**PRACTICAL NO. 03**

**Introduction to Spring Framework**

|  |  |
| --- | --- |
| LOB3 | Demonstrate Data Access with Spring framework. |
| LO3 | Develop application using Spring Framework, Lightweight Containers and Dependency Injection. |

Spring is the most popular lightweight and open-source application development framework for enterprise Java. Spring Framework -

* enables Plain Old Java Object (POJO) based programming model
* with POJO you don’t need EJB container product
* utilizes existing technologies like JEE, ORM framework etc.

The Spring Framework consists of features organized into about 20 modules. These modules are grouped into Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation, and Test.

|  |  |
| --- | --- |
|  | Spring Architecture  Referred from –  https://docs.spring.io/spring-framework/docs/3.0.x/spring-framework-reference/html/overview.html |

**Modules of Spring Framework –**

1. **Core Container –**
   1. The Core Container consists of the Core, Beans, Context, and Expression Language modules.
   2. The **Core and Beans modules** provide the IoC and Dependency Injection features.
   3. The **Context module** is a means to access objects in a framework-style manner that is like a JNDI registry.
   4. The **Expression Language module** provides a powerful expression language for querying and manipulating an object graph at runtime. It supports setting and getting property values, property assignment, method invocation, accessing the context of arrays, collections, and indexers etc.
2. **Data Access/Integration –**
   1. The Data Access/Integration layer consists of the JDBC, ORM, OXM, JMS and Transaction modules.
   2. The **JDBC module** provides a JDBC-abstraction layer that removes the need to do tedious JDBC coding and parsing of database-vendor specific error codes.
   3. The **ORM module** provides APIs, including JPA, JDO, Hibernate, and iBatis.
   4. The **OXM module** provides an abstraction layer that supports Object/XML mapping implementations for JAXB, Castor, XMLBeans, JiBX and XStream.
   5. The **Java Messaging Service (JMS) module** contains features for producing and consuming messages.
3. **Web –**
   1. The Web layer consists of the Web, Web-Servlet, Web-Struts, and Web-Portlet modules.
   2. Spring's Web module provides basic web-oriented integration features such as multipart file-upload functionality and the initialization of the IoC container and a web-oriented application context.
4. **AOP and Instrumentation –** 
   1. Spring's AOP module provides an aspect-oriented programming implementation allowing you to define, for example, method-interceptors and pointcuts to cleanly decouple code that implements functionality that should be separated.
   2. The separate Aspects module provides integration with AspectJ.
   3. The Instrumentation module provides class instrumentation support and classloader implementations to be used in certain application servers.
5. **Test –**
   1. The Test module supports the testing of Spring components with JUnit or TestNG.
   2. It provides consistent loading of Spring ApplicationContexts and caching of those contexts.
   3. It also provides mock objects that you can use to test your code in isolation.

**What are Beans?**

* In Spring, plain old java objects (POJO) are called “beans” and those objects instantiated, managed, created by Spring IoC container.
* Beans are created with the help of configuration metadata (XML file) which is supplied to the container.
* Bean definition contains configuration metadata. With this information container will create bean, manage beans life cycle and beans dependencies.

**Inversion of Control (IoC)**

* Inversion of control (IoC) is the principle where the control flow of a program is inverted: instead, the programmer controls the flow of a program, the external sources (framework, services, other components) take control of it.
* Inversion means reversal of the natural order. In Java, the natural order is to declare variables and then initialize those variables.

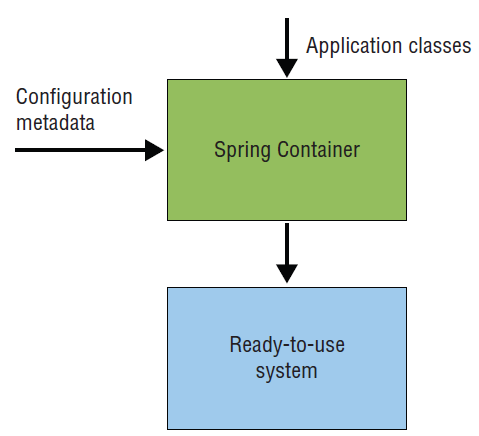
Example: Question q1 = new Question();

* In IoC, the developer could not use the new keyword and call the constructor of the Question class. Instead, the IoC container will do the job.

