极客学院
jikexueyuan.com

AOP概述

AOP概述 — **课程概要**

- AOP的基本概念
- AOP的实现方法(一)
- AOP的实现方法(二)

AOP概述 — 学习目标

知识目标：

- 理解AOP的基本概念以及原理
- 熟悉AOP的相关基本术语
- 简单了解AOP的相关实现者

技能目标：

- 利用Spring注解方式实现AOP功能
- 利用Spring XML文件配置方式实现AOP功能

AOP的基本概念

AOP的基本概念 — **课时知识点**

- AOP的简介
- AOP的示例
- AOP的术语
- AOP的实现者

AOP的基本概念 — AOP的简介

- ## AOP的基本概念

AOP即Aspect-Oriented Programming的缩写，中文意思是面向切面（或方面）编程。它是一种思想，可在不改变程序源码的情况下为程序添加额外的功能；

- ## AOP的发展阶段

静态AOP：Aspect形式，通过特定的编译器，将实现后的Aspect编译并织入到系统的静态类中；

动态AOP：AOP的织入过程在系统运行开始之后进行，而不是预先编译到系统中；

- ## AOP的主要意图

允许通过分离应用的业务逻辑与系统级服务进行内聚性的开发。应用对象只实现业务逻辑即可，并不负责其它的系统级关注点；

- ## AOP的发展阶段

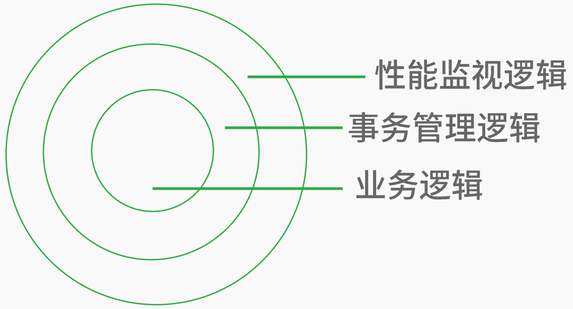
日志记录、跟踪、监控和优化，性能统计、优化，安全、权限控制，应用系统的异常捕捉及处理，事务处理，缓存，持久化，懒加载（Lazy loading），内容传递，调试，资源池，同步等等

AOP的基本概念 — AOP的示例

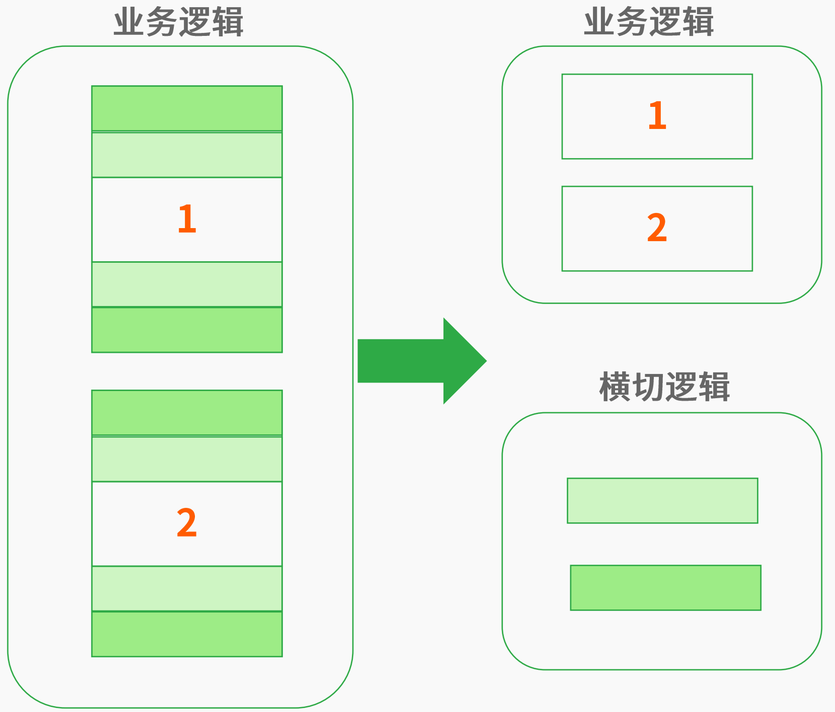
示例代码：

```
Public class ForumService {
   public void removeTopic(int topicId) {
       pmomitor.start();
       transManager.beginTransaction();
       topicDao.removeTopic(topicId);①
       transManager.commit();
       pmonitor.end();
   }
}
```

