**Q1. Invoke context events start(), stop() and close().**

**Solution**

**package demo;**

**import org.springframework.context.ConfigurableApplicationContext;**

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

**public class Main {**

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

**ConfigurableApplicationContext applicationContext=new ClassPathXmlApplicationContext("spring-config.xml");**

***//Question1***

**applicationContext.start();**

**applicationContext.stop();**

**applicationContext.close();**

**}**

**}**

**Q2. Create listeners for spring events.**

**Solution**

**SpringEventsListener.java**

**package demo.listeners;**

**import org.springframework.context.ApplicationEvent;**

**import org.springframework.context.ApplicationListener;**

***//Question2---->Listener for all spring events***

**public class SpringEventsListener implements ApplicationListener<ApplicationEvent>{**

**@Override**

**public void onApplicationEvent(ApplicationEvent event) {**

**System.*out*.println("----------Event triggerred--------"+event.getClass());**

**}**

**}**

**SpringMultiEventListener.java**

**package demo.listeners;**

**import org.springframework.context.event.\*;**

***//Question2-->In case we want to do separate things on each event in single listener***

**public class SpringMultiEventListener {**

**@EventListener(ContextRefreshedEvent.class)**

**public void referesh(){**

**System.*out*.println("Listening to refresh event");**

**}**

**@EventListener(ContextStartedEvent.class)**

**public void start()**

**{**

**System.*out*.println("Listening to start event");**

**}**

**@EventListener(ContextStoppedEvent.class)**

**public void stop()**

**{**

**System.*out*.println("Listening to stop event");**

**}**

**@EventListener(ContextClosedEvent.class)**

**public void close()**

**{**

**System.*out*.println("Listening to close event");**

**}**

**}**

**Main.java**

**package demo;**

**import org.springframework.context.ConfigurableApplicationContext;**

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

**public class Main {**

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

**ConfigurableApplicationContext applicationContext=new ClassPathXmlApplicationContext("spring-config.xml");**

***//Question1***

**applicationContext.start();**

***//Question1***

**applicationContext.stop();**

**applicationContext.close();**

**}**

**}**

**Spring-config.xml**

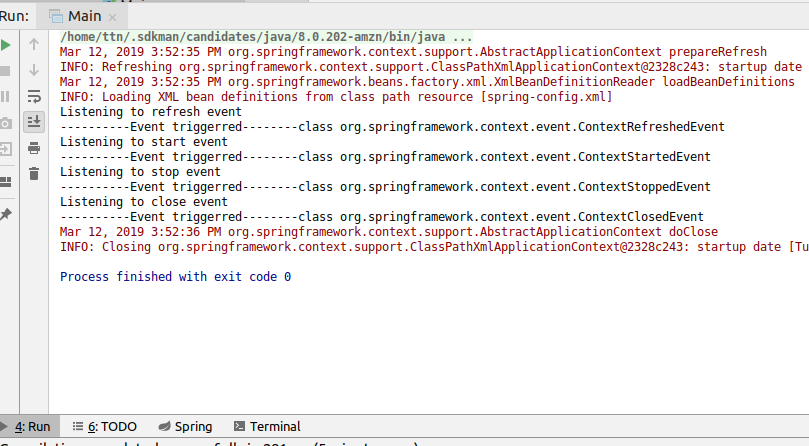
