43 离开了 Spring AOP，我们如何切面编程？

更新时间：2020-08-28 10:24:23

> 不安于小成，然后足以成大器；不诱于小利，然后可以立远功。——方孝孺

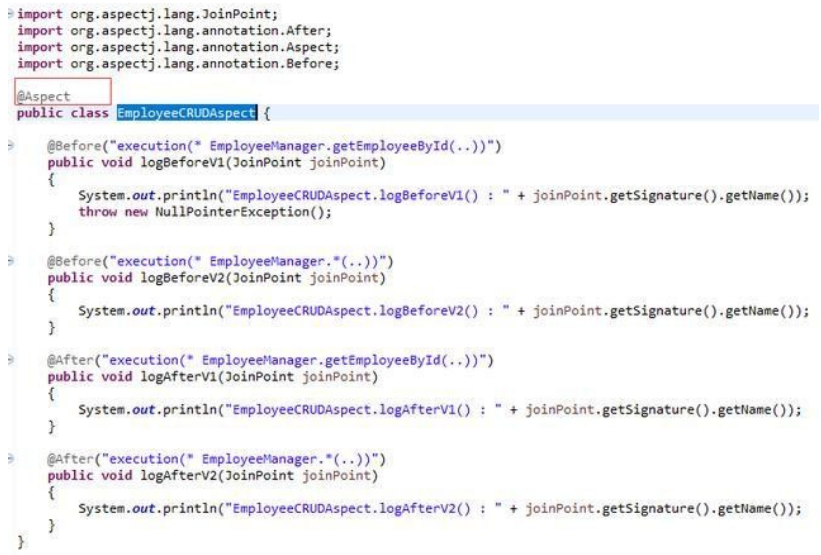
背景

Spring 框架的 AOP 机制可以让开发者把业务流程中的通用功能抽取出来，单独编写功能代码。在业务流程执行过程中，Spring 框架会根据业务流程要求，自动把独立编写的功能代码切入到流程的合适位置。Spring 提供了两种方式的 AOP 使用：

使用 XML 配置方式：

```
<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"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="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/aop
    http://www.springframework.org/schema/aop/spring-aop.xsd">
    <aop:aspectj-autoproxy />
    <context:component-scan base-package="com.davidwang456.test" />
    <bean id="loggingAspect" class="com.davidwang456.test.EmployeeCRUDAspect" />
</beans>
```

使用注解方式：



```java
import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.After;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;

@Aspect
public class EmployeeCRUDAspect {

    @Before("execution(* EmployeeManager.getEmployeeById(..))")
    public void logBeforeV1(JoinPoint joinPoint)
    {
        System.out.println("EmployeeCRUDAspect.logBeforeV1() : " + joinPoint.getSignature().getName());
        throw new NullPointerException();
    }

    @Before("execution(* EmployeeManager.*(..))")
    public void logBeforeV2(JoinPoint joinPoint)
    {
        System.out.println("EmployeeCRUDAspect.logBeforeV2() : " + joinPoint.getSignature().getName());
    }

    @After("execution(* EmployeeManager.getEmployeeById(..))")
    public void logAfterV1(JoinPoint joinPoint)
    {
        System.out.println("EmployeeCRUDAspect.logAfterV1() : " + joinPoint.getSignature().getName());
    }

    @After("execution(* EmployeeManager.*(..))")
    public void logAfterV2(JoinPoint joinPoint)
    {
        System.out.println("EmployeeCRUDAspect.logAfterV2() : " + joinPoint.getSignature().getName());
    }
}
```

这里需要注意的是：

> Spring AOP currently supports only method execution join points (advising the execution of methods on Spring beans). Field interception is not implemented, although support for field interception could be added without breaking the core Spring AOP APIs. If you need to advise field access and update join points, consider a language such as AspectJ.

Spring AOP 目前仅仅支持方法级别的切面，成员的 interception 并没有实现。

另外：Spring AOP 仅仅是集成框架，并没有参与 AOP 的具体开发。

> Spring AOP never strives to compete with AspectJ to provide a comprehensive AOP solution. We believe that both proxy-based frameworks such as Spring AOP and full-blown frameworks such as AspectJ are valuable and that they are complementary, rather than in competition. Spring seamlessly integrates Spring AOP and IoC with AspectJ, to enable all uses of AOP within a consistent Spring-based application architecture. This integration does not affect the Spring AOP API or the AOP Alliance API. Spring AOP remains backward-compatible.

故如果想利用 AOP 的更多功能，或者在不使用 Spring 的框架中使用 AOP 的功能，该怎么办呢？

AspectJ 简介

我们大都知道 Spring AOP 集成了 AspectJ，可是到底什么是 AspectJ 呢

官方描述

- a seamless aspect-oriented extension to the Javatm programming language（可以和 Java 编程语言无缝结合的一个面向切面编程的可扩展框架）；
- Java platform compatible （兼容 Java 平台）；
- easy to learn and use （易学易用）。