横切逻辑示意图：



横向抽取示意图：



AOP的基本概念 — **AOP的术语**

- 连接点：程序执行的某个特定位置，比如类初始化前，初始化后，方法调用前，方法调用后等等
- 切点：通过切点来定位特定的连接点
- 增强：织入到目标类连接点上的一段程序代码
- 目标对象：增强逻辑的织入目标类
- 引介：引介是一种特殊的增强，它为类添加一些属性和方法
- 织入：将增强添加对目标类的具体连接点上的过程
- 代理：一个类被AOP织入增强后，会产生一个结果类，该类融合了原类和增强逻辑的代理类
- 切面：由切点和增强组成，既包括了横切逻辑的定义，也包括了连接点的定义

AOP的基本概念 — AOP的实现者

- AspectJ

  AspectJ是目前最完善的AOP语言，对Java编程语言的进行了扩展，定义了AOP语法，能够在编译期提供横切代码的织入。AspectJ提供了两种横切实现机制，一种称为动态横切（Dynamic Crosscutting），另一种称为静态横切（Static Crosscutting）。

- AspectWerkz

  基于Java的简单、动态和轻量级的AOP框架，支持运行期或类装载期织入横切代码，它拥有一个特殊的类装载器。它与AspectJ项目已经合并，第一个发布版本是AspectJ5：扩展AspectJ语言，以基于注解的方式支持类似AspectJ的代码风格。

- JBoss AOP

  JBoss是一个开源的符合J2EE规范的应用服务器，作为J2EE规范的补充，JBoss中引入了AOP框架，为普通Java类提供了J2EE服务，而无需遵循EJB规范。JBoss通过类载入时，使用Javassist对字节码操作实现动态AOP框架。

- Spring AOP

  Spring AOP使用纯Java实现，不需要专门的编译过程，不需要特殊的类装载器，它在运行期通过代理方式向目标类织入增强代码。Spring并不尝试提供最完整的AOP实现，主要侧重于提供一种和Spring IoC容器整合的AOP实现，以解决企业级开发中常见问题。

AOP的实现方法(一)

AOP的实现方法(一) — 课时知识点

- 利用Proxy 实现AOP功能
- 利用CGLib 实现AOP功能

AOP的实现方法(一) — 利用Proxy实现AOP功能

采用Proxy类方法，基本流程为：主函数-->代理-->目标对象的方法。对于Proxy类有一个使用前提，就是目标对象必须要实现接口，否则不能使用这个方法。实现AOP功能步骤如下：

- 创建接口：StudengInterface.java
- 创建接口实现类：Student.java
- 创建代理工厂类：ProxyFactory.java

AOP的实现方法(一) — 利用Proxy实现AOP功能

利用Proxy实现AOP功能的总结如下：

- 目标对象必须实现接口
- 返回创建的代理对象
- 重写invoke()方法
- 限制条件放在invoke()方法

AOP的实现方法 (一) — **利用CGLib实现AOP功能**

CGLib(Code Generation Library)是一个开源项目,它是一个强大的，高性能，高质量的Code生成类库，它可以在运行期扩展Java类与实现Java接口。实现AOP功能步骤如下所示：

- 引入Jar文件
- 创建实体类
- 创建CGLIB代理类
- 创建入口类进行测试

AOP的实现方法(二)

AOP的实现方法(二) — **课时知识点**

- 利用Spring 注解方式实现AOP功能
- 利用Spring XML配置方式实现AOP功能

AOP的实现方法(二) — **利用Spring注解方式实现AOP功能**

利用Spring注解方式来实现前置通知，后置通知，例外通知以及环绕通知等。实现AOP功能步骤如下：

- 引入Jar文件
- 配置AOP命名空间
- 创建目标对象类
- 创建切面
- 在配置文件中配置切面
- 创建入口类进行测试

AOP的实现方法(二) — 利用Spring XML文件配置方式实现AOP功能

利用Spring XML文件配置方式实现AOP功能步骤如下：

- 引入Jar文件
- 配置AOP命名空间
- 创建目标对象类
- 创建切面
- 在配置文件中配置
- 创建入口类进行测试

AOP概述 — 课后练习

请根据以上讲解的示例，利用Spring XML配置的方式实现一个日志信息输出的AOP功能，要求如下：

- 创建一个Java Bean，例如PersonBean；
- PersonBean包含一些简单的业务逻辑，例如运动相关，包括走路，跑步，爬山，打球等；
- 在每个运动的方法前，方法中，以及方法后输出相关的日志信息；

AOP概述

本套课程中我们学习了AOP的基本概念，相关术语以及实现方法。你应当掌握了以下知识：

- AOP的基本概念
- AOP的相关术语
- AOP的实现原理
- AOP的实现方法

本课程学习的内容仅涉及到了AOP的基本概念以及相关原理，如果想继续提高，深入了解Spring AOP的相关知识，你可以继续在极客学院学习Spring AOP相关的课程。

极客学院

jikexueyuan.com

中国最大的IT职业在线教育平台