***<!--Beans for Question 2-->***

**<bean class="demo.listeners.SpringEventsListener"/>**

**<bean class="demo.listeners.SpringMultiEventListener"/>**

**<context:annotation-config/>**

**Output**



**Q3. Create a CustomEvent which should get published when you invoke connect method of database bean.**

**Solution**

**Database.java**

**package demo;**

**import demo.event.DatabaseConnectEvent;**

**import org.springframework.context.ApplicationEventPublisher;**

**import org.springframework.context.ApplicationEventPublisherAware;**

**public class Database implements ApplicationEventPublisherAware {**

**private int port;**

**private String name;**

**private ApplicationEventPublisher applicationEventPublisher;**

**public int getPort() {**

**return port;**

**}**

**public void setPort(int port) {**

**this.port = port;**

**}**

**public String getName() {**

**return name;**

**}**

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

**this.name = name;**

**}**

**public void connect()**

**{**

**System.*out*.println("Connecting to database with port "+port+" and name "+name);**

**applicationEventPublisher.publishEvent(new DatabaseConnectEvent(this));**

**}**

**@Override**

**public void setApplicationEventPublisher(ApplicationEventPublisher applicationEventPublisher) {**

**this.applicationEventPublisher=applicationEventPublisher;**

**}**

**}**

**DatabaseConnectEvent.java**

**package demo.event;**

**import org.springframework.context.ApplicationEvent;**

**public class DatabaseConnectEvent extends ApplicationEvent {**

**public DatabaseConnectEvent(Object source)**

**{**

**super(source);**

**}**

**}**

**DatabaseConnectListener.java**

**package demo.listeners;**

**import demo.event.DatabaseConnectEvent;**

**import org.springframework.context.ApplicationListener;**

**public class DatabaseConnectListener implements ApplicationListener<DatabaseConnectEvent> {**

**@Override**

**public void onApplicationEvent(DatabaseConnectEvent event) {**

**System.*out*.println("Database connect event called");**

**}**

**}**

**Main.java**

**package demo;**

**import org.springframework.context.ConfigurableApplicationContext;**

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

**public class Main {**

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

**ConfigurableApplicationContext applicationContext=new ClassPathXmlApplicationContext("spring-config.xml");**

***//Question3***

**Database database=applicationContext.getBean("database",Database.class);**

**System.*out*.println(database.getPort());**

**System.*out*.println(database.getName());**

**database.connect();**

**}**

**}**

**Spring-config.xml**

***<!--Bean for Question3-->***

**<bean id="database" class="demo.Database">**

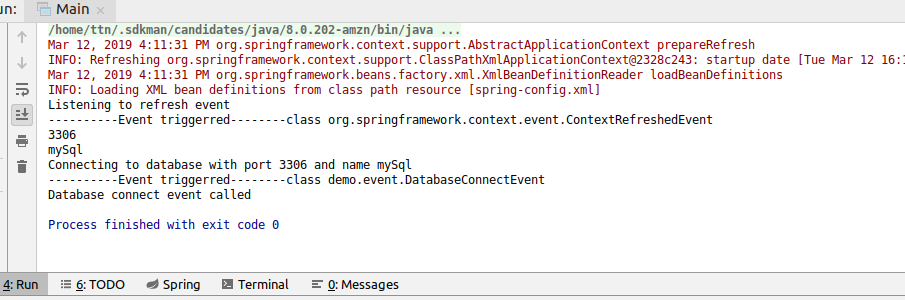
**<property name="port" value="3306"/>**

