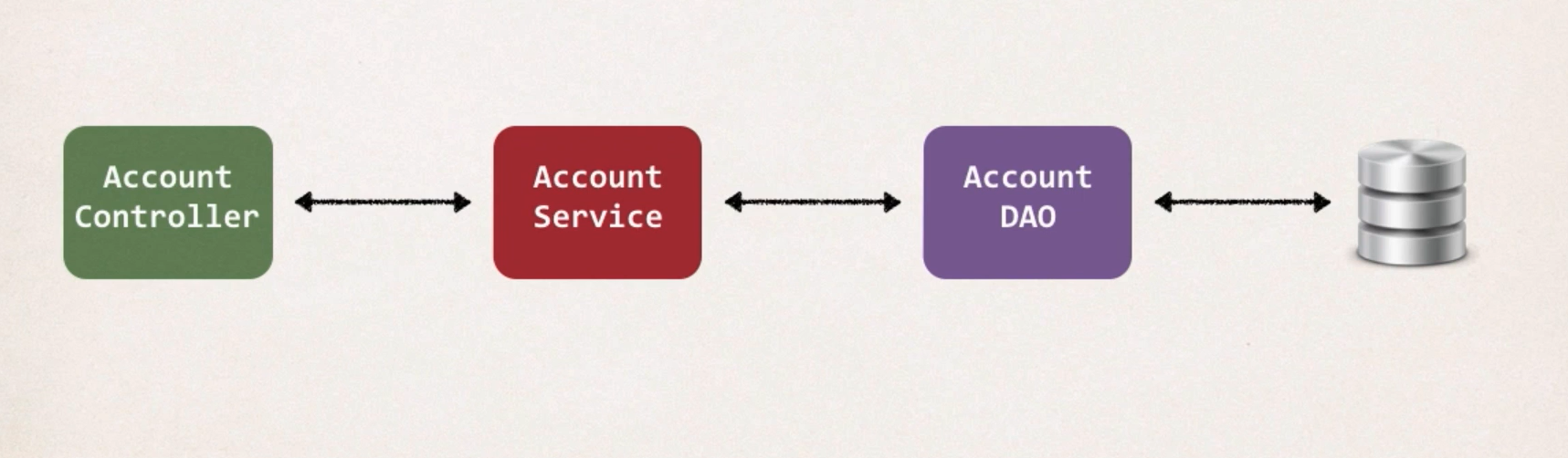
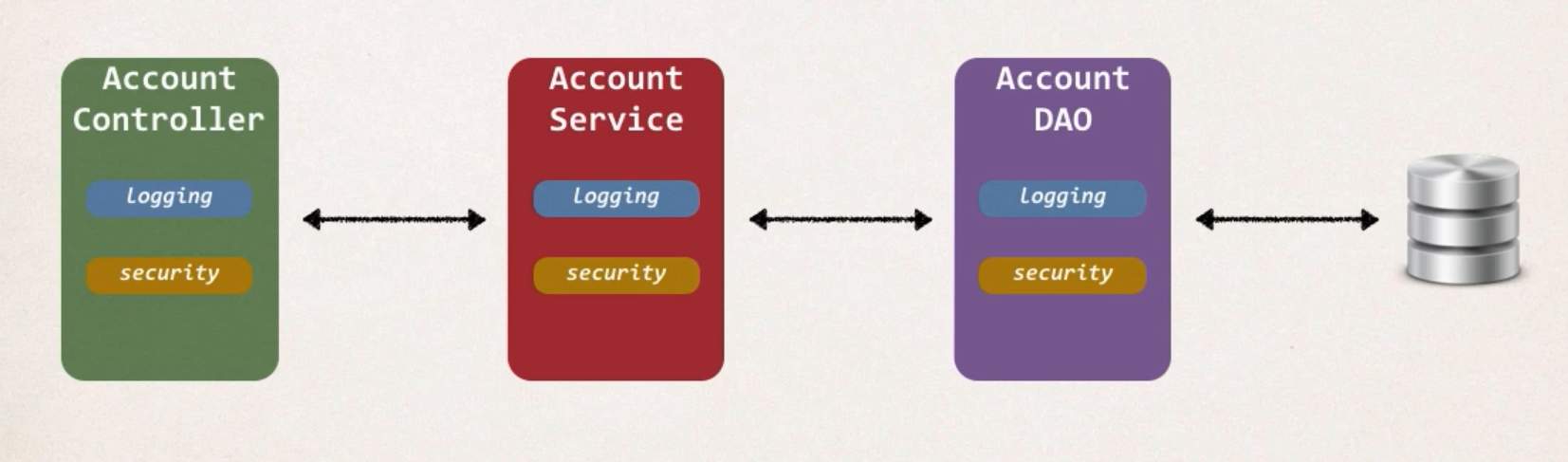
ASPECT ORIENTED PROGRAMMING

