



# 黑马程序员

## 《Java EE 企业级应用开发教程 （SSM）（第2版）》

### 教学设计

课程名称： Java EE 企业级应用开发教程

授课年级： XXXX 年级

授课学期： 第 X 学期

教师姓名： 某某老师

2021年6月



课题名称	第 8 章 Spring AOP	计划课时	4 课时
教学引入	Spring 的 AOP 模块是 Spring 框架体系中十分重要的内容，该模块一般适用于具有横切逻辑的场景，如访问控制、事务管理和性能监控等，本章将对 Spring AOP 的相关知识进行详细讲解。		
教学目标	<ul style="list-style-type: none"> <li>● 使学生了解 Spring AOP 的概念及其术语</li> <li>● 使学生熟悉 Spring AOP 的 JDK 动态代理</li> <li>● 使学生熟悉 Spring AOP 的 CGLib 动态代理</li> <li>● 使学生掌握基于 XML 的 AOP 实现</li> <li>● 使学生掌握基于注解的 AOP 实现</li> </ul>		
教学重点	<ul style="list-style-type: none"> <li>● 基于 XML 的 AOP 实现</li> <li>● 基于注解的 AOP 实现</li> </ul>		
教学难点	<ul style="list-style-type: none"> <li>● JDK 动态代理</li> <li>● CGLib 动态代理</li> </ul>		
教学方式	课堂教学以 PPT 讲授为主，并结合多媒体进行教学		
教学过程	<p style="text-align: center;"><b>第一课时</b></p> <p style="text-align: center;"><b>（Spring AOP 概述、Spring AOP 术语、JDK 动态代理）</b></p> <p>一、创设情景，导入新课</p> <p>前面的学习相信我们已经对 Spring 框架有了一个初步的了解，Spring 框架的分层架构简化了项目开发过程中的技术的复杂性，使得开发过程更加高效。而 Spring AOP 在整个 Spring 框架中十分重要，它涉及到事务的管理以及访问权限的控制等等。本节课，将对 Spring AOP 概述、Spring AOP 术语等内容进行详细讲解。</p> <p>二、新课讲解</p> <p><b>知识点 1-Spring AOP 概述</b></p> <p>教师通过多媒体演示 PPT 内容讲解 Spring AOP 概述。</p> <p>AOP 的全称是 Aspect Oriented Programming，即面向切面编程。和 OOP 不同，AOP 主张将程序中相同的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，以达到提高程序可重用性和开发效率的目的。</p> <p>AOP 的使用，使开发人员在编写业务逻辑时可以专心于核心业务，而不用过多地关注其他业务逻辑的实现，这不但提高了开发效率，而且增强了代码的可维护性。</p> <p><b>知识点 2-Spring AOP 术语</b></p> <p>教师讲解 Spring AOP 术语。</p> <p>AOP 中涉及很多术语，如切面、连接点、切入点、通知/增强处理、目标对象、织入、代理和引介等，下面针对 AOP 的常用术语进行简单介绍。</p> <p>1. 切面（Aspect）</p> <p>切面是指关注点形成的类（关注点是指类中重复的代码），通常是指封装</p>		



的、用于横向插入系统的功能类（如事务管理、日志记录等）。

## 2. 连接点（Joinpoint）

连接点是程序执行过程中某个特定的节点。

## 3. 切入点（Pointcut）

当某个连接点满足预先指定的条件时，AOP 就能够定位到这个连接点，在连接点处插入切面，该连接点也就变成了切入点。

## 4. 通知/增强处理（Advice）

通知/增强处理就是插入的切面程序代码。

## 5. 目标对象（Target）

目标对象是指被插入切面的方法。

## 6. 织入（Weaving）

将切面代码插入到目标对象上，从而生成代理对象的过程。

## 7. 代理（Proxy）

将通知应用到目标对象之后，程序动态创建的通知对象，就称为代理。

## 8. 引介（Introduction）

引介是一种特殊的通知，它为目标对象添加一些属性和方法。

### 知识点 3-JDK 动态代理

教师讲解 JDK 动态代理。

默认情况下，Spring AOP 使用 JDK 动态代理，JDK 动态代理是通过 `java.lang.reflect.Proxy` 类实现的，我们可以调用 `Proxy` 类的 `newProxyInstance()` 方法创建代理对象。