**<property name="name" value="mySql"/>**

**</bean>**

**<bean class="demo.listeners.DatabaseConnectListener"/>**

**Output**



**Q4. Create a logging aspect. Create a pointcut to log all methods Of services .**

**Solution**

**Spring-config.xml**

***<!--Settings for Question4onwards-->***

**<aop:aspectj-autoproxy/>**

***<!--Question4-->***

**<bean class="demo.GreeterService" name="service"/>**

**<bean class="demo.aspect.LoggingAspect"/>**

**LoggingAspect.java**

**package demo.aspect;**

**import org.aspectj.lang.JoinPoint;**

**import org.aspectj.lang.ProceedingJoinPoint;**

**import org.aspectj.lang.annotation.After;**

**import org.aspectj.lang.annotation.Around;**

**import org.aspectj.lang.annotation.Aspect;**

**import org.aspectj.lang.annotation.Pointcut;**

**@Aspect**

**public class LoggingAspect {**

**@Pointcut("this(demo.GreeterService)")**

**void pointCutService(){}**

**}**

**GreeterService.java**

**package demo;**

**public class GreeterService {**

**void greet()**

**{**

**System.*out*.println("Good Morning");**

**}**

**void greet(String name)**

**{**

**System.*out*.println("Good Morning"+name);**

**}**

**void greet(String name,String message)**

**{**

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

**}**

**}**

**Q5. Add a logging statement before and after the method ends**

**Solution**

**LoggingAspect.java**

**package demo.aspect;**

**import org.aspectj.lang.JoinPoint;**

**import org.aspectj.lang.ProceedingJoinPoint;**

**import org.aspectj.lang.annotation.After;**

**import org.aspectj.lang.annotation.Around;**

**import org.aspectj.lang.annotation.Aspect;**

**import org.aspectj.lang.annotation.Pointcut;**

**@Aspect**

**public class LoggingAspect {**

***//for Question5***

**@Around("pointCutService()")**

**public ProceedingJoinPoint logMethod(ProceedingJoinPoint joinPoint)**

**{**

**System.*out*.println("Before Method: "+joinPoint.getSignature());**

**try {**

**joinPoint.proceed();**

**} catch (Throwable throwable) {**

**throwable.printStackTrace();**

**}**

**System.*out*.println("After Method: "+joinPoint.getSignature());**

**return joinPoint;**

**}**

**@Pointcut("this(demo.GreeterService)")**

**void pointCutService(){}**

**}**

**Main.java**

**package demo;**

**import org.springframework.context.ConfigurableApplicationContext;**

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

**public class Main {**

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

