AOP

Aspect Oriented Programming

Cross cutting concerns

Aspect (class)

Advices (methods)

Bean (Bank)

Business methods (withdraw, deposit)

JoinPoint in which bean method invocation and advice connects

Pointcut Expression is in SPeL Spring Expression Language

Demo:

1. Create a maven project “Quickstart” archetype “02-jun-aop-xml-1”
2. In pom.xml add spring context dependency

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>4.3.6.RELEASE</version>

</dependency>

1. Add bean.xml to the project

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

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

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

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

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

</beans>

1. Lets create a bean class “Bank” with methods: withdraw and deposit

package com.ust.aop.model;

public class Bank {

public void withdraw()

{

System.out.println("withdraw: Remember to collect the cash");

}

public void deposit()

{

System.out.println("deposit: Thank you for banking with us.");

}

}

1. In bean.xml configure the Bank class as a bean

<bean id="bank" class="com.ust.aop.model.Bank">

</bean>

1. Lets create Aspect class “RBIAspect”

Create methods like validatePin(), checkBalance(), printReceipt()

package com.ust.aop.aspect;

public class RBIAspect {

public void validatePin() {

System.out.println("Pin number is validated...");

}

public void checkBalance()

{

System.out.println("Balance is checked and sufficient funds available");

}

public void printReceipt()

{

System.out.println("Receipt is printed...");

}

}

1. In bean.xml, configure the RBIAspect class as an aspect

Now, for AOP, we need to add dependencies to the pom.xml

1. Aspectjrt
2. Aspectjweaver
3. Aop

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>4.3.6.RELEASE</version>

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjrt</artifactId>

<version>1.9.5</version>

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.9.5</version>

</dependency>

1. In bean.xml,
   1. Import xmlns for aop
   2. Define aspect

<?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"

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/aop-3.0.xsd

">

<aop:config>

<aop:aspect id="rbi" ref="rbiAspect">

<aop:pointcut expression="execution(\* com.ust.aop.model.\*.\*(...))" id="pce1" />

<aop:before pointcut-ref="pce1" method="validatePin" />

<aop:before pointcut-ref="pce1" method="checkBalance" />

<aop:after pointcut-ref="pce1" method="printReceipt" />

</aop:aspect>

</aop:config>

<bean id="rbiAspect" class="com.ust.aop.aspect.RBIAspect">

</bean>

<bean id="bank" class="com.ust.aop.model.Bank">

</bean>

</beans>

1. In App.java, get bean of type Bank and call withdraw method

package com.ust.aop;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.ust.aop.model.Bank;

/\*\*

\* Hello world!

\*

\*/

public class App

{

public static void main( String[] args )

{

System.out.println( "Hello World!" );

ClassPathXmlApplicationContext ctx=new ClassPathXmlApplicationContext("bean.xml");

Bank bank=(Bank) ctx.getBean("bank");

bank.withdraw();

bank.deposit();

}

}

How to work with AOP Annotations.

We don’t need bean.xml file.

@Component to define bean (Bank)

@Aspect to define aspect (RBIAspect)

@PointCut to write expression

@Before

@After

Task:

Do the AOP using annotations (without xml)

Steps:

package model;

import org.springframework.stereotype.Component;

@Component("employee")

public class Employee {

private int id;

private String firstName;

private String lastName;

public Employee()

{}

public Employee(int id, String firstName, String lastName) {

super();

this.id = id;

this.firstName = firstName;

this.lastName = lastName;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

@Override

public String toString() {

return "Employee [id=" + id + ", firstName=" + firstName + ", lastName=" + lastName + "]";

}

public void method1()

{

System.out.println("Thsi is employee method1 executed now ");

}

public void accept()

{

System.out.println("Employee class Accept method");

}

}

=========================================

package model;

import org.aspectj.lang.annotation.After;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Before;

import org.aspectj.lang.annotation.Pointcut;

import org.springframework.context.annotation.EnableAspectJAutoProxy;

import org.springframework.stereotype.Component;

@Aspect

@EnableAspectJAutoProxy

@Component("logging")

public class Logging {

@Pointcut("execution(\* model.\*.m\*(..))")

public void selectAll()

{

}

@Before("selectAll()")

public void beforeMethod()

{

System.out.println("before method is executed");

}

@After("selectAll()")

public void afterMethod()

{

System.out.println("After method is executed");

}

@Pointcut("execution(\* model.\*.a\*(..))")

public void aMethods()

{

// System.out.println("select all");

}

@Before("aMethods()")

public void beforeAmethods()

{

System.out.println("before A methods");

}

}

Swagger steps:

1. In pom.xml

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger2</artifactId>

<version>2.9.2</version>

</dependency>

1. In application.properties:

spring.mvc.pathmatch.matching-strategy=ant-path-matcher

1. Application.java (under @SpringBootApplication)

@EnableSwagger2

Restart the project.

Summary of steps of swagger:

----------------------

1) Create Spring Boot App

2) Create a REST api - GET/POST

3) Add spring fox spring boot starter

4) Enable swagger2 in spring boot application

5) Run application

6) Update api documentation and run the app

io.springfox

springfox-boot-starter

3.0.0

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-boot-starter</artifactId>

<version>3.0.0</version>

</dependency>

@Configuration

@EnableSwagger2

in application.properties:

spring.mvc.pathmatch.matching-strategy=ant-path-matcher

<http://localhost:8080/swagger-ui/>