Application Architecture:





Example Requirements:

* Logging
* Security
* Audit Logging – Who, What, When Where
* Exception Handling
* API Management

Problems:

1. Code Tangling:
   1. For a given method, logging and security code are tangled in
2. Code Scattering:
   1. If we need to change logging or security, we need to change in ALL classes.

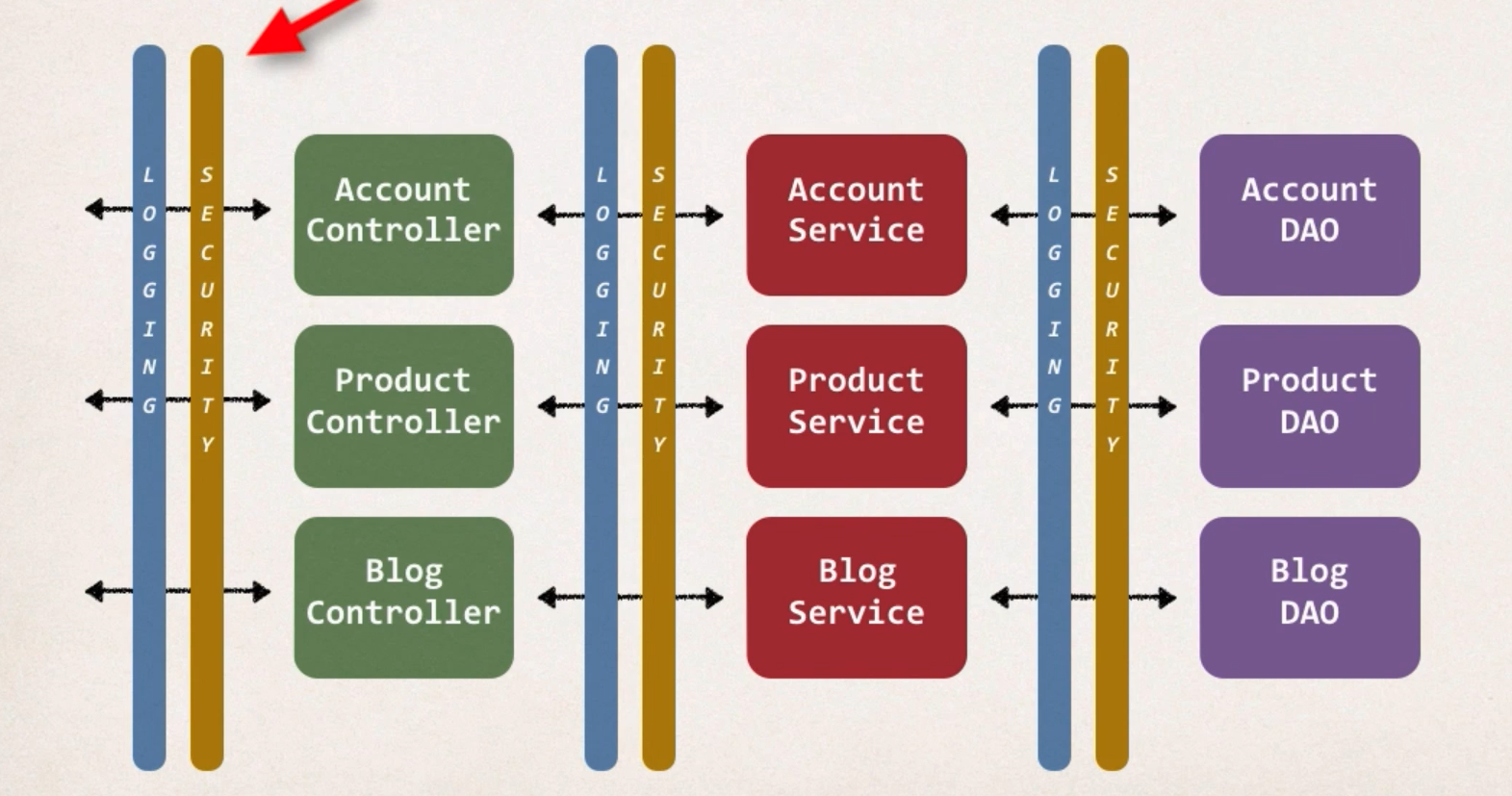
Solution:

