Spring Core Assignments

Submitted by – Sanket Bolamwar

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.

**Using Setter Injection.**

1. **Test Class**

**package** AssignmentQ1;

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

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

**public** **class** Test {

**private** **static** ApplicationContext *context*;

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

*context* = **new** ClassPathXmlApplicationContext("AssignmentQ1.xml");

Address address = *context*.getBean("Address", Address.**class**);

System.***out***.println("Address Class Bean");

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

Customer customer = *context*.getBean("Customer", Customer.**class**);

System.***out***.println("\n\nCustomer Class Bean ");

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

}

}

1. **Address Class**

**package** AssignmentQ1;

**public** **class** Address {

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

**public** Address(String street, String city, String state, String zip, String country) {

**super**();

**this**.street = street;

**this**.city = city;

**this**.state = state;

**this**.zip = zip;

**this**.country = country;

}

**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;

}

**public** String getZip() {

**return** zip;

}

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

**this**.zip = zip;

}

**public** String getCountry() {

**return** country;

}

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

**this**.country = country;

}

@Override

**public** String toString() {

**return** "Address [street=" + street + ", city=" + city + ", state=" + state + ", zip=" + zip + ", country="

+ country + "]";

}

}

1. **Customer Class**

**package** AssignmentQ1;

**public** **class** Customer {

**private** String customerId, customerName, customerContact;

**private** Address CustomerAddress;

**public** Customer(String customerId, String customerName, String customerContact, Address customerAddress) {

**super**();

**this**.customerId = customerId;

**this**.customerName = customerName;

**this**.customerContact = customerContact;

CustomerAddress = customerAddress;

}

**public** String getCustomerId() {

**return** customerId;

}

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

**this**.customerId = customerId;

}

**public** String getCustomerName() {

**return** customerName;

}

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

**this**.customerName = customerName;

}

**public** String getCustomerContact() {

**return** customerContact;

}

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

**this**.customerContact = customerContact;

}

**public** Address getCustomerAddress() {

**return** CustomerAddress;

}

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

CustomerAddress = customerAddress;

}

@Override

**public** String toString() {

**return** "Customer [customerId=" + customerId + ", customerName=" + customerName + ", customerContact="

+ customerContact + ", CustomerAddress=" + CustomerAddress + "]";

}

}

1. **.Xml File**

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

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

xmlns=*"http://www.springframework.org/schema/beans"*

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

xmlns:mvc=*"http://www.springframework.org/schema/mvc"*

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

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

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

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

<!-- creating beans using setter injections-->

<!-- creating bean for address -->

<bean id=*"Address"* class=*"AssignmentQ1.Address"*>

<property name=*"street"* value=*"Street No.6"*/>

<property name=*"city"* value=*"Ballarpur"*/>

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

<property name=*"zip"* value=*"442701"*/>

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

</bean>

<!-- creating bean for customer-->

<bean id = *"Customer"* class= *"AssignmentQ1.Customer"*>

<property name=*"customerId"* value=*"1"*/>

<property name=*"customerName"* value=*"Sanket Bollamwar"*/>

<property name=*"customerContact"* value=*"7709496916"*/>

<property name=*"CustomerAddress"* ref=*"Address"*/>

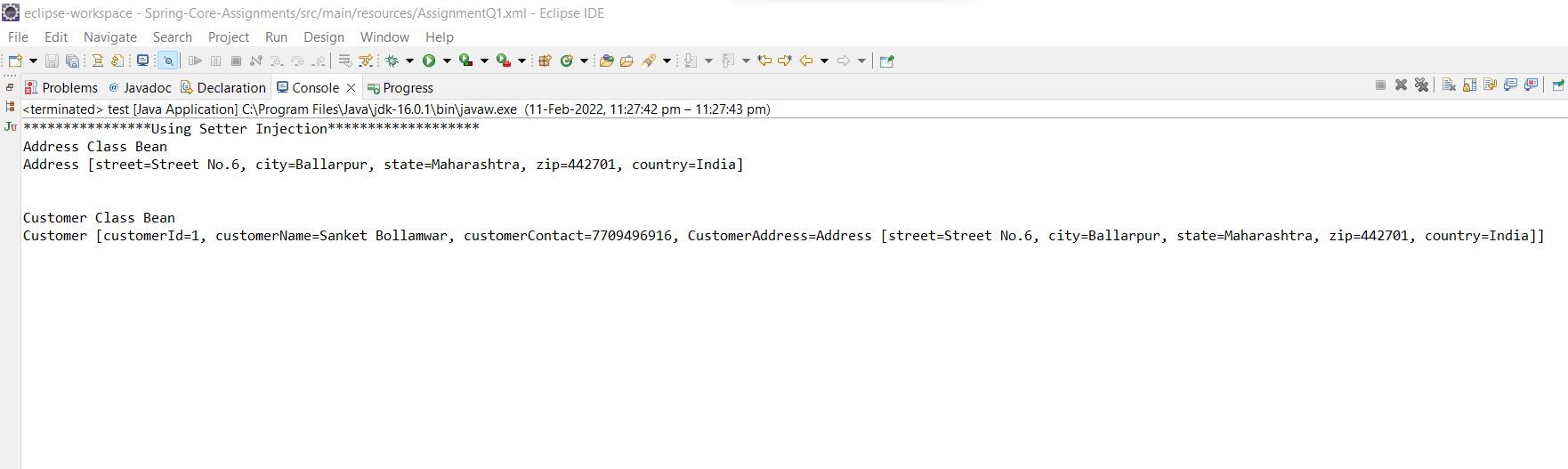
</bean>

<context:annotation-config />

<context:component-scan base-package=*"controller"* />

</beans>

1. **Output**



**Using Constructor Injection:**

1. **Test Class**

**package** AssignmentQ1;

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

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

**public** **class** Test {

**private** **static** ApplicationContext *context*;

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

*context* = **new** ClassPathXmlApplicationContext("AssignmentQ1.xml");