Example: Question q1 = ApplicationContext.getInstance(Question.class);

* Dependent components are never instantiated using a new operator within component classes. They are injected into the component by the container instance at run time. Hence, control of dependencies is moved out of components to the container.
* IoC has basically two forms:
* dependency lookup and
* dependency injection

**Configuration MetaData**

* Spring configuration metadata needs to be created to tell Spring container how to initiate, configure, wire and assemble the application specific objects.
* The Spring Container expects information from you to instantiate beans and to specify how to wire them together.
* This information is called **configuration metadata.**
* Together with this configuration metadata, the Spring Container takes classes written in the application and then creates and assembles beans in it.
* Spring has provided three ways of configurations:
* XML-based Configuration
* Annotation-based configuration
* Java-based configuration (JavaConfig)
* The **Spring Container is also a Java object**, which is created in the application at some specific point and then allowed to manage the rest of the application.

**Q1. Write a program to demonstrate dependency injection via setter method for**

**Student class.**

**Programs :-**

**Address.java**

**package** packP3Code01;

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

**private** String city;

**private** String state;

// Getter and Setter for City

**public** String getCity() {

**return** city;

}

**public** **void** setCity(String city) {

**this**.city = city;

}

// Getter and Setter for State

**public** String getState() {

**return** state;

}

**public** **void** setState(String state) {

**this**.state = state;

}

@Override

**public** String toString() {

**return** city + ", " + state;

}

}

**Student.java**

**package** packP3Code01;

**public** **class** Student {

**private** **int** id;

**private** String name;

**private** Address address;

// Getter and Setter for ID

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

// Getter and Setter for Name

**public** String getName() {

**return** name;

}

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

**this**.name = name;

}

// Getter and Setter for Address (Dependency)

**public** Address getAddress() {

**return** address;

}

**public** **void** setAddress(Address address) {

**this**.address = address;

}

// Display Student Details

**public** **void** displayInfo() {

System.***out***.println("Student ID: " + id);

System.***out***.println("Student Name: " + name);

System.***out***.println("Address: " + address);

}

}

**Address.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-3.0.xsd*

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

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

<!-- Bean for Address -->

<bean id=*"addressBean"* class=*"packP3Code01.Address"*>

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

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

</bean>

<!-- Bean for Student -->

<bean id=*"studentBean"* class=*"packP3Code01.Student"*>

<property name=*"id"* value=*"101"* />

<property name=*"name"* value=*"Manoj Kumar"* />

<property name=*"address"* ref=*"addressBean"* />

</bean>

</beans>

**P3Code1Main.java**

**package packP3Code01;**

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

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

**public class P3Code1Main {**

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

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

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

**// Get Student bean**

**Student student = (Student) context.getBean("studentBean");**

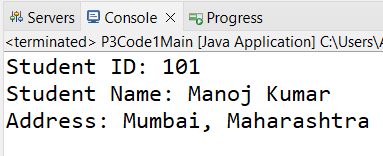
**// Display student information**

**student.displayInfo();**

**}**

**}**

**Output :-**

****

**Q2 Create a Spring application to Autowire a map of beans. The map consists of**

**courseCode (like AJP01, MFCS, WT03 etc.) and list of students enrolled to that**

**courseCode.**

**Programs :-**

**Student.java**

**package** packP3Code02;

**public** **class** Student {

**private** String name;

**private** **int** id;

// Constructor

**public** Student(**int** id, String name) {

**this**.id = id;

**this**.name = name;

}

// Getters

**public** String getName() {

**return** name;

}

**public** **int** getId() {

**return** id;

}

@Override

**public** String toString() {

**return** "Student: id=" + id + " name='" + name + "'\n";

}

}

**CourseService.java**

package packP3Code02;

import java.util.List;

import java.util.Map;

public class CourseService {

private Map<String, List<Student>> courseEnrollments;

// Setter for autowiring

public void setCourseEnrollments(Map<String, List<Student>> courseEnrollments) {

this.courseEnrollments = courseEnrollments;

}

// Display the enrolled students for each course

public void displayEnrollments() {

for (Map.Entry<String, List<Student>> entry : courseEnrollments.entrySet()) {

System.out.println("Course Code: " + entry.getKey());

System.out.println("Enrolled Students: ");

for (Student student : entry.getValue()) {

System.out.println(" - " + student);

