bank name : " + bankName + " , card number : " + cardNumber + " , create date : " + createDate);

return ret;

}

}

* @Service

public interface BankCardManager {

public BankCardDTO createBankCard(String bankName, String cardNumber, String createDate);

}

// Annotated this object as a service bean.

@Service("bcManager")

public class BankCardManagerImpl implements BankCardManager {

@Autowired

private BankCardDAO bankCardDao;

@Override

public BankCardDTO createBankCard(String bankName, String cardNumber, String createDate) {

System.out.println("Bank card has been created by BankCardManagerImpl. Bank name : " + bankName + " , card number : " + cardNumber + " , create date : " + createDate);

return this.bankCardDao.insertBankCard(bankName, cardNumber, createDate);

}

}

* @controller

// Annotated this object as a controller bean.

@Controller("bcController")

public class BankCardController {

@Autowired

private BankCardManager bcManager;

public BankCardDTO createBankCard(String bankName, String cardNumber, String createDate)

{

return this.bcManager.createBankCard(bankName, cardNumber, createDate);

}

}

public class TestAutowireUseAnnotation {

public static void main(String[] args) {

// Initiate Spring application context.

ApplicationContext springAppCtx = new ClassPathXmlApplicationContext("AutowireByAnnotationBeanSettings.xml");

// Get @Controller annotated bean by id.

BankCardController bcController = (BankCardController)springAppCtx.getBean("bcController");

bcController.createBankCard("Bank Of China", "BOC888888", "2017/08/08");

}

}

Taking another example for @autowired, @configuration

* @autowired

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

public class TextEditor {

@Autowired

private SpellChecker spellChecker;

public TextEditor() {

System.out.println("Inside TextEditor constructor." );

}

public SpellChecker getSpellChecker( ){

return spellChecker;

}

public void spellCheck(){

spellChecker.checkSpelling();

}

}

* @configuration

DemoManager.java and DemoManagerImpl.java

public interface DemoManager {

public String getServiceName();

}

public class DemoManagerImpl implements DemoManager

{

@Override

public String getServiceName()

{

return "My first service with Spring 3";

}

}

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import com.howtodoinjava.core.beans.DemoManager;

import com.howtodoinjava.core.beans.DemoManagerImpl;

@Configuration

public class ApplicationConfiguration {

@Bean(name="demoService")

public DemoManager helloWorld()

{

return new DemoManagerImpl();

}

}

import org.springframework.context.ApplicationContext;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

import com.howtodoinjava.core.beans.DemoManager;

import com.howtodoinjava.core.config.ApplicationConfiguration;

public class VerifySpringCoreFeature

{

public static void main(String[] args)

{

ApplicationContext context = new AnnotationConfigApplicationContext(ApplicationConfiguration.class);

DemoManager obj = (DemoManager) context.getBean("demoService");

System.out.println( obj.getServiceName() );

}

}

**#XML File**

<?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

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

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

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

<context:annotation-config/>

<!-- Definition for textEditor bean -->

<bean id = "textEditor" class = "io.third.TextEditor">

</bean>

<!-- Definition for spellChecker bean -->

<bean id = "spellChecker" class = "io.third.SpellChecker">

</bean>

</beans>

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**5) Write a program to demonstrate use of @Resource, @inject, @Required annotations**

# @resource

pom.xml

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.spring</groupId>

<artifactId>SpringResourceAnnotationExample</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-beans -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>5.0.8.RELEASE</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.0.8.RELEASE</version>

</dependency>

</dependencies>

<build>

<finalName>${project.artifactId}</finalName>

</build>

</project>

#COMPANY CLASS

package com.spring.pojo;

public class Company {

private String name;

private String location;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getLocation() {

return location;

}

public void setLocation(String location) {

this.location = location;

}

@Override

public String toString() {

return "Company [name=" + name + ", location=" + location + "]";

}

}

IMPLEMENTATION EMPLOYEE

EMPLOYEE.JAVA

package com.spring.pojo;

import javax.annotation.Resource;

public class Employee {

private String id;

private String name;

@Resource(name="mycompany")

private Company company;

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Company getCompany() {

return company;