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Using Construction Injection\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Address Address1 = *context*.getBean("Address1", Address.**class**);

System.***out***.println("Address Class Bean");

System.***out***.println(Address1);

Customer Customer1 = *context*.getBean("Customer1", Customer.**class**);

System.***out***.println("Customer Class Bean ");

System.***out***.println(Customer1);

}

}

1. **.Xml File**

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

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

xmlns=*"http://www.springframework.org/schema/beans"*

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

xmlns:mvc=*"http://www.springframework.org/schema/mvc"*

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

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

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

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

<!-- creating beans using constructor injections-->

<!-- creating bean for address-->

<bean id=*"Address1"* class=*"AssignmentQ1.Address"* autowire=*"byName"*>

<constructor-arg name=*"street"* value=*"Street No. 4"*/>

<constructor-arg name=*"city"* value=*"Nagpur"*/>

<constructor-arg name=*"state"* value=*"Maharashtra"*/>

<constructor-arg name=*"zip"* value=*"440016"*/>

<constructor-arg name=*"country"* value=*"India"*/>

</bean>

<!-- creating bean for customer-->

<bean id=*"Customer1"* class=*"AssignmentQ1.Customer"* autowire=*"byName"*>

<constructor-arg name=*"customerId"* value=*"102"*/>

<constructor-arg name=*"customerName"* value=*"Prajwali Sukhadeve"*/>

<constructor-arg name=*"customerContact"* value=*"8975163401"*/>

<constructor-arg name=*"CustomerAddress"* ref=*"Address1"*/>

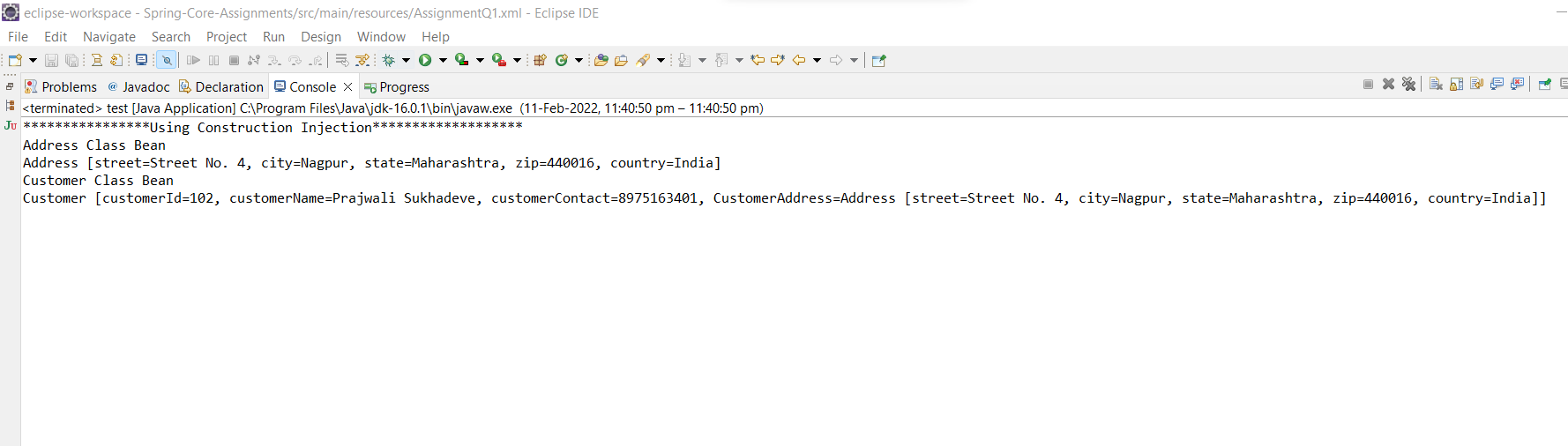
</bean>

<context:annotation-config />

<context:component-scan base-package=*"controller"* />

</beans>

1. **Output**

****

**Junit Testing Code**

**package** AssignmentQ1;

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

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

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

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

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

**class** TestTest {

**private** Customer Customer;

**private** ApplicationContext context;

@BeforeAll

**public** **void** init(){

context = **new** ClassPathXmlApplicationContext("AssignmentQ1.xml");

Customer = context.getBean("Customer",Customer.**class**);

}

@Test

**public** **void** shouldAnswerWithTrue()

{

*assertEquals*(Customer.toString(),"Customer [customerId=101, customerName=Sanket Bollamwar, customerContact=7709496916, CustomerAddress=Address [street=Street No.6, city=Ballarpur, state=Maharashtra, zip=442701, country=India]]");

}

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.