}

}

}

}

**course\_service.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-3.0.xsd*

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

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

<!-- Bean Definitions for Students -->

<bean id=*"student1"* class=*"packP3Code02.Student"*>

<constructor-arg value=*"101"* />

<constructor-arg value=*"Tejas Bhide"* />

</bean>

<bean id=*"student2"* class=*"packP3Code02.Student"*>

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

<constructor-arg value=*"Nikhil Choughule"* />

</bean>

<bean id=*"student3"* class=*"packP3Code02.Student"*>

<constructor-arg value=*"103"* />

<constructor-arg value=*"Aryan Dhuri"* />

</bean>

<!-- Bean Definition for CourseService -->

<bean id=*"courseService"* class=*"packP3Code02.CourseService"*>

<property name=*"courseEnrollments"*>

<map>

<entry key=*"AJP01"*>

<list>

<ref bean=*"student1"* />

<ref bean=*"student2"* />

</list>

</entry>

<entry key=*"WT03"*>

<list>

<ref bean=*"student3"* />

</list>

</entry>

<entry key=*"MFCS"*>

<list>

<ref bean=*"student1"* />

<ref bean=*"student2"* />

<ref bean=*"student3"* />

</list>

</entry>

</map>

</property>

</bean>

</beans>

**P3Code2Main.java**

**package** packP3Code02;

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

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

**public** **class** P3Code2Main {

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

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

// Load the Spring configuration

ApplicationContext context = **new** ClassPathXmlApplicationContext("course\_service.xml");

// Retrieve the CourseService bean

CourseService courseService = (CourseService) context.getBean("courseService");

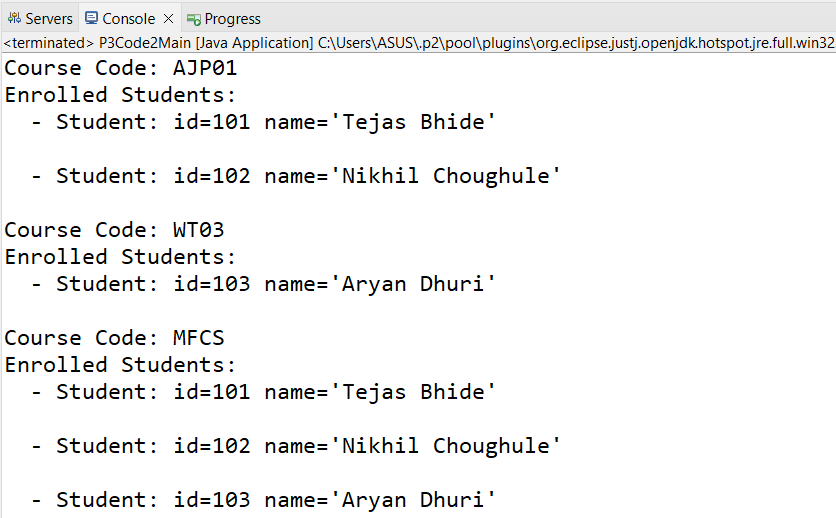
// Display the course enrollments

courseService.displayEnrollments();

