**Spring Core Assignments**

1. Create an Address class with the following attributes:- street, city, state, zip, country Create an Customer class with the following attributes:- customerId, customerName, customerContact, customerAddress.

Inject the Address bean into Customer bean using setter injection

Create a Test class with main() method, get Customer bean from ApplicationContext object and print details of Customer.

Also write the JUnit Test cases for above program.

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

Ans:

I have created three classes

Customer.java //main class

Address.java

Custo.java

////////////////////////////////

Customer.java

package org.assignments.ass1;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class customer {

public static void main(String[] args) {

ApplicationContext c=new ClassPathXmlApplicationContext("NewFile.xml");

custo ca=(custo) c.getBean("ca");

ca.details();

}

}

///////////////////////////////////////

Custo.java

**package** org.assignments.ass1;

**import** **static** org.junit.Assert.*assertEquals*;

**import** **static** org.junit.jupiter.api.Assertions.*assertEquals*;

**import** org.junit.jupiter.api.Test;

**public** **class** custo {

**int** customerId, customerContact;

String customerName;

addres customerAddress;

**public** **int** getCustomerId() {

**return** customerId;

}

**public** **void** setCustomerId(**int** customerId) {

**this**.customerId = customerId;

}

**public** **int** getCustomerContact() {

**return** customerContact;

}

**public** **void** setCustomerContact(**int** customerContact) {

**this**.customerContact = customerContact;

}

**public** String getCustomerName() {

**return** customerName;

}

**public** **void** setCustomerName(String customerName) {

**this**.customerName = customerName;

}

**public** addres getCustomerAddress() {

**return** customerAddress;

}

**public** **void** setCustomerAddress(addres customerAddress) {

**this**.customerAddress = customerAddress;

}

@Test

**void** data()

{

*assertEquals*(**this**.customerId,20,"data doesnt matched");

}

**void** details()

{

System.***out***.println(**this**.customerId+" "+**this**.getCustomerContact()+" "+**this**.customerName+"address details: "+**this**.customerAddress.getStreet()+","+**this**.customerAddress.getCity()+","+**this**.customerAddress.getCountry()+" "+**this**.customerAddress.getZip());

}

}

//////////////////////////////////////////////////////////////////

Address.java

**package** org.assignments.ass1;

**public** **class** addres {

**private** String street,city,country;

**private** **int** zip;

**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 getCountry() {

**return** country;

}

**public** **void** setCountry(String country) {

**this**.country = country;

}

**public** **int** getZip() {

**return** zip;

}

**public** **void** setZip(**int** zip) {

**this**.zip = zip;

}

}

///////////////////////////////////////////////

NewFile.xml

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.springframework.org/dtd/spring-beans.dtd">

<beans>

<bean id=*"ca"* class=*"org.assignments.ass1.custo"*>

<property name=*"customerId"*>

<value>20</value>

</property>

<property name=*"customerContact"* value=*"9182950"*/>

<property name=*"customerName"* value=*"karthik"*/>

<property name=*"customerAddress"* ref=*"addresslocation"* />

</bean>

<bean id=*"addresslocation"* class=*"org.assignments.ass1.addres"*>

<property name=*"zip"* value=*"144411"*/>

<property name=*"city"* value=*"lpu\_street"*/>

<property name=*"street"* value=*"lpu\_street"*/>

<property name=*"country"* value=*"india"*/> </bean></beans>

1. 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.

* + 1. Write a program where answers is of type List<String> or String []
    2. Write a program where answers is of type Set<String>
    3. Write a program where answers is of type Map<Integer, String>

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

* + 1. Create a Test class with main() method, get Question bean from ApplicationContext object and print question and its answers.
    2. Also write the JUnit Test cases for above program.

* + Use XML based configuration.