1. **Test Class**

**package** AssignmentQ2;

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

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

**public** **class** Test {

**private** **static** ApplicationContext *context*;

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

*context* = **new** ClassPathXmlApplicationContext("AssignmentQ2.xml");

Question1 question1 = *context*.getBean("question1",Question1.**class**);

System.***out***.println("1.A program where answers is of type List<String> or String []");

System.***out***.println(question1);

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

Question2 question2 = *context*.getBean("question2",Question2.**class**);

System.***out***.println("A program where answers is of type Set<String>");

System.***out***.println(question2);

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

Question3 question3 = *context*.getBean("question3",Question3.**class**);

System.***out***.println("A program where answers is of type Map<Integer, String>");

System.***out***.println(question3);

}

}

1. **Question1 Class**

**package** AssignmentQ2;

**import** java.util.List;

**public** **class** Question1 {

**private** **int** questionId;

**private** String question;

**private** List<String> answers;

**public** Question1() {

**super**();

}

**public** **int** getQuestionId() {

**return** questionId;

}

**public** **void** setQuestionId(**int** questionId) {

**this**.questionId = questionId;

}

**public** String getQuestion() {

**return** question;

}

**public** **void** setQuestion(String question) {

**this**.question = question;

}

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

**return** answers;

}

**public** **void** setAnswers(List<String> answers) {

**this**.answers = answers;

}

@Override

**public** String toString() {

**return** "Question1: [" +

"questionId=" + questionId + '\n' +

", question='" + question + '\n' +

", answers=" + answers + '\n' +

']';

}

}

1. **.Xml file**

<!-- creating bean for class Question1-->

<bean class=*"AssignmentQ2.Question1"* name=*"question1"*>

<property name=*"questionId"* value=*"1"*/>

<property name=*"question"* value=*"What is Spring Framework?"*/>

<property name=*"answers"*>

<list>

<value>Spring is a powerful open-source, loosely coupled, lightweight, java framework meant for reducing the complexity of developing enterprise-level applications.</value>

<value>This framework is also called the “framework of frameworks” as spring provides support to various other important frameworks like JSF, Hibernate, Structs, EJB, etc.</value>

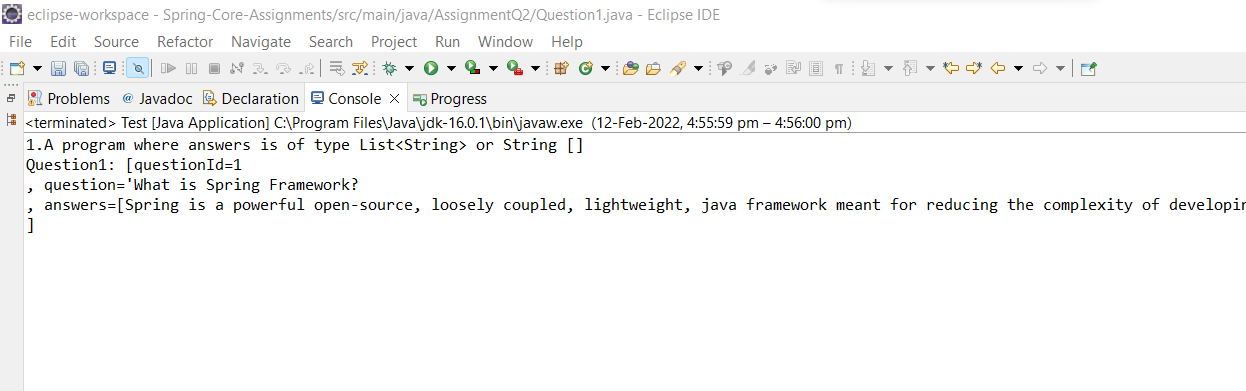
<value>Spring handles all the infrastructure-related aspects which lets the programmer focus mostly on application development.</value>

</list>

</property>

</bean>

1. **Output**

****

1. **Question2 Class**

**package** AssignmentQ2;

**import** java.util.Set;

**public** **class** Question2 {

**private** **int** questionId;

**private** String question;

**private** Set<String> answers;

**public** Question2() {

**super**();

}

**public** **int** getQuestionId() {

**return** questionId;

}

**public** **void** setQuestionId(**int** questionId) {

**this**.questionId = questionId;

}

**public** String getQuestion() {

**return** question;

}

**public** **void** setQuestion(String question) {

**this**.question = question;

}

**public** Set<String> getAnswers() {

**return** answers;

}

**public** **void** setAnswers(Set<String> answers) {

**this**.answers = answers;

}

@Override

**public** String toString() {

**return** "Question2: [" +

"questionId=" + questionId +

", question='" + question + '\'' +

", answers=" + answers +

']';

}

}

1. **.Xml file**

<!-- creating a bean for class Question2-->

<bean class=*"AssignmentQ2.Question2"* name=*"question2"*>

<property name=*"questionId"* value=*"2"*/>

<property name=*"question"* value=*"What are the features of Spring Framework?"*/>

<property name=*"answers"*>

<set>

<value>Spring is a lightweight, java based, loosely coupled framework.</value>

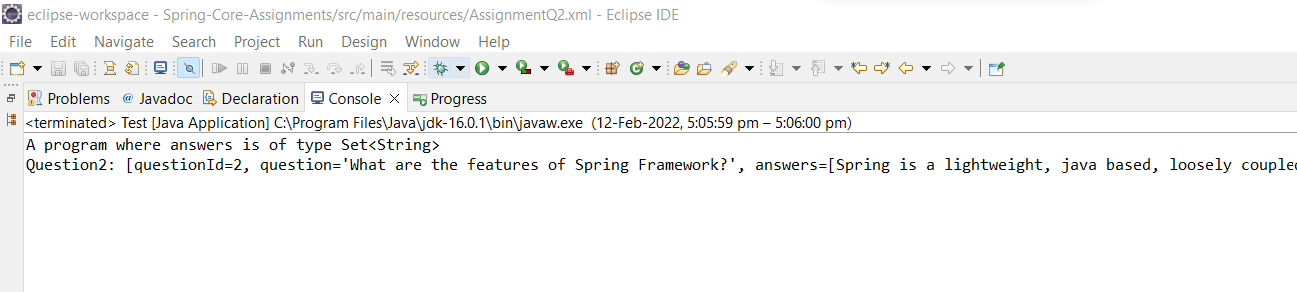
<value>Spring provides generic abstraction layer for transaction management that is also very useful for container-less environments.</value>

