**Spring AOP AspectJ Xml Configuration Example**

Spring enables you to define the aspects, advices and pointcuts in xml file.

In the previous page, we have seen the aop examples using annotations. Now we are going to see same examples by the xml configuration file.

Let's see the xml elements that are used to define advice.

1. **aop:before** It is applied before calling the actual business logic method.
2. **aop:after** It is applied after calling the actual business logic method.
3. **aop:after-returning** it is applied after calling the actual business logic method. It can be used to intercept the return value in advice.
4. **aop:around** It is applied before and after calling the actual business logic method.
5. **aop:after-throwing** It is applied if actual business logic method throws exception.

#### 1) aop:before Example

The AspectJ Before Advice is applied before the actual business logic method. You can perform any operation here such as conversion, authentication etc.

Create a class that contains actual business logic.

**Education.java**

**package** com.doj.spring;

**public** **class** Education {

String[] course={"B.tech","MCA","B.Phrma","M.Tech","MBA"};

**public** **void** courseDetails()

{

**for**(**int** i=0;i<course.length;i++)

System.*out*.println("\t\t\t "+course[i]);

}

}

**Colleges.java**

**package** com.doj.spring;

**public** **class** Colleges {

String name[]={"Dr. K.N. Modi","MIET","KIET","SASIIT&R"};

**int** collegeCode[]={101,102,103,104};

String city[]={"Modinagar","Meerut","Muradnagar","Mohali"};

**public** **void** collegeDetails()

{

System.*out*.println("\t\t\t College Code\t\t\t College Name\t\t\t College City");

**for**(**int** i=0;i<name.length;i++)

{

System.*out*.println("\t\t\t "+collegeCode[i]+"\t\t\t\t "+name[i]+"\t\t\t "+city[i]);

}

}

}

Now, create the aspect class that contains before advice and after advice

**LoggingAspect.java**

**package** com.doj.spring.aop;

**public** **class** LoggingAspect {

**public** **void** myEducationAspectStart()

{

System.*out*.println("------Courses Details are here please go through these Details------");

}

**public** **void** myEducationAspectEnd()

{

System.*out*.println("------Courses Details are Ended please see above------");

}

**public** **void** myCollegesAspectStart()

{

System.*out*.println(" =======College Details are here please go through these Details==========");

}

**public** **void** myCollegesAspectEnd()

{

System.*out*.println(" =======College Details are Ended please see above==========");

}

}

Now create the applicationContext.xml file that defines beans.

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

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/aop*

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

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

<!-- Component scaning from this base package -->

<!-- <context:component-scan base-package="com.doj.spring"/>

-->

<aop:aspectj-autoproxy />

<bean id=*"LoggingAspect"* class=*"com.doj.spring.aop.LoggingAspect"*> </bean>

<bean id=*"colleges"* class=*"com.doj.spring.Colleges"*></bean>

<bean id=*"education"* class=*"com.doj.spring.Education"*></bean>

<aop:config>

<aop:aspect id=*"myaspect"* ref=*"LoggingAspect"* >

<!-- @Before -->

<aop:pointcut id=*"pointCutBeforeEducation"* expression=*"execution(\* com.doj.spring.Education.courseDetails(..))"* />

<aop:before method=*"myEducationAspectStart"* pointcut-ref=*"pointCutBeforeEducation"* />

<aop:pointcut expression=*"execution(\* com.doj.spring.Education.courseDetails(..))"* id=*"pointCutAfterEducation"*/>

<aop:after method=*"myEducationAspectEnd"* pointcut-ref=*"pointCutAfterEducation"*/>

<aop:pointcut expression=*"execution(\* com.doj.spring.Colleges.collegeDetails(..))"* id=*"pointCutBeforeColleges"*/>

<aop:before method=*"myCollegesAspectStart"* pointcut-ref=*"pointCutBeforeColleges"*/>

<aop:pointcut expression=*"execution(\* com.doj.spring.Colleges.collegeDetails(..))"* id=*"pointCutAfterColleges"*/>

<aop:after method=*"myCollegesAspectEnd"* pointcut-ref=*"pointCutAfterColleges"*/>

</aop:aspect>

</aop:config>

</beans>

Now, let's call the actual method.