**ConfigurableApplicationContext applicationContext=new ClassPathXmlApplicationContext("spring-config.xml");**

***//Question5***

**GreeterService greeterService=applicationContext.getBean("service",GreeterService.class);**

**greeterService.greet();**

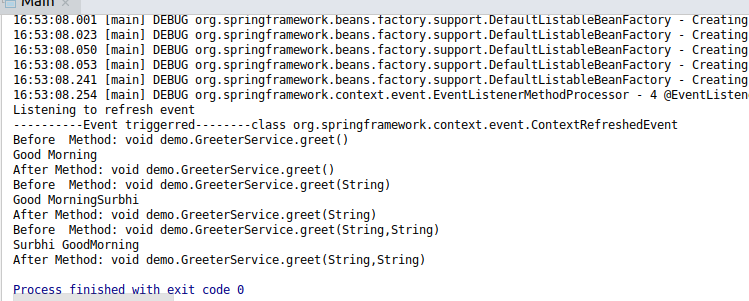
**greeterService.greet("Surbhi");**

**greeterService.greet("Surbhi","GoodMorning");**

**}**

**}**

**Output**



**Q6. Log all the methods which throw IOException**

**Solution**

**LoggingAspect.java**

**package demo.aspect;**

**import org.aspectj.lang.JoinPoint;**

**import org.aspectj.lang.ProceedingJoinPoint;**

**import org.aspectj.lang.annotation.\*;**

**import java.io.IOException;**

**@Aspect**

**public class LoggingAspect {**

***//for Question6***

**@AfterThrowing(pointcut = "execution(\* demo.\*.\*())",throwing = "ioException")**

**public void logExceptionMethods(JoinPoint joinPoint,IOException ioException) throws Throwable**

**{**

**System.*out*.println("IOException thrown from: "+joinPoint.getSignature());**

**}**

**}**

**IoExceptionThrowers.java**

**package demo;**

**import java.io.IOException;**

**public class IoExceptionThrowers {**

**public void display() throws IOException {**

**System.*out*.println("Display: Throwing IOException");**

**throw new IOException();**

**}**

**public void info() throws IOException**

**{**

**System.*out*.println("Info: Throwing IOException");**

**throw new IOException();**

**}**

**}**

**Spring-config.xml**

***<!--Question4 and 10-->***

**<bean class="demo.aspect.LoggingAspect"/>**

***<!--Question10-->***

**<bean class="demo.IoExceptionThrowers"/>**

**Main.java**

**package demo;**

**import org.springframework.context.ConfigurableApplicationContext;**

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

**import java.io.IOException;**

**public class Main {**

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

**ConfigurableApplicationContext applicationContext = new ClassPathXmlApplicationContext("spring-config.xml");**

***//Question6***

**System.*out*.println("------------Question6----------------------");**

**IoExceptionThrowers ioExceptionThrowers = applicationContext.getBean(IoExceptionThrowers.class);**

**try {**

**ioExceptionThrowers.info();**

**} catch (IOException ioException) {**

**System.*out*.println("Executing catch");**

**System.*out*.println("caught "+ioException.getClass());**

**}**

**try {**

**ioExceptionThrowers.display();**

**} catch (IOException e) {**

**System.*out*.println("Executing catch");**

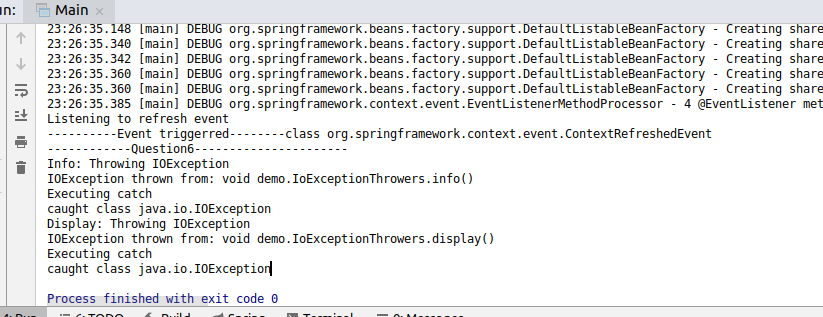
**System.*out*.println("caught "+e.getClass());**

