AspectJ, Spring, Maven

Меня в нашей группе попросили настроить среду и показать использование аспектов AspectJ и интеграция его со Spring.  
Мне показалось, что хабросообществу это тоже может быть интересно.  
  
Не буду рассказывать тут о том, что такое AspectJ, кто знает — тому будет полезно, замечу лишь, что аспекты — это возможность добавить на этапе компиляции или рантайма в классы некую функциональность, которой раньше там не было. Или изменить существующую.  
  
Далее: конфигурация проекта и 3 примера аспектов.  
  
  
Начнём с конфигурации maven:

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>org.habloexample</groupId>

<artifactId>aspects</artifactId>

<packaging>jar</packaging>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.8.2</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>3.0.5.RELEASE</version>

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjrt</artifactId>

<version>1.6.7</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>2.0.2</version>

<configuration>

<source>${maven.compiler.source}</source>

<target>${maven.compiler.target}</target>

</configuration>

</plugin>

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>aspectj-maven-plugin</artifactId>

<version>1.3</version>

<configuration>

<source>${maven.compiler.source}</source>

<target>${maven.compiler.target}</target>

</configuration>

<executions>

<execution>

<goals>

<goal>compile</goal>

<goal>test-compile</goal>

</goals>

</execution>

</executions>

</plugin>

</plugins>

</build>

<properties>

<org.springframework.version>3.0.5.RELEASE</org.springframework.version>

<maven.compiler.source>1.6</maven.compiler.source>

<maven.compiler.target>1.6</maven.compiler.target>

</properties>

</project>

Далее, в Intellij IDEA:

Plugin AspectJ: enable

Plugin AspectJ Weaver: disable

Settings/Compile/JavaCompiler: Ajc

Пример 1, AspectJ с аннотациями, Spring с аннотациями:

(пример взят из [www.javacodegeeks.com/2010/07/aspect-oriented-programming-with-spring.html](http://www.javacodegeeks.com/2010/07/aspect-oriented-programming-with-spring.html))

Создаём класс:

|  |
| --- |
| package org.habr.springaspectj.services;  import org.springframework.stereotype.Service;  @Service("greetingService")  public class GreetingService {  public static final String HELLO\_FROM\_GREETING\_SERVICE = "Hello from Greeting Service";  public String sayHello() { return HELLO\_FROM\_GREETING\_SERVICE; }  } |

Создаём аспект:

|  |
| --- |
| package org.habr.springaspectj.aspects;  import org.aspectj.lang.ProceedingJoinPoint;  import org.aspectj.lang.annotation.Around;  import org.aspectj.lang.annotation.Aspect;  @Aspect  public class GreetingAspect {  private String message;  public void setMessage(String message) { this.message = message; }  @Around("execution(\* org.habr.springaspectj.services.GreetingService.\*(..))")  public Object advice(ProceedingJoinPoint pjp) throws Throwable {  String serviceGreeting = (String) pjp.proceed(); return message + " and " + serviceGreeting; }  } |

Создаём test-aspectj.xml для Spring:

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

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

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

xmlns:jee="http://www.springframework.org/schema/jee" xmlns:tx="http://www.springframework.org/schema/tx"

xmlns:task="http://www.springframework.org/schema/task"

xsi:schemaLocation="

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

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

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

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

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

<context:component-scan base-package="org.habr.springaspectj" />

<bean class="org.habr.springaspectj.aspects.GreetingAspect" factory-method="aspectOf">

<property name="message" value="Hello from Greeting Aspect"/>

</bean>

<beans>

Пишем тест:

|  |
| --- |
| package org.habr.springaspectj;  import org.habr.springaspectj.services.GreetingService;  import org.habr.springaspectj.services.IncrementService;  import org.junit.Before;  import org.junit.Test;  import org.springframework.context.ApplicationContext;  import org.springframework.context.support.ClassPathXmlApplicationContext;  import java.lang.reflect.InvocationTargetException;  import java.lang.reflect.Method;  import static junit.framework.Assert.assertEquals;  import static junit.framework.Assert.assertTrue;  public class AspectAnnotationTest {  ApplicationContext context ;  @Before  public void init() {  context = new ClassPathXmlApplicationContext("test-aspectj.xml");  }  @Test  public void testAnnotationService() {  GreetingService greetingService = (GreetingService) context.getBean("greetingService"); assertTrue(greetingService.sayHello().contains(GreetingService.HELLO\_FROM\_GREETING\_SERVICE)); assertTrue(greetingService.sayHello().length()>GreetingService.HELLO\_FROM\_GREETING\_SERVICE.length()); }  } |

Сервис и аспект идут в соответствующие пакеты в src/main/java

Тест в src/test/java

Тест проверяет что к исходному сообщению что то добавилось.

Можете распечатать greetingService.sayHello() и увидеть, что добавилась строка, которую передали в xml аспекту.

Пример 2. Без аннотаций, изменение поведения метода.

Добавим сервис:

|  |
| --- |
| package org.habr.springaspectj.services;  public class IncrementService {  public int inc(int i){ return i+1; }  } |

Добавим аспект:

|  |
| --- |
| package org.habr.springaspectj.aspects;  public aspect DecrementAspect {  pointcut incMethod(): execution(public int inc(int));  int around(int number): incMethod() && args(number) { return proceed(number) - 1;  }  } |

Добавим сервис в xml (обещали — в этот раз без аннотаций):

|  |
| --- |
| <bean name="inc-bean" class="org.habr.springaspectj.services.IncrementService"/> |

Добавим метод в юнит тест:

|  |
| --- |
| @Test  public void testBeanAndAJService() {  IncrementService service = context.getBean(IncrementService.class);  int i = 10; assertEquals(i,service.inc(i));  } |

Ура! делаем инкремент, а число не увеличивается!

Примечание: поддержка аспектов в IDEA не совершенна. Приведенный выше аспект будет отмечен красным, но компилироваться бует — и мавеном и IDEA.

Пример 3: Добавление не существующего метода

Сервис из предыдущего примера, добавим аспект:

|  |
| --- |
| package org.habr.springaspectj.aspects;  public aspect AddingAspect {  public String org.habr.springaspectj.services.IncrementService.myCall(String name) {  return "Cognac " + name; }  } |

Добавим метод в тест:

|  |
| --- |
| @Test  public void testAddAMethod() throws NoSuchMethodException, InvocationTargetException, IllegalAccessException {  IncrementService service = context.getBean(IncrementService.class); assertEquals(service.myCall("Hennesy"),"Cognac Hennesy");  } |

Несмотря на отсутствие в классе сервиса метода myCall, IDEA не жалуется и всё компилироется

Выводы:

~~Аспекты мягко говоря не повышают понимание, читабельность кода, я бы не советовал пользоваться ими без особой нужды.~~

~~Но если очень нужно — эти примеры могут помочь.~~

В комментах посоветовали изменить вывод…

Аспекты надо применять в нескольких областях: журналирование, аудит, транзакции.

И очень осторожно надо их использовать в других областях. С аспектами легко сделать так, что читаете в коде одно, а работает что то совсем другое.