接下来，通过一个案例演示 Spring 中 JDK 动态代理的实现过程，案例具体实现步骤如下。

- (1) 在 IDEA 中创建一个名为 chapter08 的 Maven 项目，然后在项目的 `pom.xml` 文件中加载需使用到的 Spring 基础包和 Spring 的依赖包。
- (2) 在项目的 `src/main/java` 目录下，创建一个 `com.itheima.demo01` 包，在该包下创建接口 `UserDao`，在 `UserDao` 接口中编写添加和删除的方法。
- (3) 在 `com.itheima.demo01` 包中，创建 `UserDao` 接口的实现类 `UserDaoImpl`，分别实现接口中的方法。
- (4) 在 `com.itheima.demo01` 包下创建切面类 `MyAspect`，在该类中定义一个模拟权限检查的方法和一个模拟日志记录的方法，这两个方法就是切面中的通知。
- (5) 在 `com.itheima.demo01` 包下创建代理类 `MyProxy`，该类需要实现 `InvocationHandler` 接口设置代理类的调用处理程序。在代理类中，通过 `newProxyInstance()` 生成代理方法。
- (6) 在 `com.itheima.demo01` 包中，创建测试类 `JDKTest`。在该类中的 `main()` 方法中创建代理对象 `jdkProxy` 和目标对象 `userDao`，然后从代理对象 `jdkProxy` 中获得对目标对象 `userDao` 增强后的对象 `userDao1`，最后调用 `userDao1` 对象中的添加和删除方法。

三、归纳总结



教师回顾本节课所讲的内容，并通过测试题的方式引导学生解答问题并给予指导。

#### 四、布置作业

教师通过高校教辅平台 (<http://tch.ityxb.com>) 布置本节课作业以及下节课的预习作业。

### 第二课时

#### (CGLib 动态代理、基于 XML 的 AOP 实现、基于注解的 AOP 实现)

##### 一、复习巩固

教师通过上节课作业的完成情况，对学生吸收不好的知识点进行再次巩固讲解。

##### 二、通过直接导入的方式导入新课

通过上节课的学习，相信我们已经对 Spring AOP 的概述有了一个初步的了解，在实现 Spring AOP 时，需要创建一个代理对象，接下来将学习创建代理对象的另一种方式和基于 XML 的 AOP 实现等。

##### 三、新课讲解

#### 知识点 1-CGLib 动态代理

教师讲解 CGLib 动态代理。

JDK 动态代理存在缺陷，它只能为接口创建代理对象，当需要为类创建代理对象时，就需要使用 CGLib 动态代理，CGLib 动态代理不要求目标类实现接口，它采用底层的字节码技术，通过继承的方式动态创建代理对象。

接下来通过一个案例演示 CGLib 动态代理的实现过程，具体步骤如下。

- (1) 在 chapter08 项目的 src/main/java 目录下创建一个 com.itheima.demo02 包，在该包下创建目标类 UserDao，在该类中编写添加用户和删除用户的方法。
- (2) 在 com.itheima.demo02 包下创建代理类 CglibProxy，该代理类需要实现 MethodInterceptor 接口用于设置代理类的调用处理程序，并实现接口中的 intercept() 方法。
- (3) 在 com.itheima.demo02 包中创建测试类 CglibTest，在 main() 方法中首先创建代理对象 cglibProxy 和目标对象 userDao，然后从代理对象 cglibProxy 中获得增强后的目标对象 userDao1，最后调用 userDao1 对象的添加和删除方法。

#### 知识点 2-基于 XML 的 AOP 实现

教师讲解基于 XML 的 AOP 实现。

上一个小节我们介绍了 Spring AOP 的实现机制，接下来讲解 Spring AOP 的实现方法。Spring AOP 的常用实现方法有两种，分别是基于 XML 文件的实现和基于注解的实现。

对 Spring 提供的配置 Spring AOP 的 XML 元素进行讲解。

##### 1. 配置切面

在 Spring 的配置文件中，配置切面使用的是 <aop:aspect> 元素，该元素会将一个已定义好的 Spring Bean 转换成切面 Bean，因此，在使用 <aop:aspect> 元素之前，要在配置文件中先定义一个普通的 Spring Bean。