**}**

**}**

**}**

**Output**



**Q7. Apply execution, within, args, this and bean expressions on the before and after advice of service bean.**

**Solution**

**LoggingAspect.java**

**package demo.aspect;**

**import org.aspectj.lang.JoinPoint;**

**import org.aspectj.lang.ProceedingJoinPoint;**

**import org.aspectj.lang.annotation.\*;**

**import java.io.IOException;**

**@Aspect**

**public class LoggingAspect {**

***//for Question4***

**@Pointcut("this(demo.GreeterService)")**

**void pointCutService(){}**

***//for Question7***

***//use-->execution***

**@Before("execution(public \* demo.GreeterService.\*(..))")**

**public void executeBeforeService(JoinPoint joinPoint)**

**{**

**System.*out*.println("Logging before method of greeter service "+joinPoint.getSignature());**

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

**}**

**@After("execution(public \* demo.GreeterService.\*(..))")**

**public void executeAfterService(JoinPoint joinPoint)**

**{**

**System.*out*.println("Logging after method of greeter service "+joinPoint.getSignature());**

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

**}**

***//use-->within***

**@Before("within(demo.GreeterService)")**

**public void executeBeforeAllService(JoinPoint joinPoint)**

**{**

**System.*out*.println("Logging before service methods using within for method"+ joinPoint.getSignature());**

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

**}**

**@After("within(demo.GreeterService)")**

**public void executeAfterAllService(JoinPoint joinPoint)**

**{**

**System.*out*.println("Logging after service methods using within for method"+ joinPoint.getSignature());**

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

**}**

***//use-->args***

**@Before("args(String)")**

**public void logBeforeMethodsWithStringArgument(JoinPoint joinPoint)**

**{**

**System.*out*.println("Logging before methods with string argument"+joinPoint.getSignature());**

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

**}**

**@After("args(String)")**

**public void logAfterMethodsWithStringArgument(JoinPoint joinPoint)**

**{**

**System.*out*.println("Logging after methods with string argument"+joinPoint.getSignature());**

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

**}**

***//use-->this***

**@Before("pointCutService()")**

**public void logBeforeUsingThis(JoinPoint joinPoint)**

**{**

**System.*out*.println("Logging before using this: "+joinPoint.getSignature());**

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

**}**

**@After("pointCutService()")**

**public void logAfterUsingThis(JoinPoint joinPoint)**

**{**

**System.*out*.println("Logging after using this: "+joinPoint.getSignature());**

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

**}**

***//use-->bean***

**@Before("bean(service)")**

**public void logBeforeUsingBean(JoinPoint joinPoint)**

**{**

**System.*out*.println("Logging before using bean: "+joinPoint.getSignature());**

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

**}**

**@After("bean(service)")**

**public void logAfterUsingBean(JoinPoint joinPoint)**

**{**

**System.*out*.println("Logging after using bean: "+joinPoint.getSignature());**

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

**}**

**}**

**Main.java**

**package demo;**

**import org.springframework.context.ConfigurableApplicationContext;**

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

**import java.io.IOException;**

**public class Main {**

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

**ConfigurableApplicationContext applicationContext = new ClassPathXmlApplicationContext("spring-config.xml");**

***//Question7***

**System.*out*.println("------------Question7----------------------");**

**GreeterService greeterService=applicationContext.getBean("service",GreeterService.class);**

**greeterService.greet();**

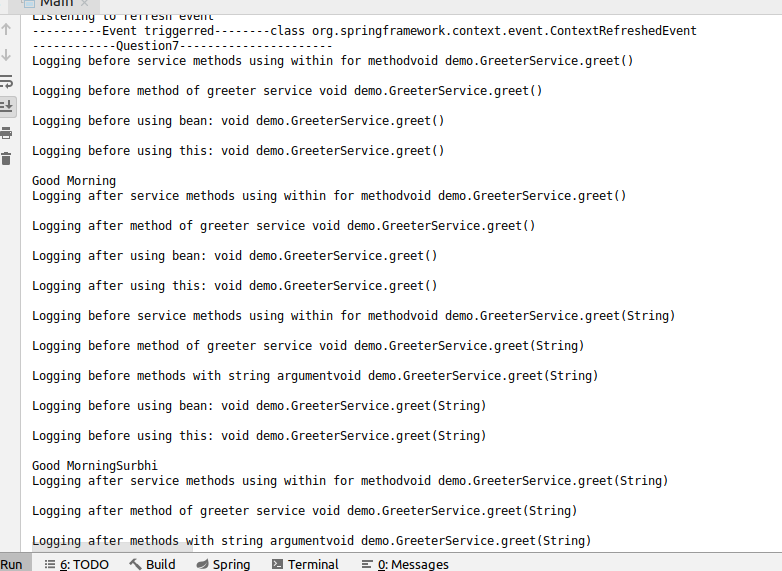
**greeterService.greet("Surbhi");**

**greeterService.greet("Surbhi","GoodMorning");**

**}**

**}**

**Output**



**Q8. Demonstrate the use of pointCut @Pointcut annotation and Reuse the expression created**

**Solution**

**LoggingAspect.java**

**package demo.aspect;**

**import org.aspectj.lang.JoinPoint;**

**import org.aspectj.lang.ProceedingJoinPoint;**

**import org.aspectj.lang.annotation.\*;**

**import java.io.IOException;**

**@Aspect**

**public class LoggingAspect {**

***//for Question4***

**@Pointcut("this(demo.GreeterService)")**

**void pointCutService(){}**

***//use-->this***

***//point-cut reuse--Question8***

**@Before("pointCutService()")**

**public void logBeforeUsingThis(JoinPoint joinPoint)**

**{**

**System.*out*.println("Logging before using this: "+joinPoint.getSignature());**

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

**}**

**@After("pointCutService()")**

**public void logAfterUsingThis(JoinPoint joinPoint)**

**{**

**System.*out*.println("Logging after using this: "+joinPoint.getSignature());**

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

**}**

**}**

**Q9. Access the properties of the JoinPoint in before advice. Print Signature of method being called and its arguments**

