**Spring 中基于 AOP 的 @AspectJ**

项目地址：https://github.com/xuegang20/mystudyrecode/tree/main/code/SpringExample

**Spring 中基于 AOP 的 @AspectJ**

@AspectJ 作为通过 Java 5 注释注释的普通的 Java 类，它指的是声明 aspects 的一种风格。通过在你的基于架构的 XML 配置文件中包含以下元素，@AspectJ 支持是可用的。

<aop:aspectj-autoproxy/>

你还需要在你的应用程序的 CLASSPATH 中使用以下 AspectJ 库文件。这些库文件在一个 AspectJ 装置的 ‘lib’ 目录中是可用的，如果没有，你可以在 Internet 中下载它们。

* aspectjrt.jar
* aspectjweaver.jar
* aspectj.jar
* aopalliance.jar

**声明一个 aspect**

Aspects 类和其他任何正常的 bean 一样，除了它们将会用 @AspectJ 注释之外，它和其他类一样可能有方法和字段，如下所示：

package org.xyz;

import org.aspectj.lang.annotation.Aspect;

@Aspect

public class AspectModule {

}

它们将在 XML 中按照如下进行配置，就和其他任何 bean 一样：

<bean id="myAspect" class="org.xyz.AspectModule">

<!-- configure properties of aspect here as normal -->

</bean>

**声明一个切入点**

一个**切入点**有助于确定使用不同建议执行的感兴趣的连接点（即方法）。在处理基于配置的 XML 架构时，切入点的声明有两个部分：

* 一个切入点表达式决定了我们感兴趣的哪个方法会真正被执行。
* 一个切入点标签包含一个名称和任意数量的参数。方法的真正内容是不相干的，并且实际上它应该是空的。

下面的示例中定义了一个名为 ‘businessService’ 的切入点，该切入点将与 com.xyz.myapp.service 包下的类中可用的每一个方法相匹配：

import org.aspectj.lang.annotation.Pointcut;

@Pointcut("execution(\* com.xyz.myapp.service.\*.\*(..))") // expression

private void businessService() {} // signature

下面的示例中定义了一个名为 ‘getname’ 的切入点，该切入点将与 com.tutorialspoint 包下的 Student 类中的 getName() 方法相匹配：

import org.aspectj.lang.annotation.Pointcut;

@Pointcut("execution(\* com.tutorialspoint.Student.getName(..))")

private void getname() {}

**声明建议**

你可以使用 @{ADVICE-NAME} 注释声明五个建议中的任意一个，如下所示。这假设你已经定义了一个切入点标签方法 businessService()：

@Before("businessService()")

public void doBeforeTask(){

...

}

@After("businessService()")

public void doAfterTask(){

...

}

@AfterReturning(pointcut = "businessService()", returning="retVal")

public void doAfterReturnningTask(Object retVal){

// you can intercept retVal here.

...

}

@AfterThrowing(pointcut = "businessService()", throwing="ex")

public void doAfterThrowingTask(Exception ex){

// you can intercept thrown exception here.

...

}

@Around("businessService()")

public void doAroundTask(){

...

}

你可以为任意一个建议定义你的切入点内联。下面是在建议之前定义内联切入点的一个示例：

@Before("execution(\* com.xyz.myapp.service.\*.\*(..))")

public doBeforeTask(){

...

}

**基于 AOP 的 @AspectJ 示例**

为了理解上面提到的关于基于 AOP 的 @AspectJ 的概念，让我们编写一个示例，可以实现几个建议。为了在我们的示例中使用几个建议，让我们使 Eclipse IDE 处于工作状态，然后按照如下步骤创建一个 Spring 应用程序：