##### 2. 配置切入点

在 Spring 的配置文件中，切入点是通过 <aop:pointcut> 元素来定义的。



### 3. 配置通知

在 Spring 的配置文件中，使用<aop:aspect>元素配置了 5 种常用通知，5 种通知分别为前置通知、后置通知、环绕通知、返回通知和异常通知。

了解了如何在 XML 中配置切面、切入点和通知后，接下来通过一个案例演示如何在 Spring 中使用 XML 实现 Spring AOP，具体实现步骤如下。

- (1) 在 chapter08 项目的 pom.xml 文件中导入 AspectJ 框架的相关 JAR 包。
- (2) 在 chapter08 项目的 src/main/java 目录下创建一个 com.itheima.demo03 包，在该包下创建接口 UserDao，并在该接口中编写添加、删除、修改和查询的方法。
- (3) 在 com.itheima.demo03 包下创建 UserDao 接口的实现类 UserDaoImpl，实现 UserDao 接口中的方法。
- (4) 在 com.itheima.demo03 包下创建 XmlAdvice 类，用于定义通知。
- (5) 在 chapter08 项目的 resource 文件夹下创建 applicationContext.xml 文件，在该文件中引入 AOP 命名空间，使用<bean>元素添加 Spring AOP 的配置信息。
- (6) 在 com.itheima.demo03 包中创建测试类 TestXml。

### 知识点 3-基于注解的 AOP 实现

教师讲解基于注解的 AOP 实现。

前一小节中我们讲解了基于 XML 的 AOP 实现，但基于 XML 的 AOP 实现需要在 Spring 文件中配置大量的代码信息，不利于代码阅读与维护。为了解决此问题，Spring AOP 允许使用基于注解的方式实现 AOP，这样做可以简化 Spring 配置文件中的臃肿代码。

下面通过一个案例演示基于注解的 AOP 的实现，案例具体实现步骤如下。

- (1) 在 chapter08 项目的 src/main/java 目录下创建一个 com.itheima.demo04 包，在该包下创建 AnnoAdvice 类，用于定义通知。
- (2) 在 chapter08 项目的 resource 文件夹下创建 applicationContext-Anno.xml 文件，在该文件中引入 AOP 命名空间，使用<bean>元素添加 Spring AOP 的配置信息。
- (3) 在 com.itheima.demo04 包中创建测试类 TestAnnotation。

### 四、归纳总结

教师回顾本节课所讲的内容，并通过测试题的方式引导学生解答问题并给予指导。

### 五、布置作业

教师通过高校教辅平台 (<http://tch.ityxb.com>) 布置本节课作业以及下节课的预习作业。

### 第三、四课时（上机练习）

上机一：（考察知识点为 Spring AOP 概述、Spring AOP 术语、JDK 动态代理、CGLib 动态代理、基于 XML 的 AOP 实现、基于注解的 AOP 实现）

形式：单独完成

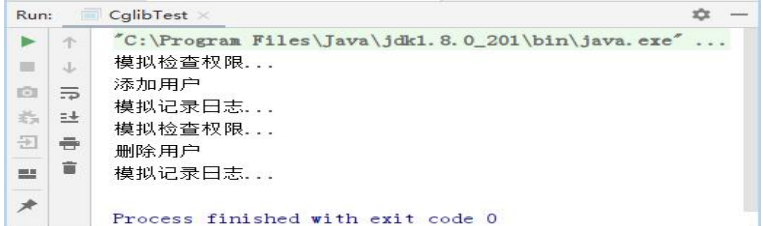
题目：

1. 代码演示 CGLib 动态代理的实现过程。

要求：

- (1) 创建项目名称为 chapter08；



	<p>(2) 创建包名为 com.itheima.demo02;</p> <p>(3) 创建代理类名称为 CglibProxy;</p> <p>(4) 创建测试类名称为 CglibTest;</p> <p>在控制台输出如下结果:</p> 
教学后记	<p>2. 代码演示基于注解的 AOP 的实现过程。</p> <p>(1) 创建项目名称为chapter08;</p> <p>(2) 创建包名为 com.itheima.demo04;</p> <p>(3) 创建配置文件名称为 applicationContext-Anno.xml;</p>