}

public void setCompany(Company company) {

this.company = company;

}

@Override

public String toString() {

return "Employee [id=" + id + ", name=" + name + ", company=" + company.toString() + "]";

}

}

IMPLEMENTAION OF UTILITY CLASS

APPMAIN. JAVA

package com.spring.util;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.spring.pojo.Employee;

public class AppMain {

@SuppressWarnings("resource")

public static void main(String[] args) {

ApplicationContext ac = new ClassPathXmlApplicationContext("resource-annotation.xml");

Employee emp = ac.getBean("myemployee", Employee.class);

System.out.println(emp.toString());

}

}

RESOURCE ANNOTATION FILE

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

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

<!-- To activate the '@Resource' annotation in the spring framework -->

<context:annotation-config />

<bean id="mycompany" class="com.spring.pojo.Company">

<property name="name" value="Test Pvt. Ltd." />

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

</bean>

<bean id="myemployee" class="com.spring.pojo.Employee">

<property name="id" value="123456" />

<property name="name" value="Charlotte O' Neil" />

</bean>

</beans>

#@INJECT

public class Address {

private String street;

private String city;

private String state;

public Address() {

}

public String getStreet() {

return street;

}

public void setStreet(String street) {

this.street = street;

}

public String getCity() {

return city;

}

public void setCity(String city) {

this.city = city;

}

public String getState() {

return state;

}

public void setState(String state) {

this.state = state;

}

}

TEST

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Test {

public static void main(String[] args) {

ApplicationContext context =

new ClassPathXmlApplicationContext(("inject.xml"));

Employee employee = (Employee)context.getBean("employee");

if(employee.getAddress()==null){

System.out.println("The Employee Name : " + employee.getName());

System.out.println("The Employee Age : " + employee.getAge());

System.out.println("The Employee Address : " + "is not provided");

}

else{

System.out.println("The Employee Name : " + employee.getName());

System.out.println("The Employee Age : " + employee.getAge());

System.out.println("The Employee Address : " +

employee.getAddress().getStreet() + " " +

employee.getAddress().getCity() + " " +

employee.getAddress().getState());

}

}

}

#SPRING.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

http://www.springframework.org/schema/beans/spring-beans-3.1.xsd

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

http://www.springframework.org/schema/context/spring-context-3.1.xsd">

<bean

class="org.springframework.beans.factory.

annotation.AutowiredAnnotationBeanPostProcessor">

</bean>

<bean id="employee" class="com.hubberspot.spring.inject.Employee">

<property name="age" value="28" />

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

</bean>

<bean id="address" class="com.hubberspot.spring.inject.Address">

<property name="street" value="Town Hall Street" />

<property name="city" value="Pune" />

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

</bean>

</beans>

#@REQUIRED

ACTIVATING @REQUIRED

<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

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

<bean class="org.springframework.beans.factory.annotation.RequiredAnnotationBeanPostProcessor"/>

</beans>

#DEPENDENCIES

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.spring</groupId>

<artifactId>SpringRequiredAnnotation</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-beans -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>5.0.6.RELEASE</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.0.6.RELEASE</version>

</dependency>

</dependencies>

<build>

<finalName>${project.artifactId}</finalName>

</build>

</project>

EMPLOYEE.JAVA

package com.spring.pojo;

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

public class Employee {

private String name;

private String designation;

private String company;

@Required

public void setName(String name) {

this.name = name;

}

public String getName() {

return name;

}

@Required

public void setDesignation(String designation) {

this.designation = designation;

}

public String getDesignation() {

return designation;

}

public void setCompany(String company) {

this.company = company;

}

public String getCompany() {

return company;

}

@Override

public String toString() {

return "Employee [name=" + name + ", designation=" + designation + ", company=" + company + "]";

}

}

#APP MAIN .JAVA

package com.spring;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.spring.pojo.Employee;

public class AppMain {

@SuppressWarnings("resource")

public static void main(String[] args) {

ApplicationContext ac = new ClassPathXmlApplicationContext("required-annotation.xml");

Employee emp = ac.getBean("myemployee", Employee.class);

System.out.println(emp.toString());