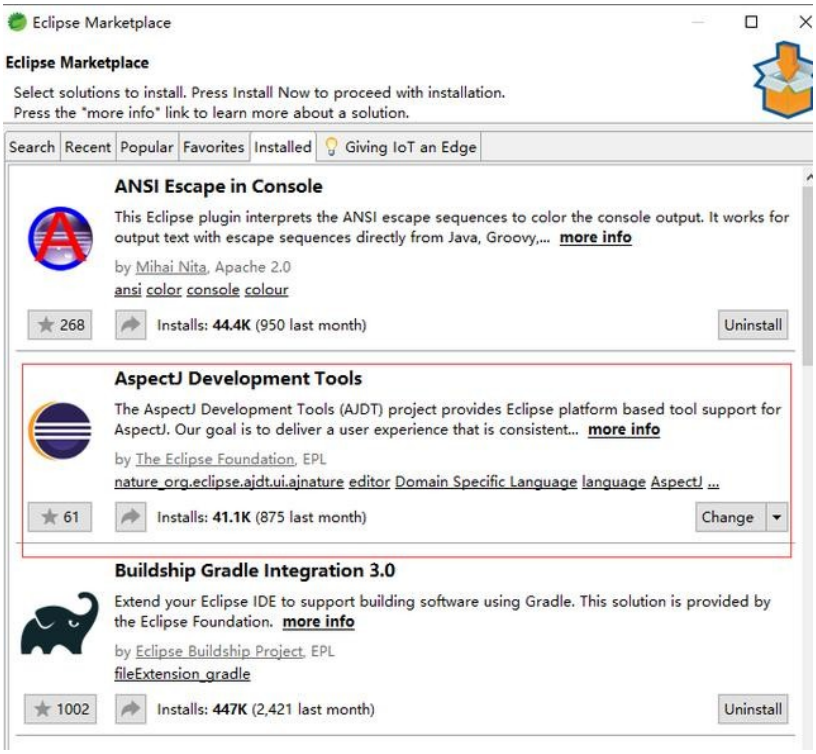
Aspect 能做什么呢？

> clean modularization of crosscutting concerns, such as error checking and handling, synchronization, context-sensitive behavior, performance optimizations, monitoring and logging, debugging support, and multi-object protocols

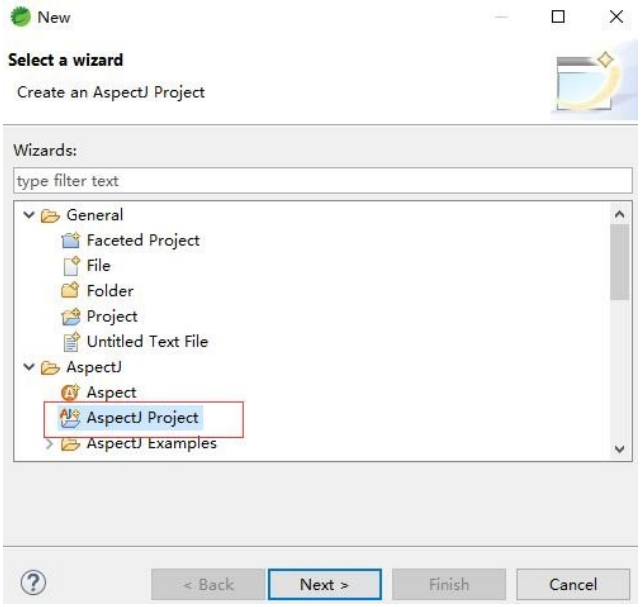
清除横切关注点的模块化，如错误检查和处理、同步、上下文敏感的行为、性能优化、监视和日志记录、调试支持和多对象协议。

AspectJ 的使用实例

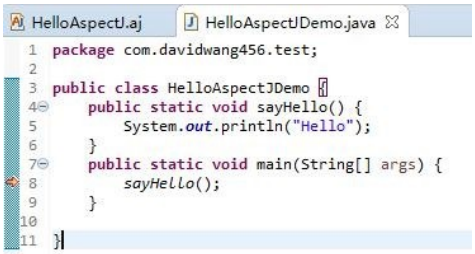
1. Eclipse Marketplace 安装插件 AJDT：



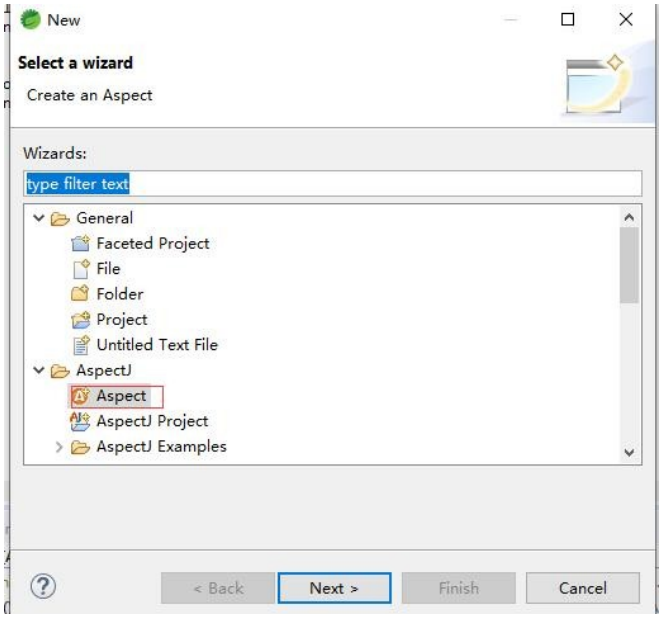
2. 创建 Aspect 工程：



3. 创建 Aspect 测试类：



创建一个切面 Aspect 文件：



.aj 文件：

运行 HelloAspectJDemo 的 Java 程序，结果为：

> Before call sayHello Hello
>
> After call sayHello

总结

学习 Spring AOP，AspectJ 的使用是个绕不过去的坎。少了这一部分，一些 Spring AOP 的源码或者原理总少了点什么，不太容易懂。这篇文章通过抛开 Spring AOP 框架单独来使用 Aspect 来加深大家对 AOP 的认识。

}

42 离开Spring AOP，我们如何实现AOP功能？

44 Spring控制器Controller如何设置AOP？