</set>

</property>

</bean>

1. **Output**

****

1. **Question3 Class**

**package** AssignmentQ2;

**import** java.util.Map;

**public** **class** Question3 {

**private** **int** questionId;

**private** String questionName;

**private** Map<Integer,String> answers;

**public** Question3() {

**super**();

}

**public** **int** getQuestionId() {

**return** questionId;

}

**public** **void** setQuestionId(**int** questionId) {

**this**.questionId = questionId;

}

**public** String getQuestionName() {

**return** questionName;

}

**public** **void** setQuestionName(String questionName) {

**this**.questionName = questionName;

}

**public** Map<Integer, String> getAnswers() {

**return** answers;

}

**public** **void** setAnswers(Map<Integer, String> answers) {

**this**.answers = answers;

}

@Override

**public** String toString() {

**return** "Question3: [" +

"questionId=" + questionId +'\n' +

", questionName='" + questionName + '\n' +

", answers=" + answers +'\n' +

']';

}

}

1. **.Xml file**

<!-- creating a bean for class Question3-->

<bean class=*"AssignmentQ2.Question3"* name=*"question3"*>

<property name=*"questionId"* value=*"3"*/>

<property name=*"questionName"* value=*"Explain the difference between constructor and setter injection?"*/>

<property name=*"answers"*>

<map>

<entry key=*"1"*>

<value>In constructor injection, partial injection is not allowed whereas it is allowed in setter injection.</value>

</entry>

<entry key=*"2"*>

<value>The constructor injection doesn’t override the setter property whereas the same is not true for setter injection.</value>

</entry>

<entry key=*"3"*>

<value>Constructor injection creates a new instance if any modification is done. The creation of a new instance is not possible in setter injection.</value>

</entry>

<entry key=*"4"*>

<value>In case the bean has many properties, then constructor injection is preferred. If it has few properties, then setter injection is preferred.</value>

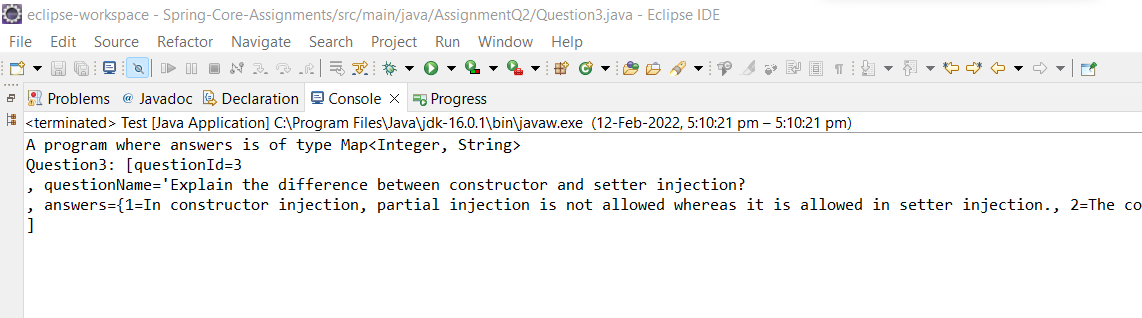
</entry>

</map>

</property>

</bean>

1. **Output**

****

1. Example on autowiring

Design and Develop a Banking Application as follows:

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

**BankAccount Class**

**package** AssignmentQ3;

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

**private** **int** accountId;

**private** String accountHolderName;

**private** String accountType;

**private** **double** accountBalance;

**public** BankAccount() {

**super**();

}

**public** **int** getAccountId() {

**return** accountId;

}

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

**this**.accountId = 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(**double** accountBalance) {

**this**.accountBalance = accountBalance;

}

@Override

**public** String toString() {

**return** "BankAccount: [" + '\n' +

"accountId=" + accountId + '\n' +

", accountHolderName='" + accountHolderName + '\n' +

", accountType='" + accountType + '\n' +

", accountBalance=" + accountBalance + '\n' +

']';

}

**public** BankAccount(**int** accountId, String accountHolderName, String accountType, **double** accountBalance) {

**super**();

**this**.accountId = accountId;

**this**.accountHolderName = accountHolderName;

**this**.accountType = accountType;

**this**.accountBalance = 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.

**BankAccountRepository Interface:**

**package** AssignmentQ3;

**public** **interface** BankAccountRepository {

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

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

}

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

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

**BankAccountRepositoryImpl :**

**package** AssignmentQ3;

**public** **class** BankAccountRepositoryImpl **implements** BankAccountRepository {

**private** BankAccount bankAccount1;

**private** BankAccount bankAccount2;

**public** BankAccountRepositoryImpl() {

}

**public** BankAccountRepositoryImpl(BankAccount bankAccount1, BankAccount bankAccount2) {

**this**.bankAccount1 = bankAccount1;

**this**.bankAccount2 = bankAccount2;

}

@Override

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

**if**(**this**.bankAccount1.getAccountId()==accountId){

**return** bankAccount1.getAccountBalance();

}

**else** {

**return** bankAccount2.getAccountBalance();

}

}

@Override

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

**double** updateBal = 0;

**if**(**this**.bankAccount1.getAccountId()==accountId){

updateBal = newBalance;

bankAccount1.setAccountBalance(updateBal);

