第八章习题答案

1.填空题

（1）面向切面编程

（2）5

（3）Proxy、CGLib

（4）JDK

（5）CGLib

2.选择题

（1）B

（2）C

（3）D

（4）D

（5）D

3.思考题

（1）

答：AOP是Aspect Oriented Programming（面向切面编程）的缩写，和OOP不同，它主张将程序中的相同业务逻辑进行横向隔离，并将重复的业务逻辑抽取到一个独立的模块中，最终实现提升程序可复用性和开发效率的目的。

（2）

答：在Spring中，AOP代理由Spring的IOC容器负责创建，其依赖关系也将由IOC容器负责管理，因此Spring AOP可以直接将IOC容器中的其他Bean实例作为目标对象。在默认情况下，Spring AOP使用JDK动态代理，当目标对象是一个类并且这个类没有实现接口时，Spring会切换为使用CGLib代理。

4.编程题

Ⅰ.创建Calculation类

1. package com.qfedu.utils;
2. public class Calculation {
3. public void add(){
4. System.out.println("add()执行中……");
5. }
6. public void subtract(){
7. System.out.println("subtract()执行中……");
8. }
9. public void multiply(){
10. System.out.println("multiply()执行中……");
11. }
12. public void divide(){
13. System.out.println("divide()执行中……");
14. }
15. }

Ⅱ.创建前置通知BeforeAdvice

1. package com.qfedu.advice;
2. import java.lang.reflect.Method;
3. import org.springframework.aop.MethodBeforeAdvice;
4. public class BeforeAdvice implements MethodBeforeAdvice{
5. @Override
6. public void before(Method arg0, Object[] arg1, Object arg2) throws Throwable {
7. System.out.println("——————————方法开始执行——————————");
8. }
9. }
10. Ⅲ.创建后置通知AfterAdvice
11. package com.qfedu.advice;
12. import java.lang.reflect.Method;
13. import org.springframework.aop.AfterReturningAdvice;
14. public class AfterAdvice implements AfterReturningAdvice{
15. @Override
16. public void afterReturning(Object arg0, Method arg1, Object[] arg2, Object arg3) throws Throwable {
17. System.out.println("——————————方法执行完毕——————————");
18. }
19. }

Ⅳ.编写application.xml配置文件信息

1. <?xml version="1.0" encoding="UTF-8"?>
2. <beans xmlns="http://www.springframework.org/schema/beans"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xmlns:context="http://www.springframework.org/schema/context"
5. xmlns:aop="http://www.springframework.org/schema/aop"
6. xsi:schemaLocation="
7. http://www.springframework.org/schema/beans
8. http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
9. http://www.springframework.org/schema/context
10. http://www.springframework.org/schema/context/spring-context-3.2.xsd
11. http://www.springframework.org/schema/aop
12. http://www.springframework.org/schema/aop/spring-aop-3.2.xsd
13. ">
15. <!-- 创建原始对象 -->
16. <bean id="Calculation" class="com.qfedu.utils.Calculation"></bean>
17. <!-- 创建额外功能 -->
18. <bean id="beforeAdvice" class="com.qfedu.advice.BeforeAdvice"></bean>
19. <bean id="afterAdvice" class="com.qfedu.advice.AfterAdvice"></bean>
20. <aop:config>
21. <!-- 定义切点 -->
22. <aop:pointcut expression="execution(\* \*(..))" id="pct"/>
23. <!-- 编织 -->
24. <aop:advisor advice-ref="beforeAdvice" pointcut-ref="pct"/>
25. <aop:advisor advice-ref="afterAdvice" pointcut-ref="pct"/>
26. </aop:config>
27. </beans>

Ⅴ.书写测试类TestAop

1. package com.qfedu.test;
2. import org.springframework.context.ApplicationContext;
3. import org.springframework.context.support.ClassPathXmlApplicationContext;
4. import com.qfedu.utils.Calculation;
5. public class TestAop {
6. public static void main(String[] args) {
7. ApplicationContext context = new ClassPathXmlApplicationContext("application.xml");
8. Calculation c = (Calculation) context.getBean("Calculation");
9. c.add();
10. System.out.println();
11. c.divide();
12. System.out.println();
13. c.subtract();
14. System.out.println();
15. c.multiply();
16. }
17. }