Aspect-Oriented Programming (AOP)
Spring AOP Support

AOP Terminology

- **Aspect**: module of code for a cross-cutting concern (logging, security, …)
- **Advice**: What action is taken and when it should be applied
- **Join Point**: When to apply code during program execution
- **Pointcut**: A predicate expression for where advice should be applied

Advice Types

- **Before advice**: run before the method
- **After finally advice**: run after the method (finally)
- **After returning advice**: run after the method (success execution)
- **After throwing advice**: run after method (if exception thrown)
- **Around advice**: run before and after method

Weaving

- Connecting aspects to target objects to create an advised object
- Different types of weaving
  - Compile-time, load-time or run-time
- Regarding performance: run-time weaving is the slowest

AOP Frameworks

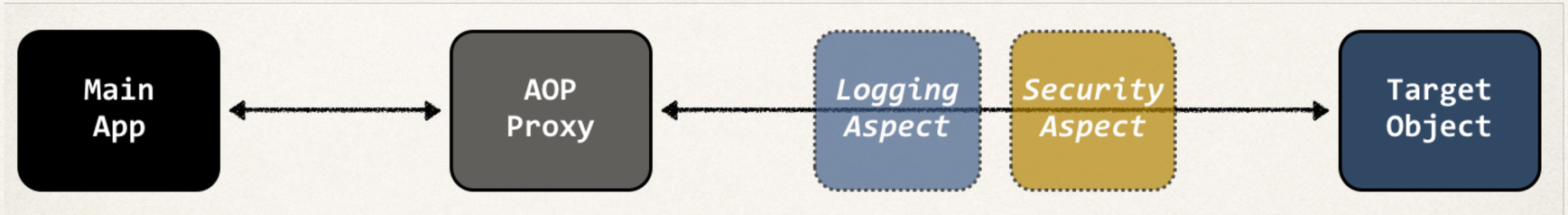
- Two leading AOP Frameworks for Java

Spring AOP

AspectJ

Spring AOP Support

- Spring provides AOP support
- Key component of Spring
  - Security, transactions, caching etc
- Uses run-time weaving of aspects



AspectJ

- Original AOP framework, released in 2001
  - www.eclipse.org/aspectj
- Provides complete support for AOP
- Rich support for
  - join points: method-level, constructor, field
  - code weaving: compile-time, post compile-time and load-time

Spring AOP Comparison

Advantages

- Simpler to use than AspectJ
- Uses Proxy pattern
- Can migrate to AspectJ when using @Aspect annotation

Disadvantages

- Only supports method-level join points
- Can only apply aspects to beans created by Spring app context
- Minor performance cost for aspect execution (run-time weaving)

AspectJ Comparison

Advantages

- Support all join points
- Works with any POJO, not just beans from app context
- Faster performance compared to Spring AOP
- Complete AOP support

Disadvantages

- Compile-time weaving requires extra compilation step
- AspectJ pointcut syntax can become complex

Comparing Spring AOP and AspectJ

- Spring AOP only supports
  - Method-level join points
  - Run-time code weaving (slower than AspectJ)

- AspectJ supports
  - join points: method-level, constructor, field
  - weaving: compile-time, post compile-time and load-time

Comparing Spring AOP and AspectJ

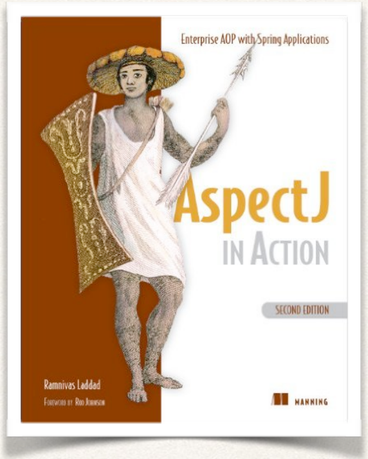
- Spring AOP is a light implementation of AOP
- Solves common problems in enterprise applications

- My recommendation
  - Start with Spring AOP … easy to get started with
  - If you have complex requirements then move to AspectJ

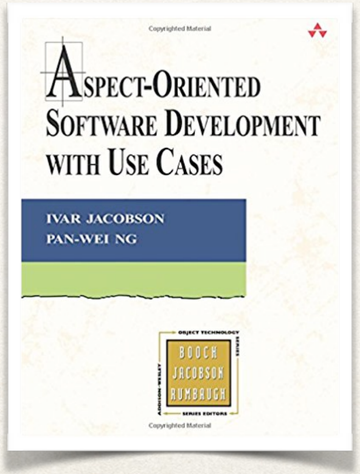
Additional Resources

- Spring Reference Manual: www.spring.io

- *AspectJ in Action*
  - by Raminvas Laddad



- *Aspect-Oriented Development with Use Cases*
  - by Ivar Jacobson and Pan-Wei Ng



Our Spring AOP Roadmap

Step-By-Step

- Create **Aspects**
- Develop **Advices**
  - *Before, After returning, After throwing,*
  - *After finally, Around*
- Create **Pointcut** expressions
- Apply it to our big CRM project (Spring MVC + Hibernate)