}

**else** {

updateBal = newBalance;

bankAccount2.setAccountBalance(updateBal);

}

**return** updateBal;

}

}

* 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)

**BankAccountService Interface:**

**package** AssignmentQ3;

**public** **interface** BankAccountService {

**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.

**BankAccountServiceImpl Class:**

**package** AssignmentQ3;

**public** **class** BankAccountServiceImp **implements** BankAccountService {

**private** BankAccount bankAccount;

@Override

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

**return** 0;

}

@Override

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

**return** 0;

}

@Override

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

**return** 0;

}

@Override

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

**return** **false**;

}

}

* 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)

**BankAccount Controller Class:**

**package** AssignmentQ3;

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

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

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

**public** **class** BankAccountController **implements** ApplicationContextAware {

**private** ApplicationContext context;

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

**double** newBalance = 0;

BankAccountRepositoryImpl accountRepository = (BankAccountRepositoryImpl) context.getBean("BankRepo");

**if**(accountRepository.getBalance(accountId)>=balance){

newBalance = accountRepository.getBalance(accountId)-balance;

accountRepository.updateBalance(accountId,newBalance);

}

**return** newBalance;

}

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

BankAccountRepositoryImpl bankAccountRepository = context.getBean("BankRepo",BankAccountRepositoryImpl.**class**);

**double** newBalance = bankAccountRepository.getBalance(accountId)+balance;

**return** bankAccountRepository.updateBalance(accountId,newBalance);

}

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

BankAccountRepositoryImpl bankAccountRepository = (BankAccountRepositoryImpl) context.getBean("BankRepo");

**return** bankAccountRepository.getBalance(accountId);

}

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

BankAccountRepositoryImpl accountRepository = (BankAccountRepositoryImpl) context.getBean("BankRepo");

**if**(accountRepository.getBalance(fromAccount)>=amont){

**double** updateAmt = amont+accountRepository.getBalance(toAccount);

accountRepository.updateBalance(fromAccount,accountRepository.getBalance(fromAccount)-amont);

accountRepository.updateBalance(toAccount,accountRepository.getBalance(toAccount)+amont);

**return** **true**;

}

**return** **false**;

}

@Override

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

**this**.context = applicationContext;

}

}

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

**Test Class:**

**package** AssignmentQ3;

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

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

**public** **class** Test {

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

ApplicationContext context = **new** ClassPathXmlApplicationContext("Assignment3.xml");

BankAccountController controller = (BankAccountController) context.getBean("controller");

System.***out***.println(controller.getBalance(5498));

System.***out***.println(controller.deposit(5498,5000));

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

System.***out***.println(controller.withdraw(5498,10000));

System.***out***.println(controller.getBalance(5498));

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

System.***out***.println(controller.fundTransfer(5498,5496,10000));

System.***out***.println(controller.getBalance(5498));

System.***out***.println(controller.getBalance(5496));

}

}

* 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.

**Using byName:**

<bean class=*"org.example.AssignmentQ3.BankAccountRepositoryImpl"* id=*"BankRepo"* autowire=*"byName"*/>

**Using byType:**

<bean class=*"org.example.AssignmentQ3.BankAccountRepositoryImpl"* id=*"BankRepo"* autowire=*"byType"*/>

**Using constructor:**

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

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

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

xmlns=*"http://www.springframework.org/schema/beans"*

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

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*

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

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

<bean class=*"AssignmentQ3.BankAccount"* id=*"bankAccount1"*>

<constructor-arg type=*"int"* value=*"5496"*/>

<constructor-arg value=*"Sanket Bolamwar"*/>

<constructor-arg value=*"Saving"*/>

<constructor-arg type=*"double"* value=*"50000"*/>

</bean>

<bean class=*"AssignmentQ3.BankAccount"* id=*"bankAccount2"*>

<constructor-arg type=*"int"* value=*"5498"*/>

<constructor-arg value=*"Prajwali Sukhadeve"*/>

<constructor-arg value=*"Current"*/>

<constructor-arg type=*"double"* value=*"60000"*/>

</bean>

<bean class=*"AssignmentQ3.BankAccountRepositoryImpl"* id=*"BankRepo"* autowire=*"constructor"*/>

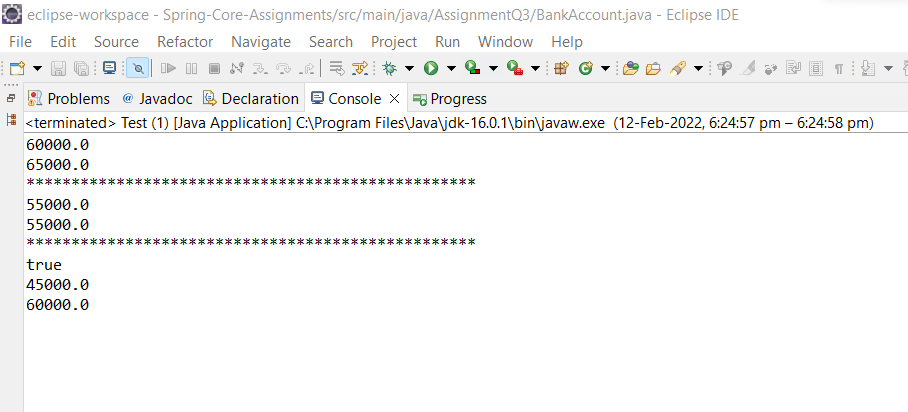
<bean class=*"AssignmentQ3.BankAccountController"* id=*"controller"*/>

