Spring 框架的一个关键组件是**面向方（切）面的编程**(AOP)框架。OOP 中，关键单元模块度是类，而在 AOP 中单元模块度是方面。面向方面的编程把程序逻辑分解成不同的部分(关注点)。跨一个应用程序的多个点的功能被称为**横切关注点**，在概念上独立于应用程序的业务逻辑。Spring AOP 模块提供拦截器来拦截一个应用程序，例如，当执行一个方法时，你可以在方法执行之前或之后添加额外的功能。底层用 aspectj

AOP 术语

项	描述
Aspect	一个模块具有一组提供横切需求的 APIs。例如，一个日志模块为了记录日志将被 AOP 方面调用。应用程序可以拥有任意数量的方面，这取决于需求。
Join point	在你的应用程序中它代表一个点，你可以在插件 AOP 方面。你也能说，它是在实际的应用程序中，其中一个操作将使用 Spring AOP 框架。
Advice	这是实际行动之前或之后执行的方法。这是在程序执行期间通过 Spring AOP 框架实际被调用的代码。
Pointcut	这是一组一个或多个连接点，通知应该被执行。你可以使用表达式或模式指定切入点。
Introduction	引用允许你添加新方法或属性到现有的类中。
Target object	被一个或者多个方面所通知的对象，这个对象永远是一个被代理对象。也称为被通知对象。
Weaving	Weaving 把方面连接到其它的应用程序类型或者对象上，并创建一个被通知的对象。这些可以在编译时，类加载时和运行时完成。

Aspect（切面）： Aspect 声明类似于 Java 中的类声明，在 Aspect 中会包含着Pointcut 以及相应的 Advice。

Joint point（连接点）：表示在程序中明确定义的点（对象），典型的包括方法调用，对类成员的访问以及异常处理程序块的执行等等，它自身还可以嵌套其它 joint point。

Pointcut（切点）：提供一组规则来匹配一组joinpoint，这些 joint point 或是通过逻辑关系组合起来，或是通过通配、正则表达式等方式集中起来，给joint point添加Advice。

Advice（增强）：Advice 定义了在 Pointcut 里面定义的程序点具体要做的操作，它通过 before、after 和 around 来区别是在每个 joint point 之前、之后还是代替执行的代码。

Target（目标对象）：织入 Advice 的目标对象.。

Weaving（织入）：将 Aspect 和其他对象连接起来，并创建 Adviced object 的过程

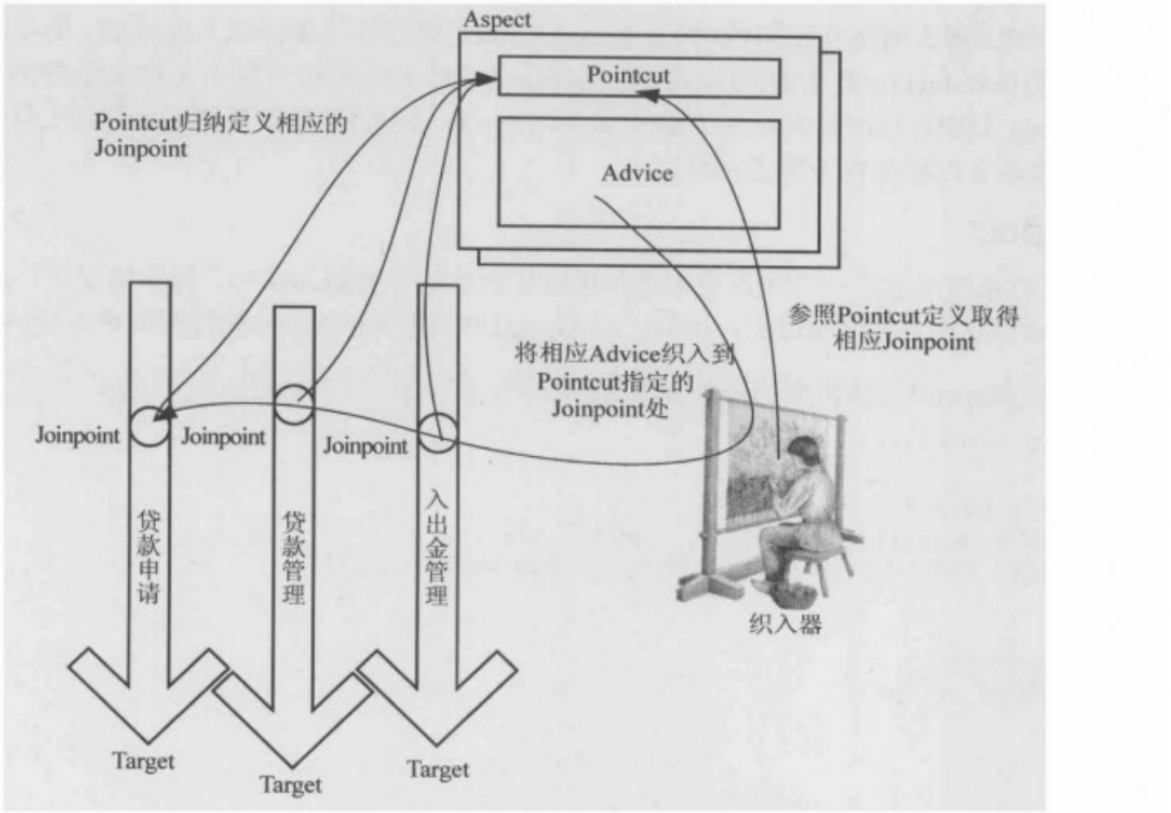

图7-9　AOP各个概念所处的场景

通知的类型

通知	描述	需要实现的接口
前置通知	方法执行之前，执行通知。	org.springframework.aop.MethodBeforeAdvice
后置通知	方法执行之后，不考虑其结果，执行通知。	org.springframework.aop.AfterReturningAdvice
返回后通知	方法执行之后，只有**在方法成功**完成时，才能执行通知。	
异常通知	方法执行之后，在方法退出抛出异常时，才能执行通知。	org.springframework.aop.ThrowsAdvice
环绕通知	在建议方法调用之前和之后，执行通知。	org.springframework.a

		op.MethodInterceptor

Expression表达式