}

}

**Output :-**

****

**Q** **3 The library management system manages library books using TreeMap for**

**sorted book IDs. Create a Spring application for injecting TreeMap of ISBN and**

**Book Object.**

**Programs :-**

**Book.java**

**package** packP3Code03;

**public** **class** Book {

**private** String title;

**private** String author;

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

// Constructor

**public** Book(String title, String author, **double** price) {

**this**.title = title;

**this**.author = author;

**this**.price = price;

}

// Getters

**public** String getTitle() {

**return** title;

}

**public** String getAuthor() {

**return** author;

}

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

**return** price;

}

@Override

**public** String toString() {

**return** "Book: title='" + title + "', author='" + author + "', price=" + price +"\n";

}

}

**LibraryService.java**

**package packP3Code03;**

**import java.util.Map;**

**import java.util.TreeMap;**

**public class LibraryService {**

**private TreeMap<String, Book> libraryBooks;**

**// Setter for dependency injection**

**public void setLibraryBooks(TreeMap<String, Book> libraryBooks) {**

**this.libraryBooks = libraryBooks;**

**}**

**// Display all books in the library**

**public void displayBooks() {**

**System.out.println("Books in the Library (Sorted by ISBN):");**

**for (Map.Entry<String, Book> entry : libraryBooks.entrySet()) {**

**System.out.println("ISBN: " + entry.getKey() + " - " + entry.getValue());**

**}**

**}**

**}**

**library\_service.xml**

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

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

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

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

*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"*>

<!-- Bean Definitions for Books -->

<bean id=*"book1"* class=*"packP3Code03.Book"*>

<constructor-arg value=*"Effective Java"* />

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

<constructor-arg value=*"45.00"* />

</bean>

<bean id=*"book2"* class=*"packP3Code03.Book"*>

<constructor-arg value=*"Clean Code"* />

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

<constructor-arg value=*"50.00"* />

</bean>

<bean id=*"book3"* class=*"packP3Code03.Book"*>

<constructor-arg value=*"Head First Java"* />

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

<constructor-arg value=*"40.00"* />

</bean>

<!-- Bean Definition for LibraryService -->

<bean id=*"libraryService"* class=*"packP3Code03.LibraryService"*>

<property name=*"libraryBooks"*>

<map key-type=*"java.lang.String"* value-type=*"packP3Code03.Book"*>

<entry key=*"9780134685991"* value-ref=*"book1"* />

<entry key=*"9780132350884"* value-ref=*"book2"* />

<entry key=*"9780596009205"* value-ref=*"book3"* />

</map>

</property>

</bean>

</beans>

**P3Code3Main.java**

**package packP3Code03;**

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

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

**public class P3Code3Main {**

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

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

**// Load the Spring configuration**

**ApplicationContext context = new ClassPathXmlApplicationContext("library\_service.xml");**

**// Retrieve the LibraryService bean**

**LibraryService libraryService = (LibraryService) context.getBean("libraryService");**

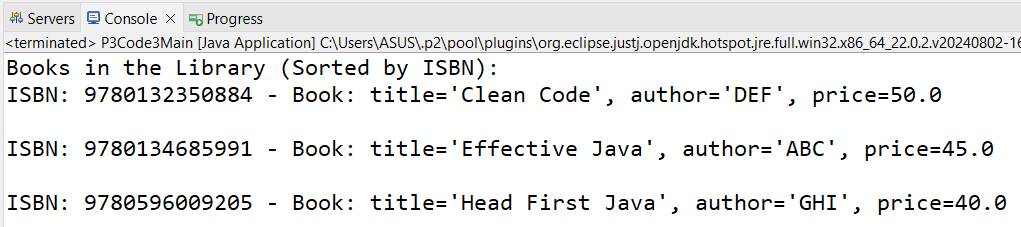
**// Display the books in the library**

**libraryService.displayBooks();**

**}**

**}**

**Output :-**

****

**Q** **4. Write a program to demonstrate List dependency injection via Constructor for a Car Class.**

**Programs :-**

**Engine.java**

**package** CinstructorInjPack;

**public** **class** Engine {

**int** num;

String nm;

**public** Engine(**int** num, String nm) {

**super**();

**this**.num = num;

**this**.nm = nm;

}

**public** **int** getNum() {

**return** num;

}

**public** **void** setNum(**int** num) {

**this**.num = num;

}

**public** String getNm() {

**return** nm;

}

**public** **void** setNm(String nm) {

**this**.nm = nm;

}

@Override

**public** String toString() {

**return** "Engine [num=" + num + ", nm=" + nm + "]";

}

}

**Car.java**

**package** CinstructorInjPack;

**public** **class** Car {

**int** id;

String carnm;

**double** price;

Engine engObj;