<context:annotation-config />

<context:component-scan base-package=*"controller"* />

</beans>

**Output:**

****

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

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

**BankAccount.java**

**package** AssignmentQ4;

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

**private** **int** accountId;

**private** String accountHolderName;

**private** String accountType;

**private** **double** accountBalance;

**public** BankAccount() {

**super**();

}

**public** BankAccount(**int** accountId, String accountHolderName, String accountType, **double** accountBalance) {

**this**.accountId = accountId;

**this**.accountHolderName = accountHolderName;

**this**.accountType = accountType;

**this**.accountBalance = accountBalance;

}

**public** **int** getAccountId() {

**return** accountId;

}

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

**this**.accountId = 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(**double** accountBalance) {

**this**.accountBalance = accountBalance;

}

@Override

**public** String toString() {

**return** "BankAccount: [" + '\n' +

"accountId=" + accountId + '\n' +

", accountHolderName='" + accountHolderName + '\n' +

", accountType='" + accountType + '\n' +

", accountBalance=" + accountBalance + '\n' +

']';

}

}

**BankAccountRepository Interface:**

**package** AssignmentQ4;

**public** **interface** BankAccountRepository {

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

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

}

**BankAccountRepositoryImpl Class:**

package AssignmentQ4;

import AssignmentQ3.BankAccount;

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

import org.springframework.stereotype.Repository;

@Repository("BankRepo")

public class BankAccountRepositoryImpl implements BankAccountRepository {

@Autowired

private BankAccount bankAccount1;

@Autowired

private BankAccount bankAccount2;

public BankAccountRepositoryImpl() {

}

public BankAccountRepositoryImpl(BankAccount bankAccount1, BankAccount bankAccount2) {

this.bankAccount1 = bankAccount1;

this.bankAccount2 = bankAccount2;

}

@Override

public double getBalance(long accountId) {

if(this.bankAccount1.getAccountId()==accountId){

return bankAccount1.getAccountBalance();

}

else {

return bankAccount2.getAccountBalance();

}

}

@Override

public double updateBalance(long accountId, double newBalance) {

double updateBal = 0;

if(this.bankAccount1.getAccountId()==accountId){

updateBal = newBalance;

bankAccount1.setAccountBalance(updateBal);

}

else {

updateBal = newBalance;

bankAccount2.setAccountBalance(updateBal);

}

return updateBal;

}

}

**Interface BankAccountService:**

**package** AssignmentQ4;

**public** **interface** BankAccountService {

**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);

}

**BankAccountServiceImp Class:**

**package** AssignmentQ4;

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

**public** **class** BankAccountServiceImp **implements** BankAccountService {

@Autowired

**private** BankAccount bankAccount1;

@Autowired

**private** BankAccount bankAccount2;

**public** BankAccountServiceImp(){

}

**public** BankAccountServiceImp(BankAccount bankAccount1, BankAccount bankAccount2) {

**this**.bankAccount1 = bankAccount1;

**this**.bankAccount2 = bankAccount2;

}

@Override

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

**return** 0;

}

@Override

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

**return** 0;

}

@Override

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

**if**(**this**.bankAccount1.getAccountId()==accountId){

**return** bankAccount1.getAccountBalance();

}

**else** {

**return** bankAccount2.getAccountBalance();

}

}

@Override

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

**return** **false**;

}

}

**BankAccountController Class:**

package AssignmentQ4;

import org.springframework.beans.BeansException;

import org.springframework.context.ApplicationContext;

import org.springframework.context.ApplicationContextAware;

public class BankAccountController implements BankAccountService, ApplicationContextAware {

private ApplicationContext context;

@Override

public double withdraw(long accountId, double balance){

double newBalance = 0;

BankAccountRepositoryImpl accountRepository = (BankAccountRepositoryImpl) context.getBean("BankRepo");

if(accountRepository.getBalance(accountId)>=balance){

newBalance = accountRepository.getBalance(accountId)-balance;

accountRepository.updateBalance(accountId,newBalance);

}

return newBalance;

}

@Override

public double deposit(long accountId, double balance){

BankAccountRepositoryImpl bankAccountRepository = context.getBean("BankRepo", BankAccountRepositoryImpl.class);

double newBalance = bankAccountRepository.getBalance(accountId)+balance;

return bankAccountRepository.updateBalance(accountId,newBalance);

}

@Override

public double getBalance(long accountId){

BankAccountRepositoryImpl bankAccountRepository = (BankAccountRepositoryImpl) context.getBean("BankRepo");

return bankAccountRepository.getBalance(accountId);

}

@Override

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

BankAccountRepositoryImpl accountRepository = (BankAccountRepositoryImpl) context.getBean("BankRepo");

if(accountRepository.getBalance(fromAccount)>=amont){

double updateAmt = amont+accountRepository.getBalance(toAccount);

accountRepository.updateBalance(fromAccount,accountRepository.getBalance(fromAccount)-amont);

accountRepository.updateBalance(toAccount,accountRepository.getBalance(toAccount)+amont);

return true;

}

return false;

}

@Override

public void setApplicationContext(ApplicationContext applicationContext) throws BeansException {

this.context = applicationContext;

}

}

**Config.java**

**package** AssignmentQ4;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.ComponentScan;

**import** org.springframework.context.annotation.Configuration;

@Configuration

@ComponentScan(basePackages = "AssignmentQ4")