**MainClass.java**

**package** com.doj.spring;

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

**public** **class** MainClass {

/\*\*

\* **@param** args

\*/

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

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

Education education= (Education)context.getBean("education");

Colleges colleges=(Colleges)context.getBean("colleges");

//System.out.println("Education's method is calling.....");

education.courseDetails();

//System.out.println("Colleges's method is calling.......");

colleges.collegeDetails();

}

}

# Spring AOP AspectJ Annotation Example

In this Example We are going to use Java Based Configuration.

Business Logic class.

**Car.java**

package com.doj.spring;

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

import org.springframework.stereotype.Component;

/\*

\* @Component annotation is used to define bean.

\* It is like <bean id=*"car"* class=*"com.doj.spring.Car"*/>

\*/

@Component

public class Car {

public Car() {

super();

}

public void driveCar(int carNumber){

System.out.println("You are driving a car which has a engine”);

try{

String s = null;

s.charAt(0);

}catch(Exception e){

}

}

public void startCar(){

System.out.println("Car is started!!!");

}

}

Java Based Configuration file

**SpringConfiguration.java**

package com.doj.spring;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.EnableAspectJAutoProxy;

import com.doj.spring.aop.LoggingAspect;

@Configuration

/\*

\* @Configuration annotation is used to define java based configuration this works like spring.xml

\* @EnableAspectJAutoProxy in XML <aop:aspectj-autoproxy></aop:aspectj-autoproxy>

\*/

@ComponentScan(basePackages={"com.doj.spring"})

@EnableAspectJAutoProxy

public class SpringConfiguration {

@Bean

public LoggingAspect getLoggingAspect(){

return new LoggingAspect();

}

}

Aspect class is here.

**LoggingAspect.java**

package com.doj.spring.aop;

import org.aspectj.lang.JoinPoint;

import org.aspectj.lang.annotation.After;

import org.aspectj.lang.annotation.AfterReturning;

import org.aspectj.lang.annotation.AfterThrowing;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Before;

import org.aspectj.lang.annotation.Pointcut;

@Aspect

public class LoggingAspect {

@Pointcut("execution(\* com.doj.spring.Car.driveCar(int))")

public void getDL(){}

@Pointcut("execution(\* com.doj.spring.Car.driveCar()) || within(com.doj.spring.Car)")

public void getBasic(){}

@Pointcut("execution(\* com.doj.spring.Car.driveCar(int)) and args(carNumber1)")

public void getExpert(){}

@Before("getBasic() || getDL()")

public void getDrivingLiensense(JoinPoint joinPoint){

System.out.println("Before Driving Get Driving Liencense!!!"+joinPoint.getSignature().getDeclaringTypeName());

}

@Before("getBasic()")

public void getBasicAboutCar(JoinPoint joinPoint){

System.out.println("Get Basic Knowledge about Car!!!"+joinPoint.toLongString());

}

@After("execution(\* com.doj.spring.Car.driveCar(..))")

public void getElegibleForDriving(JoinPoint joinPoint){

System.out.println("You are eligible for drive a Car!!!"+joinPoint.getKind());

}

@AfterReturning("getExpert()")

public void getExpertInDriving(JoinPoint joinPoint){

System.out.println("You are expert in Driving!!!"+joinPoint.getStaticPart().getKind());

}

@AfterThrowing("execution(\* com.doj.spring.Car.driveCar(..))")

public void getNotElegibleForDriving(JoinPoint joinPoint){

System.out.println("You are not eligible for drive a Car!!!");

}

/\*@Around("execution(\* com.doj.spring.Car.driveCar(..))")

public void checkYourCar(JoinPoint joinPoint){

System.out.println("Check your car before and after driving a Car!!!");

}\*/

}

Main Class is here

**CarDemo.java**

package com.doj.spring;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

import org.springframework.context.support.AbstractApplicationContext;

public class CarDemo {

public static void main(String[] args) {

AbstractApplicationContext context = new AnnotationConfigApplicationContext(SpringConfiguration.class);

Car car = (Car) context.getBean("car");

car.startCar();

car.driveCar(1);

context.close();

}

}