**public** Car() {

**super**();

}

**public** Car(**int** id, String carnm, **double** price, Engine engObj) {

**super**();

**this**.id = id;

**this**.carnm = carnm;

**this**.price = price;

**this**.engObj = engObj;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getCarnm() {

**return** carnm;

}

**public** **void** setCarnm(String carnm) {

**this**.carnm = carnm;

}

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

**return** price;

}

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

**this**.price = price;

}

**public** Engine getEngObj() {

**return** engObj;

}

**public** **void** setEngObj(Engine engObj) {

**this**.engObj = engObj;

}

@Override

**public** String toString() {

**return** "Car [id=" + id + ", carnm=" + carnm + ", price=" + price + ", engObj=" + engObj + "]";

}

}

**Car.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-3.0.xsd*

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

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

<bean id=*"carBean"* class=*"CinstructorInjPack.Car"*>

<constructor-arg index=*"0"* value=*"123"*></constructor-arg>

<constructor-arg index=*"1"* value=*"Glanza"*></constructor-arg>

<constructor-arg index=*"2"* value=*"1000000.55"*></constructor-arg>

<constructor-arg>

<ref bean=*"enginBean"*/>

</constructor-arg>

</bean>

<bean id=*"enginBean"* class=*"CinstructorInjPack.Engine"*>

<constructor-arg index=*"0"* value=*"1"*></constructor-arg>

<constructor-arg index=*"1"* value=*"suzuki"*></constructor-arg>

</bean>

<bean id=*"enginBeanSet"* class=*"CinstructorInjPack.Car"*>

<property name=*"id"* value=*"456"*></property>

<property name=*"carnm"* value=*"Baleno"*></property>

<property name=*"price"* value=*"1500000.60"*></property>

<property name=*"engObj"* ref=*"enginBean"*></property>

</bean>

</beans>

**CarDisplay.java**

**package CinstructorInjPack;**

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

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

**public class CarDisplay {**

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

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

**ClassPathXmlApplicationContext ctx=new ClassPathXmlApplicationContext("car.xml");**

**Car cobj=(Car)ctx.getBean("carBean");**

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

**Engine engObj=(Engine)ctx.getBean("enginBean");**

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

**Engine engObj1=(Engine)ctx.getBean("enginBean");**

**engObj1.setNum(2);**

**engObj1.setNm("maruti");**

**System.out.println(engObj1);**

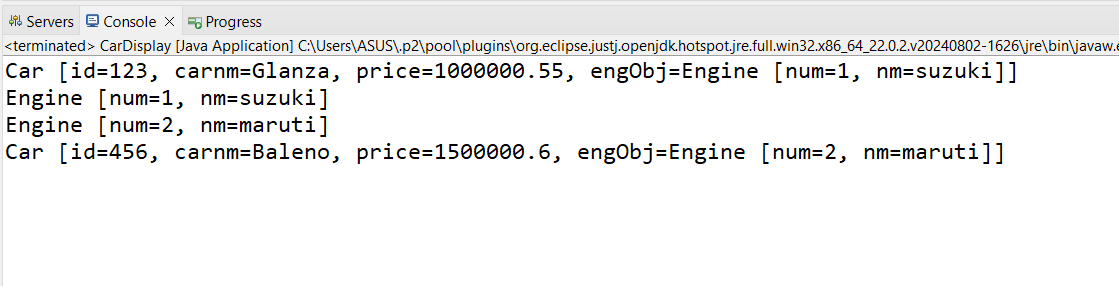
**Car carObj=(Car)ctx.getBean("enginBeanSet");**

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

**}**

**}**

**Output :-**

****

**Q5.** **Create a Spring application to inject dependencies for a simple calculator class.**

**Programs :-**

**CalculatorService.java**

**package** packP3Code05;

**public** **interface** CalculatorService {

**double** add(**double** a, **double** b);

**double** subtract(**double** a, **double** b);

**double** multiply(**double** a, **double** b);

**double** divide(**double** a, **double** b);

}

**CalculatorServiceImpl.java**

**package** packP3Code05;

**public** **class** CalculatorServiceImpl **implements** CalculatorService {

@Override

**public** **double** add(**double** a, **double** b) {

**return** a + b;

}

@Override

**public** **double** subtract(**double** a, **double** b) {

**return** a - b;

}

@Override

**public** **double** multiply(**double** a, **double** b) {

**return** a \* b;

}

@Override

**public** **double** divide(**double** a, **double** b) {

**if** (b == 0) {

**throw** **new** ArithmeticException("Cannot divide by zero");

}

**return** a / b;

}

}

**Calculator.java**

**package** packP3Code05;

**public** **class** Calculator {

**private** CalculatorService calculatorService;

// Setter for dependency injection

**public** **void** setCalculatorService(CalculatorService calculatorService) {

**this**.calculatorService = calculatorService;

}

**public** **void** performOperations() {

**double** a = 10;

**double** b = 5;

System.***out***.println("Addition: " + calculatorService.add(a, b));

System.***out***.println("Subtraction: " + calculatorService.subtract(a, b));

System.***out***.println("Multiplication: " + calculatorService.multiply(a, b));

System.***out***.println("Division: " + calculatorService.divide(a, b));

}

}

**calculator\_service.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-3.0.xsd*

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

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

<!-- Bean for CalculatorService Implementation -->

<bean id=*"calculatorService"* class=*"packP3Code05.CalculatorServiceImpl"* />

<!-- Bean for Calculator -->

<bean id=*"calculator"* class=*"packP3Code05.Calculator"*>

<property name=*"calculatorService"* ref=*"calculatorService"* />

</bean>

</beans>

**P3Code5Main.java**

**package packP3Code05;**

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

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

**public class P3Code5Main {**

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

**// Load the Spring configuration**

**ApplicationContext context = new ClassPathXmlApplicationContext("calculator\_service.xml");**

**// Retrieve the Calculator bean**

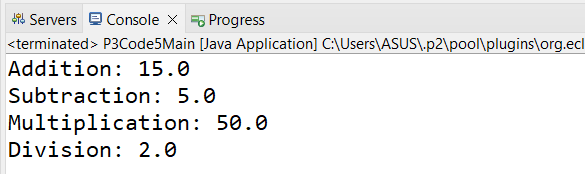
**Calculator calculator = (Calculator) context.getBean("calculator");**

**// Perform operations**

**calculator.performOperations();**

**}**

**}**

**Output :- **