**public** **class** Config {

@Bean

**public** BankAccount getBankAccount(){

BankAccount bankAccount1 = **new** BankAccount(5498,"Sanket Bolamwar","Saving",50000);

**return** bankAccount1;

}

@Bean

**public** BankAccount getBankAccount1(){

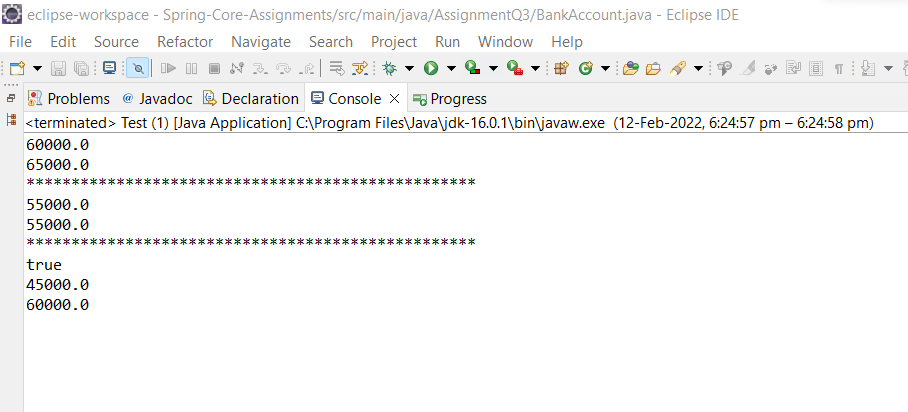
BankAccount bankAccount1 = **new** BankAccount(5496,"Prajwali Sukhadeve","Current",60000);

**return** bankAccount1;

}

}

**Output:**

****

1. Write a program to demonstrate use of @Resource, @Inject, @Required annotations
2. 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).
3. Write a Java program to demonstrate SPEL (Spring Expression language)

**Test Class:**

**package** AssignmentQ7;

**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 expressionParser = **new** SpelExpressionParser();

Expression expression = expressionParser.parseExpression("'Hello i am Sanket Bolamwar'");

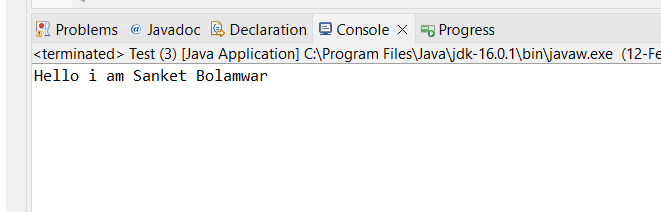
String message = (String) expression.getValue();

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

}

}

**Output:**

****

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)

**AnnotationBeanLifeCycle Class:**

**package** AssignmentQ8;

**import** javax.annotation.PostConstruct;

**import** javax.annotation.PreDestroy;

**public** **class** AnnotationBeanLifeCycle {

**int** price;

**public** AnnotationBeanLifeCycle() {

**super**();

}

@Override

**public** String toString() {

**return** "AnnotationBeanLifeCycle Price: {" +

"price=" + price +

'}';

}

**public** **int** getPrice() {

**return** price;

}

**public** **void** setPrice(**int** price) {

**this**.price = price;

}

@PostConstruct

**public** **void** init(){

System.***out***.println("Calling init using Annotation");

}

@PreDestroy

**public** **void** destroy(){

System.***out***.println("Calling destroy from Annotation");

}

}

**InterfaceBeanlifeCycle Class:**

package AssignmentQ8;

import org.springframework.beans.factory.DisposableBean;

import org.springframework.beans.factory.InitializingBean;

public class InterfaceBeanLifeCycle implements DisposableBean, InitializingBean {

private double price;

public InterfaceBeanLifeCycle() {

super();

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

@Override

public String toString() {

return "Interface Price: {" +

"price=" + price +

'}';

}

public void init() throws Exception {

System.out.println("Calling init method using interface");

}

@Override

public void destroy() throws Exception {

System.out.println("Calling destroy method using interface");

}

@Override

public void afterPropertiesSet() throws Exception {

// TODO Auto-generated method stub

}

}

**XmlBeanLifeCycle Class:**

**package** AssignmentQ8;

**public** **class** XmlBeanLifeCycle {

**private** **double** price;

**public** XmlBeanLifeCycle() {

**super**();

}

**public** **double** getPrice() {

**return** price;

}

**public** **void** setPrice(**double** price) {

System.***out***.println("Setting the value of data members");

**this**.price = price;

}

**public** **void** init(){

System.***out***.println("Calling the initialization method of bean");

}

**public** **void** destroy(){

System.***out***.println("Calling the destroy method of bean");

}

@Override

**public** String toString() {

**return** "ORDER PRICE: {" +

"price=" + price +

'}';

}

}

**App Class:**

**package** AssignmentQ8;

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

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

**public** **class** App {

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

//This is the code segment for implementing bean lifecycle for XML.

AbstractApplicationContext context = **new** ClassPathXmlApplicationContext("AssignmentQ8.xml");

XmlBeanLifeCycle obj = (XmlBeanLifeCycle) context.getBean("XML1");

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

context.registerShutdownHook();

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

//This is the code segment for implementing bean lifecycle for interfaces.

InterfaceBeanLifeCycle obj1 = (InterfaceBeanLifeCycle) context.getBean("InterfaceBeanLifeCycle");

System.***out***.println(obj1);

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

//This is the code segment for implementing bean lifecycle for Annotations.