}

}

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

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

<!-- Used to activate the @Required annotation in Spring -->

<context:annotation-config />

<bean id="myemployee" class="com.spring.pojo.Employee">

<!-- Required property -->

<property name="name" value="Charlotte O' Neil" />

<!-- Required property -->

<property name="designation" value="Technical Leader" />

<property name="company" value="Test Ltd." />

</bean>

</beans>

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**6) Example of @Component, @Value, @PropertySource & Environment**

**a. Create a dbConfig.properties file which contains database configuration details**

**like driver class name, dburl, username, password.**

**b. Create a Java class in which you have to read all properties and display on a**

**console. (Use @Component, @Value or Environment and @Property Resource).**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

7) Write a Java program to demonstrate SPEL (Spring Expression language)

import org.springframework.expression.Expression;

import org.springframework.expression.ExpressionParser;

import org.springframework.expression.spel.standard.SpelExpressionParser;

public class Test {

public static void main(String[] args) {

ExpressionParser parser = new SpelExpressionParser();

Expression exp = parser.parseExpression("'Hello SPEL'");

String message = (String) exp.getValue();

System.out.println(message);

//OR

//System.out.println(parser.parseExpression("Hello").getValue());

}

}

#Using concat() method with String

ExpressionParser parser = new SpelExpressionParser();

Expression exp = parser.parseExpression("Hi.concat('!')");

String message = (String) exp.getValue();

System.out.println(message);

#Converting String into byte array

Expression exp = parser.parseExpression("'HI VEDANT'.bytes");

byte[] bytes = (byte[]) exp.getValue();

for(int i=0;i<bytes.length;i++){

System.out.print(bytes[i]+" ");

}

#Getting length after converting string into bytes

Expression exp = parser.parseExpression("'HI VEDANT'.bytes.length");

int length = (Integer) exp.getValue();

System.out.println(length);

#Converting String contents into uppercase letter

Expression exp = parser.parseExpression("new String('hello world').toUpperCase()");

String message = exp.getValue(String.class);

System.out.println(message);

//OR

System.out.println(parser.parseExpression("'hello world'.toUpperCase()").getValue());

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

8) Write a Java program to demonstrate InitializingBean and DisposableBean.

Try Different ways:

(Use init-method and destroy-method in xml config file)

(Use @PostConstruct and @PreDestroy)

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

import org.springframework.beans.factory.DisposableBean;

import org.springframework.beans.factory.InitializingBean;

import org.springframework.stereotype.Component;

@Component

public class DatabaseInitiaizer implements InitializingBean, DisposableBean {

private List < User > listOfUsers = new ArrayList < > ();

@Override

public void afterPropertiesSet() throws Exception {

User user = new User(1, "User");

User user1 = new User(2, "Admin");

User user2 = new User(3, "SuperAdmin");

listOfUsers.add(user);

listOfUsers.add(user1);

listOfUsers.add(user2);

System.out.println("-----------List of users added in init() method ------------");

for (Iterator < User > iterator = listOfUsers.iterator(); iterator.hasNext();) {

User user3 = (User) iterator.next();

System.out.println(user3.toString());

}

// save to database

}

@Override

public void destroy() {

// Delete from database

listOfUsers.clear();

System.out.println("-----------After of users removed from List in destroy() method ------------");

for (Iterator < User > iterator = listOfUsers.iterator(); iterator.hasNext();) {

User user3 = (User) iterator.next();

System.out.println(user3.toString());

}

System.out.println("List is clean up ..");

}

}

Create POJO - User.java

public class User {

private Integer id;

private String name;

public User() {}

public User(Integer id, String name) {

super();

this.id = id;

this.name = name;

}

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

@Override

public String toString() {

return "User [id=" + id + ", name=" + name + "]";

}

}

Annotation Based Configuration - AppConfig.java

Create AppConfig class and write the following code in it.

package net.javaguides.spring;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.Configuration;

@Configuration

@ComponentScan(basePackages = "net.javaguides.spring")

public class AppConfig {

}

Application.java

Let's create a main class and run an application.

package net.javaguides.spring;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class Application {

public static void main(String[] args) {

AnnotationConfigApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);

context.close();