**Solution**

**LoggingAspect.java**

**package demo.aspect;**

**import org.aspectj.lang.JoinPoint;**

**import org.aspectj.lang.ProceedingJoinPoint;**

**import org.aspectj.lang.annotation.\*;**

**import java.io.IOException;**

**import java.util.Arrays;**

**@Aspect**

**public class LoggingAspect {**

***//for Question4***

**@Pointcut("this(demo.GreeterService)")**

**void pointCutService(){}**

***//for Question7***

***//for Question9--used joint point and printed method signature and arguments in before***

***//use-->execution***

**@Before("execution(public \* demo.GreeterService.\*(..))")**

**public void executeBeforeService(JoinPoint joinPoint)**

**{**

**System.*out*.println("Logging before method of greeter service "+joinPoint.getSignature());**

**System.*out*.println("Arguments: "+ Arrays.*toString*(joinPoint.getArgs()));**

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

**}**

***//use-->within***

**@Before("within(demo.GreeterService)")**

**public void executeBeforeAllService(JoinPoint joinPoint)**

**{**

**System.*out*.println("Logging before service methods using within for method"+ joinPoint.getSignature());**

**System.*out*.println("Arguments: "+ Arrays.*toString*(joinPoint.getArgs()));**

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

**}**

***//use-->args***

**@Before("args(String)")**

**public void logBeforeMethodsWithStringArgument(JoinPoint joinPoint)**

**{**

**System.*out*.println("Logging before methods with string argument"+joinPoint.getSignature());**

**System.*out*.println("Arguments: "+ Arrays.*toString*(joinPoint.getArgs()));**

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

**}**

***//use-->this***

***//point-cut reuse--Question8***

**@Before("pointCutService()")**

**public void logBeforeUsingThis(JoinPoint joinPoint)**

**{**

**System.*out*.println("Logging before using this: "+joinPoint.getSignature());**

**System.*out*.println("Arguments: "+ Arrays.*toString*(joinPoint.getArgs()));**

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

**}**

***//use-->bean***

**@Before("bean(service)")**

**public void logBeforeUsingBean(JoinPoint joinPoint)**

**{**

**System.*out*.println("Logging before using bean: "+joinPoint.getSignature());**

**System.*out*.println("Arguments: "+ Arrays.*toString*(joinPoint.getArgs()));**

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

**}**

**}**

**Main.java(Combined for all Questions)**

**Other questions not in main require either creating a class or some configurations in spring-config.xml, which are step by step mentioned above.**

**package demo;**

**import org.springframework.context.ConfigurableApplicationContext;**

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

**import java.io.IOException;**

**public class Main {**

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

**ConfigurableApplicationContext applicationContext = new ClassPathXmlApplicationContext("spring-config.xml");**

***//Question1***

**applicationContext.start();**

***//Question3***

**System.*out*.println("------------Question3----------------------");**

**Database database=applicationContext.getBean("database",Database.class);**

**System.*out*.println(database.getPort());**

**System.*out*.println(database.getName());**

**database.connect();**

***//Question5 and Question7 andQuestion9***

**System.*out*.println("------------Question5,7,9----------------------");**

**GreeterService greeterService=applicationContext.getBean("service",GreeterService.class);**

**greeterService.greet();**

**greeterService.greet("Surbhi");**

**greeterService.greet("Surbhi","GoodMorning");**

***//Question6***

**System.*out*.println("------------Question6----------------------");**

**IoExceptionThrowers ioExceptionThrowers = applicationContext.getBean(IoExceptionThrowers.class);**

**try {**

**ioExceptionThrowers.info();**

**} catch (IOException ioException) {**

**System.*out*.println("Executing catch");**

**System.*out*.println("caught "+ioException.getClass());**

**}**

**try {**

**ioExceptionThrowers.display();**

**} catch (IOException e) {**

**System.*out*.println("Executing catch");**

**System.*out*.println("caught "+e.getClass());**

**}**

***// Question1***

**applicationContext.stop();**

**applicationContext.close();**

**}**

**}**