Ans:

Collec.java

**package** secound;

**import** java.util.\*;

**public** **class** collec {

List<String> list=**new** ArrayList<String>();

Map<Integer,String> map=**new** HashMap<Integer,String>();

Set<String> hash\_Set = **new** HashSet<String>();

**public** List<String> getList() {

**return** list;

}

**public** **void** setList(List<String> list) {

**this**.list = list;

}

**public** Map getMap() {

**return** map;

}

**public** **void** setMap(Map map) {

**this**.map = map;

}

**public** Set<String> getHash\_Set() {

**return** hash\_Set;

}

**public** **void** setHash\_Set(Set<String> hash\_Set) {

**this**.hash\_Set = hash\_Set;

}

**void** details()

{

**for**(String i:list)

{

System.***out***.println(i);

}

}

**void** details\_set()

{

**for**(String i:hash\_Set)

{

System.***out***.println(i);

}

}

Iterator hmIterator = map.entrySet().iterator();

**void** details\_map()

{

**while** (hmIterator.hasNext()) {

Map.Entry mapElement = (Map.Entry)hmIterator.next();

System.***out***.println(mapElement.getKey() + " : " + mapElement.getValue());

}

}

@Test

**void** data()

{

*assertEquals*(**this**.list.get(0),"siva","data doesnt matched");

}

}

Test.java

**package** secound;

**import** org.assignments.ass1.custo;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** test {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

ApplicationContext c=**new** ClassPathXmlApplicationContext("b.xml");

collec ca=(collec) c.getBean("ca");

ca.details();

ca.details\_set();

ca.details\_map();

}

}

b.xml

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.springframework.org/dtd/spring-beans.dtd">

<beans>

<bean id=*"ca"* class=*"secound.collec"* autowire=*"byName"*>

<property name=*"list"*>

<list>

<value>siva</value>

<value>sai</value>

<value>karthik</value>

</list>

</property>

<property name=*"hash\_Set"*>

<set>

<value>siva</value>

<value>sai</value>

<value>karthik</value>

</set>

</property>

<property name=*"map"*>

<map>

<entry key=*"1"* value=*"siva"*/>

<entry key=*"2"* value=*"sai"*/>

<entry key=*"3"* value=*"karthik"*/>

</map>

</property>

</bean>

</beans>

3.Design and Develop a Banking Application as follows:

* + 1. Create a BankAccount class with following attributes: accountId, accountHolderName, accountType, accountBalance

* + 1. Create an interface BankAccountRepository with following methods: public double getBalance(long accountId) public double updateBalance(long accountId, double newBalance): Note: Above method returns updated balance.

* + 1. Create a class BankAccountepositoryImpl that implements BankAccountRepository interface.

You can use database or any collection object as persistence store.

* + 1. Create an interface BankAccountService with following methods:

public double withdraw(long accountId, double balance) public double deposit(long accountId, double balance) public double getBalance(long accountId)

public boolean fundTransfer(long fromAccount, long toAccount, double amont)

* + 1. Create a class BankAccountServiceImpl that implements BankAccountService interface.

* + 1. Create a class BankAccount controller with following operations:

public double withdraw(long accountId, double balance) public double deposit(long accountId, double balance) public double getBalance(long accountId)

public boolean fundTransfer(long fromAccount, long toAccount, double amont)

* + 1. Create a Test class with main() method, get BankAccountController bean object from ApplicationContext and perform all the operations.

* + 1. Also write the JUnit Test cases for above program.

* + Use XML based configuration and perform autowiring with different types. (byName, byType and constructor). Use one autowiring type at a time.

Ans:

Main.java

package third;

import org.assignments.ass1.custo;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class mainn {

public static void main(String[] args) {

// TODO Auto-generated method stub

ApplicationContext c=new ClassPathXmlApplicationContext("a.xml");

Bankcontroller ca=(Bankcontroller) c.getBean("ca");

ca.detail();

}}

BankAccount.java

**package** third;

**public** **class** BankAccount {

String accountHolderName, accountType;

**double** accountBalance;

**long** accountId;

**public** String getAccountHolderName() {

**return** accountHolderName;

}

**public** **void** setAccountHolderName(String accountHolderName) {

**this**.accountHolderName = accountHolderName;

}

**public** String getAccountType() {

**return** accountType;

}

**public** **void** setAccountType(String accountType) {

**this**.accountType = accountType;

}

**public** **double** getAccountBalance() {

**return** accountBalance;

}

**public** **void** setAccountBalance(**int** accountBalance) {

**this**.accountBalance = accountBalance;

}

**public** **long** getAccountId() {

**return** accountId;

}

**public** **void** setAccountId(**int** accountId) {

**this**.accountId = accountId;

}

**void** detail()

{

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

}

}

BankAccountepositoryImpl.java

**package** third;

**import** java.util.List;

**public** **class** BankAccountepositoryImpl **implements** BankAccountRepository,BankAccountService {

List<BankAccount> b;