AnnotationBeanLifeCycle obj2 = (AnnotationBeanLifeCycle)context.getBean("AnnotationBeanLifeCycle");

System.***out***.println(obj2);

}

}

**.Xml file:**

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

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

xmlns=*"http://www.springframework.org/schema/beans"*

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/>

<bean class=*"AssignmentQ8.XmlBeanLifeCycle"* name=*"XML1"* init-method=*"init"* destroy-method=*"destroy"*>

<property name=*"price"* value=*"100000"*/>

</bean>

<bean class=*"AssignmentQ8.InterfaceBeanLifeCycle"* name=*"InterfaceBeanLifeCycle"* init-method=*"init"* destroy-method=*"destroy"*>

<property name=*"price"* value=*"100000"*/>

</bean>

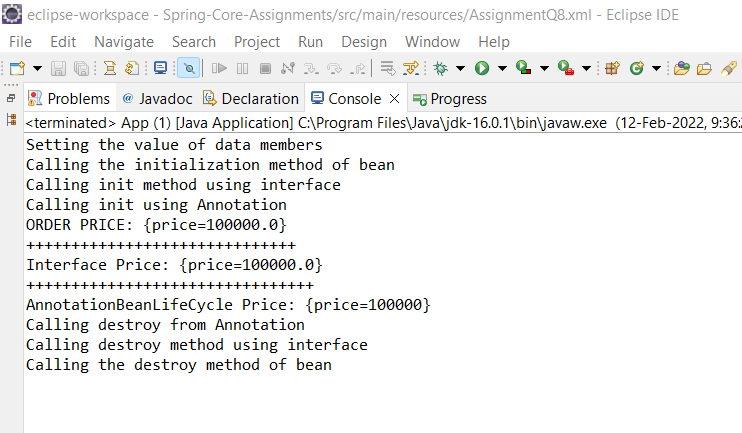
<bean class=*"AssignmentQ8.AnnotationBeanLifeCycle"* name=*"AnnotationBeanLifeCycle"* init-method=*"init"* destroy-method=*"destroy"*>

<property name=*"price"* value=*"100000"*/>

</bean>

</beans>

**Output:**

****

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

**Ans = follow Question no 8 XmlBeanLifeCycle**

1. Write a java program to demonstrate ApplicationContextAware interface.

**Test Class**

**package** AssignmentQ10;

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

**import** org.springframework.beans.factory.BeanNameAware;

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

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

**public** **class** Test **implements** ApplicationContextAware, BeanNameAware {

**private** String place;

Demo demo;

**public** String getPlace() {

**return** place;

}

**public** **void** setPlace(String place) {

**this**.place = place;

}

**public** Demo getDemo() {

**return** demo;

}

**public** **void** setDemo(Demo demo) {

**this**.demo = demo;

}

@Override

**public** String toString() {

**return** "Example: [" +

"place='" + place + '\'' +

", demo=" + demo +

']';

}

@Override

**public** **void** setBeanName(String name) {

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

}

@Override

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

System.***out***.println("Application name: "+ applicationContext);

}

}

**Demo Class**

**package** AssignmentQ10;

**public** **class** Demo {

**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;

}

}

**Main Class**

**package** AssignmentQ10;

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

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

**public** **class** Main {

**private** **static** ApplicationContext *context*;

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

*context* = **new** ClassPathXmlApplicationContext("AssignmentQ10.xml");

Test example1 = *context*.getBean("example1", Test.**class**);

System.***out***.println(example1);

}

}

**.XML File:**

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

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

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

xmlns=*"http://www.springframework.org/schema/beans"*

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/>

<bean class=*"AssignmentQ10.Demo"* id=*"demo1"*>

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

<property name=*"y"* value=*"40"*/>

</bean>

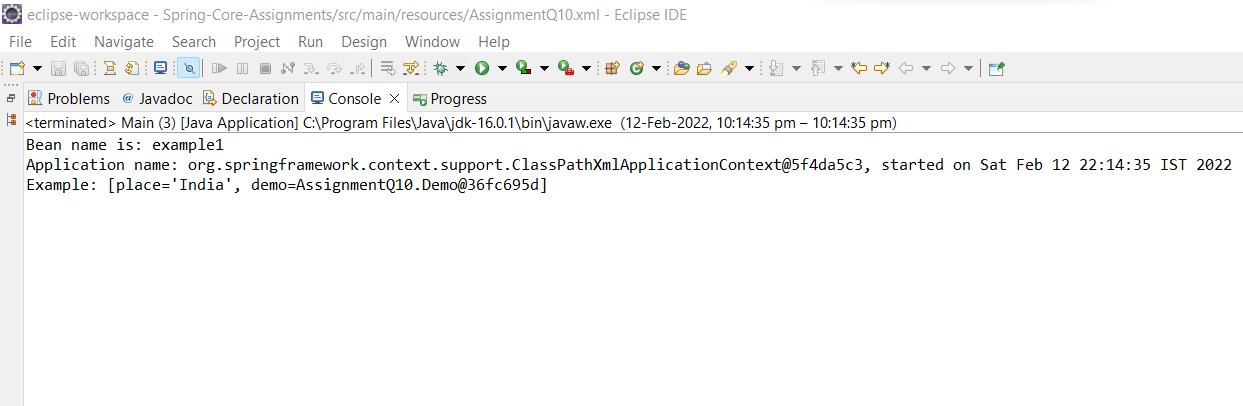
<bean class=*"AssignmentQ10.Test"* name=*"example1"* autowire=*"byType"*>

<property name=*"place"* value=*"Korba"*/>

</bean>

</beans>

**Output:**

****