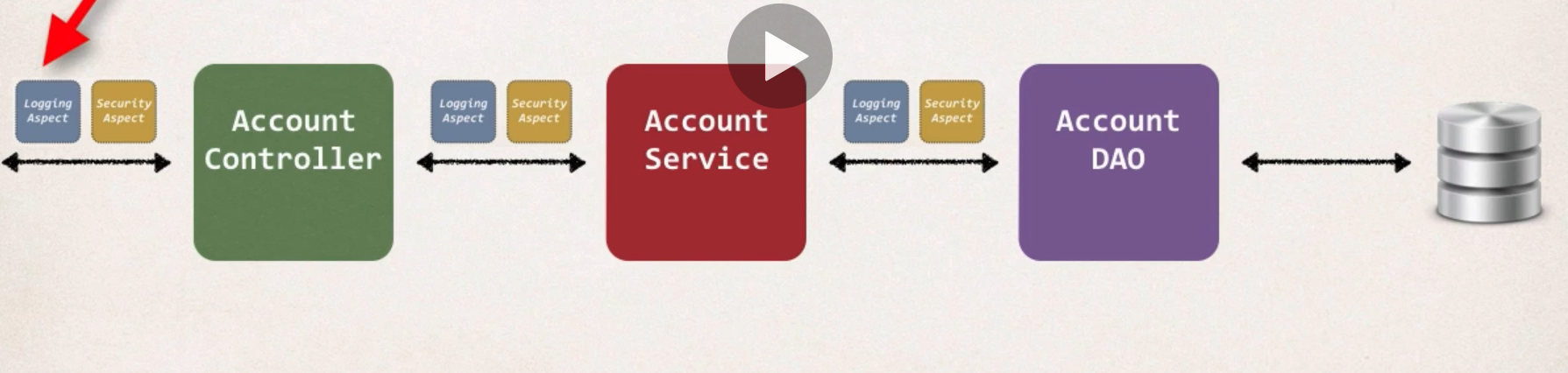
1. Inheritance
   1. But all classes have to extend
   2. Java doesn’t support multiple inheritance
2. Delegation
   1. Adding new feature would need us touch all the classes

AOP:

1. Programming technique based on concept of an Aspect
2. Aspect encapsulates **CROSS-CUTTING CONCERNS**
3. Concern means logic/functionality

Architect Diagram





Benefits of AOP

1. Code for Aspect is defined in single class
2. Business code is cleaner
3. Configurable

Advantages:

1. Reusable Modules
2. Resolves Code tangling and code scatter
3. Applied selectively based on configuration

Disadvantages:

1. Too many aspects and app flow are hard to fine
2. Minor performance cost – Run time weaving

AOP and Spring support

Terminology

1. **Aspect**: Module of code for a cross-cutting concern
2. **Advice**: What action is taken and when it should be applied
3. **Join Point**: When to apply code during program execution
4. **Pointcut**: A predicate expression for where advice should be applied

Advice Types

1. Before advice
2. After finally advice
3. After returning advice
4. After throwing advice
5. Around advice

Weaving:

1. Connecting aspects to target objects to create an advised object
2. Compile Time, Load time and run time weaving
3. Run time weaving is the slowest

AOP Framework

1. Spring AOP: run time weaving
2. AspectJ: complete support

Before Advice

1. Used for Logging, security, transactions
2. Used for Audit Logging
3. Used for API Management

NOTE: Spring uses AOP during @Transactional