**public** List<BankAccount> getB() {

**return** b;

}

**public** **void** setB(List<BankAccount> b) {

**this**.b = b;

}

@Override

**public** **double** getBalance(**long** accountId) {

BankAccount j = **null**;

**for**( BankAccount i:b)

{ j=i;

**if**(i.accountId==accountId)

{

**break**;

}

}

**return** j.accountBalance;

}

@Override

**public** **double** updateBalance(**long** accountId, **double** newBalance) {

// **TODO** Auto-generated method stub

BankAccount i=**null**;

**for**(BankAccount x:b)

{

i=x;

**if**(x.accountId==accountId)

{

**break**;

}

}

i.accountBalance=newBalance;

**return** i.accountBalance;

}

@Override

**public** **double** withdraw(**long** accountId, **double** balance) {

BankAccount i = **null**;

**for**(BankAccount x:b)

{ i=x;

**if** (x.accountId==accountId)

{

**break**;

}

}

**return** i.accountBalance-balance;

}

@Override

**public** **double** deposit(**long** accountId, **double** balance) {

BankAccount i = **null**;

**for**(BankAccount x:b)

{ i=x;

**if** (x.accountId==accountId)

{

**break**;

}

}

System.***out***.println("account balce is"+i.accountBalance+balance);

**return** i.accountBalance+balance;

}

@Override

**public** **boolean** fundTransfer(**long** fromAccount, **long** toAccount, **double** amont) {

// **TODO** Auto-generated method stub

**if**(deposit(fromAccount, amont)>amont)

{

deposit(toAccount, amont);

**return** **true**;

}

**return** **false**;

}

}

Bankcontroller.java

**package** third;

**import** java.util.Scanner;

**public** **class** Bankcontroller **extends** BankAccountepositoryImpl{

**public** **void** detail() {

// **TODO** Auto-generated method stub

System.***out***.println("1.withdraw /n2.deposit /n3.getbalance /n4.fundtransfer");

Scanner input=**new** Scanner(System.***in***);

**int** a=input.nextInt();

**if**(a==1)

{

System.***out***.println("money after withdraw the :"+withdraw(1170,2500));

}

**else** **if**(a==2)

{

System.***out***.println("money after deposit the :"+deposit(1170,2500));

}

**else** **if**(a==3)

{

System.***out***.println("balance:"+getBalance(1170));

}

**else**

{

System.***out***.println("money after the :"+fundTransfer(1170,1171,2500));

}

}

}

a.xml

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.springframework.org/dtd/spring-beans.dtd">

<beans>

<bean id=*"ca"* class=*"third.Bankcontroller"* autowire=*"byType"*>

</bean>

<bean id=*"b"* class=*"third.BankAccount"*>

<property name=*"accountType"* value=*"saving"*/>

<property name=*"accountHolderName"* value=*"karthik"*/>

<property name=*"accountBalance"* value=*"150000"*/>

<property name=*"accountId"* value=*"1170"*/>

</bean>

<!-- String accountHolderName, accountType;

int accountBalance,accountId;

-->

<bean id=*"c"* class=*"third.BankAccount"*>

<property name=*"accountType"* value=*"saving"*/>

<property name=*"accountHolderName"* value=*"siva"*/>

<property name=*"accountBalance"* value=*"10000"*/>

<property name=*"accountId"* value=*"1171"*/>

</bean>

</beans>

1. Example on @Controller, @Service, @Repository, @Autowired, @Configuration and

@Bean

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

1. Write a program to demonstrate use of @Resource, @Inject, @Required annotations

1. Example of @Component, @Value, @PropertySource & Environment

* + 1. Create a dbConfig.properties file which contains database configuration details like driver class name, dburl, username, password.
    2. Create a Java class in which you have to read all properties and display on a console. (Use @Component, @Value or Environment and @PropertyResource).

