这一章主要包括:

理解横切关注点

使用AOP模块化横切点

理解AOP语言

回顾你的上一个项目,然后将它与你前几年的项目进行对比,有什么不同?答案是:更复杂了.如今的软件系统都很复杂，在未来的几年，所有迹象都表明，软件复杂性会增加的更快。那么软件开发者应该如何去管理这些复杂性？

如果复杂性能够作为一个问题，那么模块化就是其解决方案。通过将问题分成多个可管理的片，你可以更好地将你的精力用于于实现每一个管理片。当你面临一个非常复杂的软件需求时，你很有可能需要将这个需求分成多个部分，例如业务功能，数据访问和展现逻辑部分。我们将这些功能模块称为系统的关注点。在一个银行系统中，你可能关心用户管理，账户管理和款项管理。你可能实现了数据访问模块和web模块。我们称这些模块为核心关注点，因为他们组成了系统最核心的功能。另外一些关注点，例如安全，日志记录，资源池，缓存，性能监控，兵法控制和事务管理等覆盖了（或者是横切）许多其他的模块。我们称这些功能为横切关注点。

对于核心关注点，OOP--如今被使用的最为流行的方法思想，已经可以取得很好的效果。你直接可以看到一个例如LoanManagementService这样的类来实现业务逻辑，一个AccoutRepository类来实现数据访问模块。但是如何处理横切关注点？如果你实现一个安全，认证或者性能监控这样一个确定的模块，你确定是一个良好的方案吗？仅仅使用OOP是无法做到这一点的。相反，OOP强迫你将这些功能融合在其他功能当中。而这正是面向切面编程所擅长的地方。

通过引入一种全新的模块单元-切面，AOP可以分离横切关注点。每个切面会聚焦于一个特定的横向功能。核心类不再承载横向关注点，一个切面的weaver会通过一个称为织入的过程来将核心类和横向切面连接起来，从而最终构造出一个系统。因此，AOP能够帮助我们更好地设计，实现和维护一个应用。

在这一章中了，我们将会讨论AOP的基本功能，AOP能够解决的问题和为什么你需要了解AOP。在本书其他章节，我们会讨论AspectJ-一种AOP的具体实现。让我们从讨论不用AOP来管理关注点开始，这将会帮助你理解你为什么需要使用AOP。

* 1. 没有AOP的世界

单独地使用OOP你将会如何实现横向的关注点？很显然，正如图1.1示意的那样，你会在在每个模块内需要横切逻辑的地方增加代码。

这张图展示了在一个系统实现中，核心关注点和横向关注点模块的不同。让我们通过一个代码片段来表达出相同的思想。以一种简单的方式来考虑一个压缩了一些业务逻辑的有代表性的实现类。如列表1.1所展示的那样，一个系统由许多这样的类所组成。

虽然细节不同，上述代码展示了一个许多开发者都会面临的问题：在设计时，概念上的独立者存在于于多个关注点上，但是实现逻辑将他们缠绕在一起。这样的实现，通过让一个类负责实现和核心关注点和横向关注点，破坏了单一职责原则。如果你需要更改与横向关注点相关的调用代码，你必须更改包含这个调用的每一个类。这样做也破坏了开放/封闭原则——对扩展开放，对修改封闭。总体的后果就是，你需要很高的成本来实现产品的功能和修复bug。

在传统的实现当中，核心关注点和横向关注点会在每一个模块中缠绕在一起。更严重的是，每个横向关注点散布在很多模块中。这些代码存在的缠绕性和分散性是横向关注点实现的常规的标志。让我们详细的讨论他们。