Spring – AOP

# Intro

Aspect-Oriented Programming (AOP) complements OOP by providing another way of thinking about program structure. The key unit of modularity in OOP is the class, whereas in AOP the unit of modularity is the **aspect**. *Aspect anable the modularization of concerns such as transaction management that cut across multiple types and objects.* (Such concerns are often termed crosscutting concerns in AOP literature.).

# AOP concepts

* ***Aspect***: a modularization of a concern that cuts across multiple classes (e.g. Logging and Security concerns). In Spring AOP, aspects are implemented using regular classes (the **schema-based approach(xml)**) or regular classes annotated with the @Aspect annotation (the **@AspectJ style**)
* ***Join point***: a point during the execution of a program, such as the execution of a method or the handling of an exception. In Spring AOP, a join point always represents a method execution.
* ***Advice***: action taken by an aspect at a particular join point.
* ***Pointcut***: a predicate that matches join points.
* Introduction:
* Target object:
* AOP proxy:
* Weaving:

## Declaring Aspect

*Remember to annotate class with @Component when declaring an annotated @Aspect class.*

## Declaring Pointcut

## Declaring Advice

Advice is associated with a pointcut expression, and runs before, after, or around method executions matched by the pointcut.

# Resources

<http://www.eclipse.org/aspectj/doc/released/progguide/index.html>

<http://www.eclipse.org/aspectj/doc/released/adk15notebook/index.html>