}

}

Output

-----------List of users added in init() method ------------

User [id=1, name=User]

User [id=2, name=Admin]

User [id=3, name=SuperAdmin]

-----------After of users removed from List in destroy() method -------

#POM.XML

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>net.javaguides.spring</groupId>

<artifactId>spring-bean-lifecycle</artifactId>

<version>0.0.1-SNAPSHOT</version>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.1.0.RELEASE</version>

</dependency>

</dependencies>

<build>

<sourceDirectory>src/main/java</sourceDirectory>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.5.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

9) Write a Java program to demonstrate Complete Bean Life cycle.

HelloBean.java(filename)

package beans;

public class HelloBean {

public void init() throws Exception

{

System.out.println(

"Bean HelloWorld has been "

+ "instantiated and I'm "

+ "the init() method");

}

public void destroy() throws Exception

{

System.out.println(

"Container has been closed "

+ "and I'm the destroy() method");

}

}

#spring.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

http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

<bean id="hw" class="beans.HelloBean"

init-method="init" destroy-method="destroy"/>

</beans>

#calling the bean initialized

package test;

import org.springframework

.context

.ConfigurableApplicationContext;

import org.springframework

.context.support

.ClassPathXmlApplicationContext;

import beans.HelloWorld;

// Driver class

public class Client {

public static void main(String[] args)

throws Exception

{

// Loading the Spring XML configuration

// file into the spring container and

// it will create the instance of

// the bean as it loads into container

ConfigurableApplicationContext cap

= new ClassPathXmlApplicationContext(

"resources/spring.xml");

// It will close the spring container

// and as a result invokes the

// destroy() method

cap.close();

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

10) Write a java program to demonstrate ApplicationContextAware interface.

package SPRING;

import org.springframework.beans.BeansException;

import org.springframework.beans.factory.BeanNameAware;

import org.springframework.context.ApplicationContext;

import org.springframework.context.ApplicationContextAware;

public class Triangle implements ApplicationContextAware, BeanNameAware

{

private Point pointA;

private Point pointB;

private Point pointC;

private ApplicationContext context = null;

public void setPointA(Point pointA) {

this.pointA = pointA;

}

public void setPointB(Point pointB) {

this.pointB = pointB;

}

public void setPointC(Point pointC) {

this.pointC = pointC;

}

public void draw()

{

System.out.println("PointA is ("+pointA.getX()+", "+pointA.getY()+")");

System.out.println("PointB is ("+pointB.getX()+", "+pointB.getY()+")");

System.out.println("PointC is ("+pointC.getX()+", "+pointC.getY()+")");

}

@Override

public void setApplicationContext(ApplicationContext context) throws BeansException

{

this.context = context;

}

@Override

public void setBeanName(String beanName)

{

System.out.println("Bean name is: "+beanName);

}

}

#POINT.JAVA

package SPRING;

public class Point

{

private int x;

private int y;

public int getX() {

return x;

}

public void setX(int x) {

this.x = x;

}

public int getY() {

return y;

}

public void setY(int y) {

this.y = y;

}

}

#SPRING.XML

beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.springframework.org/schema/beans" xsi:schemalocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">

<bean autowire="byName" class="SPRINGNI.Triangle" id="triangle"></bean>

<bean class="com.dineshonjava.sdnext.contextAware.tutorial.Point" id="pointA">

<property name="x" value="0"></property>

<property name="y" value="0"></property>

</bean>

<bean class="com.dineshonjava.sdnext.contextAware.tutorial.Point" id="pointB">

<property name="x" value="-20"></property>

<property name="y" value="0"></property>

</bean>

<bean class="com.dineshonjava.sdnext.contextAware.tutorial.Point" id="pointC">

<property name="x" value="20"></property>

<property name="y" value="0"></property>

</bean>

</beans>

DrawingApp.java

package com.sdnext.contextAware.tutorial;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class DrawingApp

{

public static void main(String[] args)

{

ApplicationContext context = new ClassPathXmlApplicationContext("spring.xml");

Triangle triangle = (Triangle) context.getBean("triangle");

triangle.draw();

}

}