Ans

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

**public** **class** DBConnection {

@Value("${DB\_DRIVER\_CLASS}")

**private** String driverClass;

@Value("${DB\_URL}")

**private** String dbURL;

@Value("${DB\_USERNAME}")

**private** String userName;

@Value("${DB\_PASSWORD}")

**private** **char**[] password;

**public** DBConnection() {

}

**public** **void** printDBConfigs() {

System.***out***.println("Driver Class = " + driverClass);

System.***out***.println("DB URL = " + dbURL);

System.***out***.println("User Name = " + userName);

// Never do below in production environment :D

System.***out***.println("Password = " + String.*valueOf*(password));

}

}

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

Ans

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.AbstractApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**class** adds{

**int** sum;

**public** **int** getSum() {

**return** sum;

}

**public** **void** setSum(**int** sum) {

**this**.sum = sum;

}

}

**public** **class** spring7 {

**public** **static** **void** main(String args[])

{

ApplicationContext c=**new** ClassPathXmlApplicationContext("d.xml");

adds a=c.getBean("ca", adds.**class**);

System.***out***.println(a.getSum());

((AbstractApplicationContext) c).close();

}

}

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.springframework.org/dtd/spring-beans.dtd">

<beans>

<bean id=*"ca"* class=*"adds"* >

<property name=*"sum"* value=*"#{2+5}"*></property>

</bean>

</beans>

1. 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** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.AbstractApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**class** HelloWorld

{

**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( "Conatiner has been closed " + "and I'm the destroy() method");

}

}

**public** **class** spring9 {

**public** **static** **void** main(String args[])

{

ApplicationContext c=**new** ClassPathXmlApplicationContext("c.xml");

((AbstractApplicationContext) c).close();

}

}

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.springframework.org/dtd/spring-beans.dtd">

<beans>

<bean id=*"hw"* class=*"HelloWorld"*

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

</beans>

1. Write a Java program to demonstrate Complete Bean Life cycle.

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.AbstractApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**class** HelloWorld

{

**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( "Conatiner has been closed " + "and I'm the destroy() method");

}

}

**public** **class** spring9 {

**public** **static** **void** main(String args[])

{

ApplicationContext c=**new** ClassPathXmlApplicationContext("c.xml");

((AbstractApplicationContext) c).close();

}

}

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.springframework.org/dtd/spring-beans.dtd">

<beans>

<bean id=*"hw"* class=*"HelloWorld"*

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

</beans>

1. Write a java program to demonstrate ApplicationContextAware interface.

Ans:

**package** org.assignments.ass1;

**import** **static** org.junit.Assert.*assertEquals*;

**import** **static** org.junit.jupiter.api.Assertions.*assertEquals*;

**import** org.junit.jupiter.api.Test;

**import** org.springframework.beans.BeansException;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.ApplicationContextAware;

**public** **class** custo **implements** ApplicationContextAware {

**int** customerId, customerContact;

String customerName;

addres customerAddress;

**public** **int** getCustomerId() {

**return** customerId;

}

**public** **void** setCustomerId(**int** customerId) {

**this**.customerId = customerId;

}

**public** **int** getCustomerContact() {

**return** customerContact;

}

**public** **void** setCustomerContact(**int** customerContact) {

**this**.customerContact = customerContact;

}

**public** String getCustomerName() {

**return** customerName;

}

**public** **void** setCustomerName(String customerName) {

**this**.customerName = customerName;

}

**public** addres getCustomerAddress() {

**return** customerAddress;

}

**public** **void** setCustomerAddress(addres customerAddress) {

**this**.customerAddress = customerAddress;

}

@Test

**void** data()

{

*assertEquals*(**this**.customerId,20,"data doesnt matched");

}

**void** details()

{

System.***out***.println(**this**.customerId+" "+**this**.getCustomerContact()+" "+**this**.customerName+"address details: "+**this**.customerAddress.getStreet()+","+**this**.customerAddress.getCity()+","+**this**.customerAddress.getCountry()+" "+**this**.customerAddress.getZip());

}

@Override

**public** **void** setApplicationContext(ApplicationContext context) **throws** BeansException {

// **TODO** Auto-generated method stub

System.***out***.println("the context name of the bean is :"+context);

}

}