Spring: AOP

Peter Alagna Jr.

Cross-cutting Concerns

- Code that repeats in many classes across one project.
  - Tracing (logging in general).
  - Exception Handling (logging inside a catch block).
  - Transactions (commit, rollback).
  - Security (authentication, authorization).
- Can’t be centralized with OOP only…

AOP IS NEEDED

Aspect Oriented Programming (AOP)

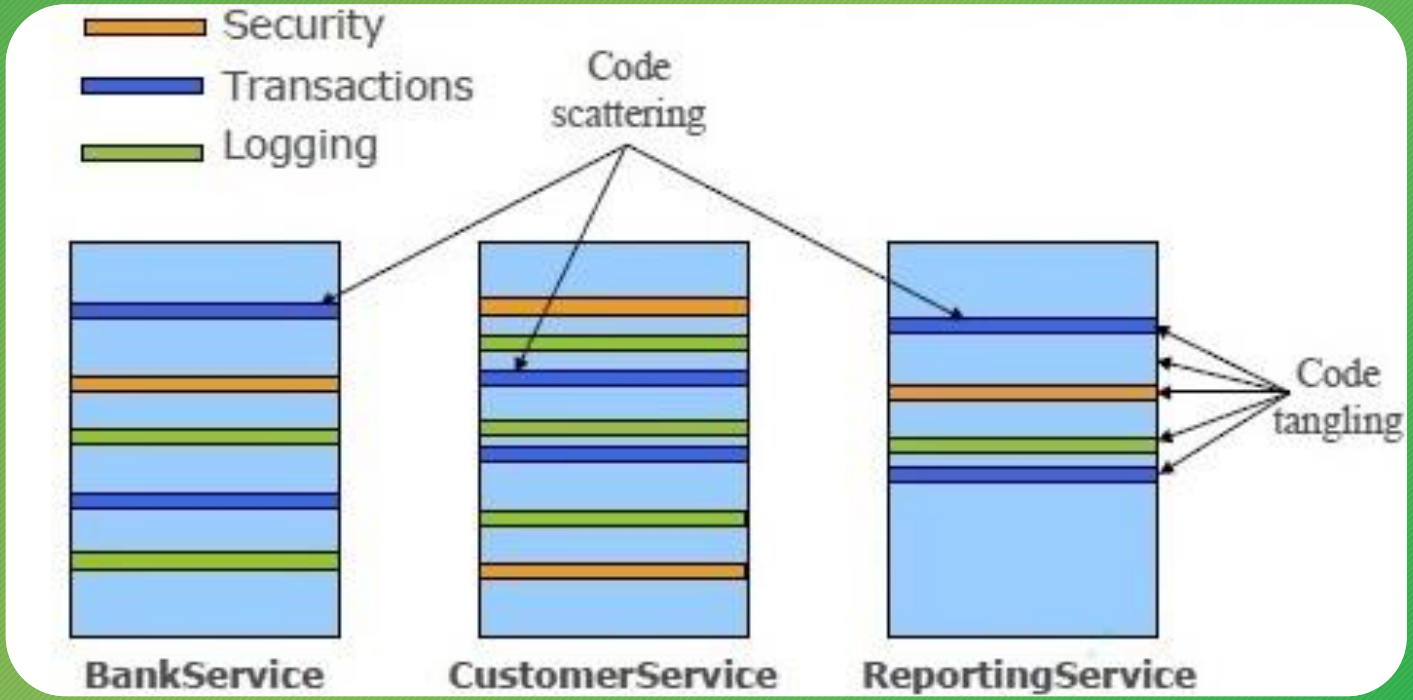
AOP's main objective is to **simplify** your code by eliminating **cross-cutting concerns** (boiler-plate code).

- You can centralize cross-cutting concerns using:
  - Exception Aspect.
  - Logging Aspect.
  - Transaction Aspect.
  - Security Aspect.
- In the future if this behavior needs to be modified, you need to go only to one place!

REVATURE

How does it work?

- Eliminates **code tangling** and **code scattering**.



REVATURE

Developing with AOP

```
<aop:aspectj-autoproxy />
```

- **Implement** business logic.
- **Write** aspects for cross-cutting concerns.
  - Aspects can be defined as boiler-plate code written in a centralized and more modular way.
- Make your code more **modular**.
- Use Spring AOP or AspectJ to **apply** aspects in your app.

JoinPoint

Point in the **control flow** of a program.

- **Advices** can receive a **JoinPoint** as a parameter (**injected**).
- With JoinPoints you can access things like:
  - Executing method **name**.
  - Executing method **parameters**.
  - Execution method **class**.
  - Others.

Advice

Action taken by a specific **Aspect** at a particular **JoinPoint**.

- @Before.
- @After.
- @AfterReturning.
- @AfterThrowing.
- @Around.
  - Adds **behavior** (**powerful**).
  - Receives **ProceedingJoinPoint** (you intersect code and let it continue afterwards).
  - pjp.proceed() after adding behavior.

Pointcut

A **predicate** that matches **JoinPoints.**

- For example, the execution of a method with a specific **return type**, **name** and/or **parameter type.**

```
@Before("execution(* print* (..))")
```

REVATURE

Spring AOP vs AspectJ

- AOP is **simpler** to use and has **less** features.
- AOP uses **same syntax** as AspectJ.
- AspectJ has **less overhead** and is more powerful.
  - Supports load-time weaving configuration/usage.

Investigate: **Load-time** and **Compile-time** **Weaving**

Materials

- Spring AOP: https://docs.spring.io/spring/docs/current/spring-framework-reference/html/aop.html

REVATURE