|  |  |
| --- | --- |
| **步骤** | **描述** |
| 1 | 创建一个名为 *SpringExample* 的项目，并且在所创建项目的 **src** 文件夹下创建一个名为 *com.tutorialspoint* 的包。 |
| 2 | 使用 *Add External JARs* 选项添加所需的 Spring 库文件，就如在 *Spring Hello World Example* 章节中解释的那样。 |
| 3 | 在项目中添加 Spring AOP 指定的库文件 **aspectjrt.jar， aspectjweaver.jar** 和 **aspectj.jar**。 |
| 4 | 在 *com.tutorialspoint* 包下创建 Java 类 **Logging**， *Student* 和 *MainApp*。 |
| 5 | 在 **src** 文件夹下创建 Beans 配置文件 *Beans.xml*。 |
| 6 | 最后一步是创建所有 Java 文件和 Bean 配置文件的内容，并且按如下解释的那样运行应用程序。 |

这里是 **Logging.java** 文件的内容。这实际上是 aspect 模块的一个示例，它定义了在各个点调用的方法。

package com.tutorialspoint;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Pointcut;

import org.aspectj.lang.annotation.Before;

import org.aspectj.lang.annotation.After;

import org.aspectj.lang.annotation.AfterThrowing;

import org.aspectj.lang.annotation.AfterReturning;

import org.aspectj.lang.annotation.Around;

@Aspect

public class Logging {

/\*\* Following is the definition for a pointcut to select

\* all the methods available. So advice will be called

\* for all the methods.

\*/

@Pointcut("execution(\* com.tutorialspoint.\*.\*(..))")

private void selectAll(){}

/\*\*

\* This is the method which I would like to execute

\* before a selected method execution.

\*/

@Before("selectAll()")

public void beforeAdvice(){

System.out.println("Going to setup student profile.");

}

/\*\*

\* This is the method which I would like to execute

\* after a selected method execution.

\*/

@After("selectAll()")

public void afterAdvice(){

System.out.println("Student profile has been setup.");

}

/\*\*

\* This is the method which I would like to execute

\* when any method returns.

\*/

@AfterReturning(pointcut = "selectAll()", returning="retVal")

public void afterReturningAdvice(Object retVal){

System.out.println("Returning:" + retVal.toString() );

}

/\*\*

\* This is the method which I would like to execute

\* if there is an exception raised by any method.

\*/

@AfterThrowing(pointcut = "selectAll()", throwing = "ex")

public void AfterThrowingAdvice(IllegalArgumentException ex){

System.out.println("There has been an exception: " + ex.toString());

}

}

下面是 **Student.java** 文件的内容：

package com.tutorialspoint;

public class Student {

private Integer age;

private String name;

public void setAge(Integer age) {

this.age = age;

}

public Integer getAge() {

System.out.println("Age : " + age );

return age;

}

public void setName(String name) {

this.name = name;

}

public String getName() {

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

return name;

}

public void printThrowException(){

System.out.println("Exception raised");

throw new IllegalArgumentException();

}

}

下面是 **MainApp.java** 文件的内容：

package com.tutorialspoint;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

public static void main(String[] args) {

ApplicationContext context =

new ClassPathXmlApplicationContext("Beans.xml");

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

student.getName();

student.getAge();

student.printThrowException();

}

}

下面是配置文件 **Beans.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: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/spring-aop-3.0.xsd ">

<aop:aspectj-autoproxy/>

<!-- Definition for student bean -->

<bean id="student" class="com.tutorialspoint.Student">

<property name="name" value="Zara" />

<property name="age" value="11"/>

</bean>

<!-- Definition for logging aspect -->

<bean id="logging" class="com.tutorialspoint.Logging"/>

</beans>

一旦你已经完成的创建了源文件和 bean 配置文件，让我们运行一下应用程序。如果你的应用程序一切都正常的话，这将会输出以下消息：

Going to setup student profile.

Name : Zara

Student profile has been setup.

Returning:Zara

Going to setup student profile.

Age : 11

Student profile has been setup.

Returning:11

Going to setup student profile.

Exception raised

Student profile has been setup.

There has been an exception: java.lang.IllegalArgumentException

.....

other exception content