# Why Aspect-oriented Programming (AOP)?

Eberhard Wolff
http://ewolff.com
eberhard.wolff@gmail.com

pluralsight
hardcore developer training

# Why AOP?

- AOP is a very important foundation of Spring
- …and also Java EE

- AOP is used to implement enterprise features
- E.g. transactions, security …
- Configurable middleware

- And to simplify code

# Simplify Code Using AOP

```
public void doSomething() {
  final String METHODNAME = "doSomething";
  logger.trace("entering " + CLASSNAME + "." + METHODNAME);
  TransactionStatus tx = transactionManager.getTransaction(
    new DefaultTransactionDefinition());
   try {
     // Business Logic
   } catch (RuntimeException ex) {
     logger.error("exception in "+CLASSNAME+"."+METHODNAME, ex);
     tx.setRollbackOnly();
     throw ex;
   } finally {
     transactionManager.commit(tx);
     logger.trace("exiting " + CLASSNAME + "." + METHODNAME);
   }
}
```

# Simplify Code Using AOP

```
public void doSomething() {
  final String METHODNAME = "doSomething";
  logger.trace("entering " + CLASSNAME + "." + METHODNAME);
  TransactionStatus tx = transactionManager.getTransaction(
    new DefaultTransactionDefinition());
  try {
    // Business Logic
  } catch (RuntimeException ex) {
    logger.error("exception in "+CLASSNAME+"."+METHODNAME, ex);
    tx.setRollbackOnly();
    throw ex;
  } finally {
    transactionManager.commit(tx);
    logger.trace("exiting " + CLASSNAME + "." + METHODNAME);
  }
}
```

Exception Handling

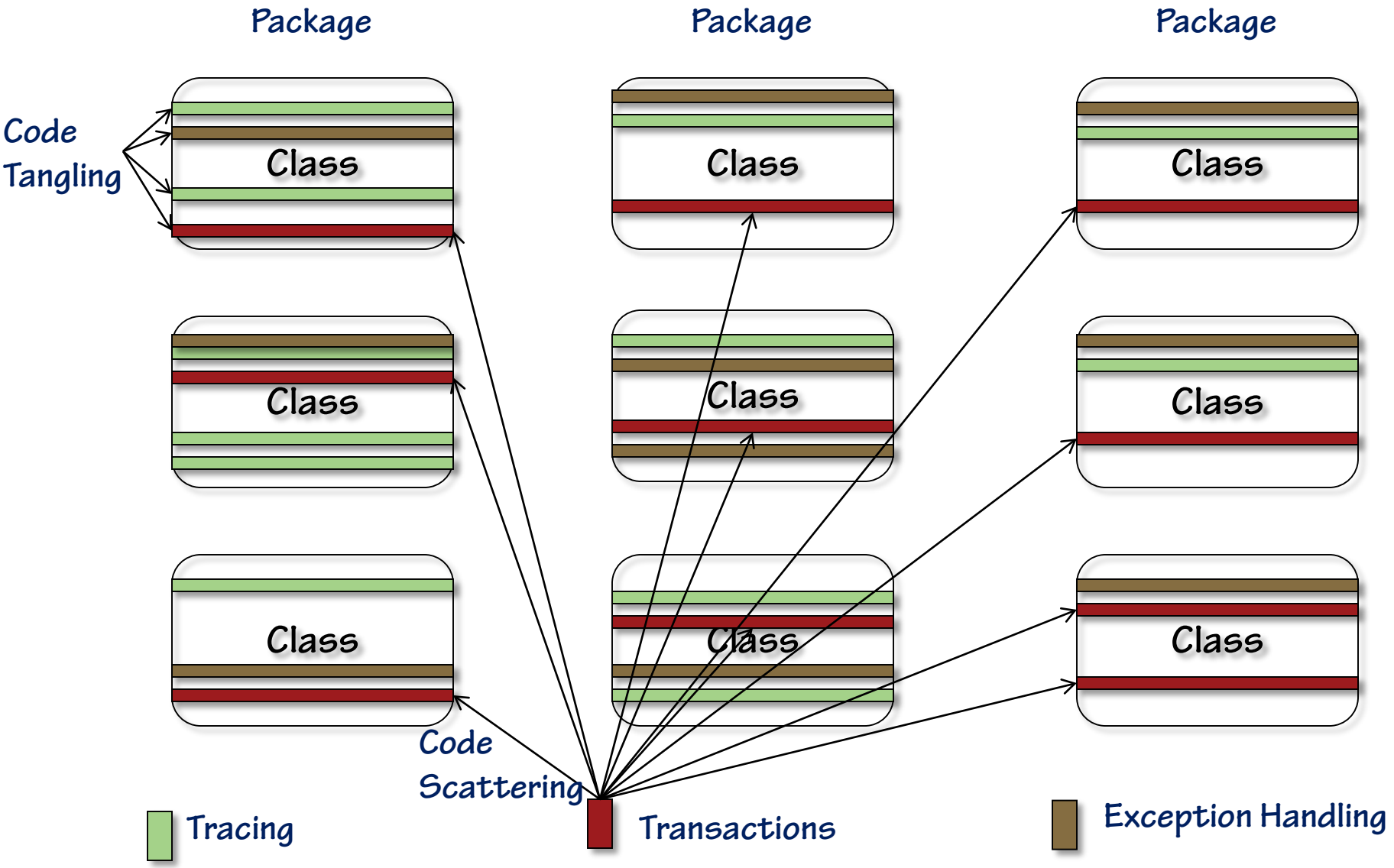
Tracing

Transactions

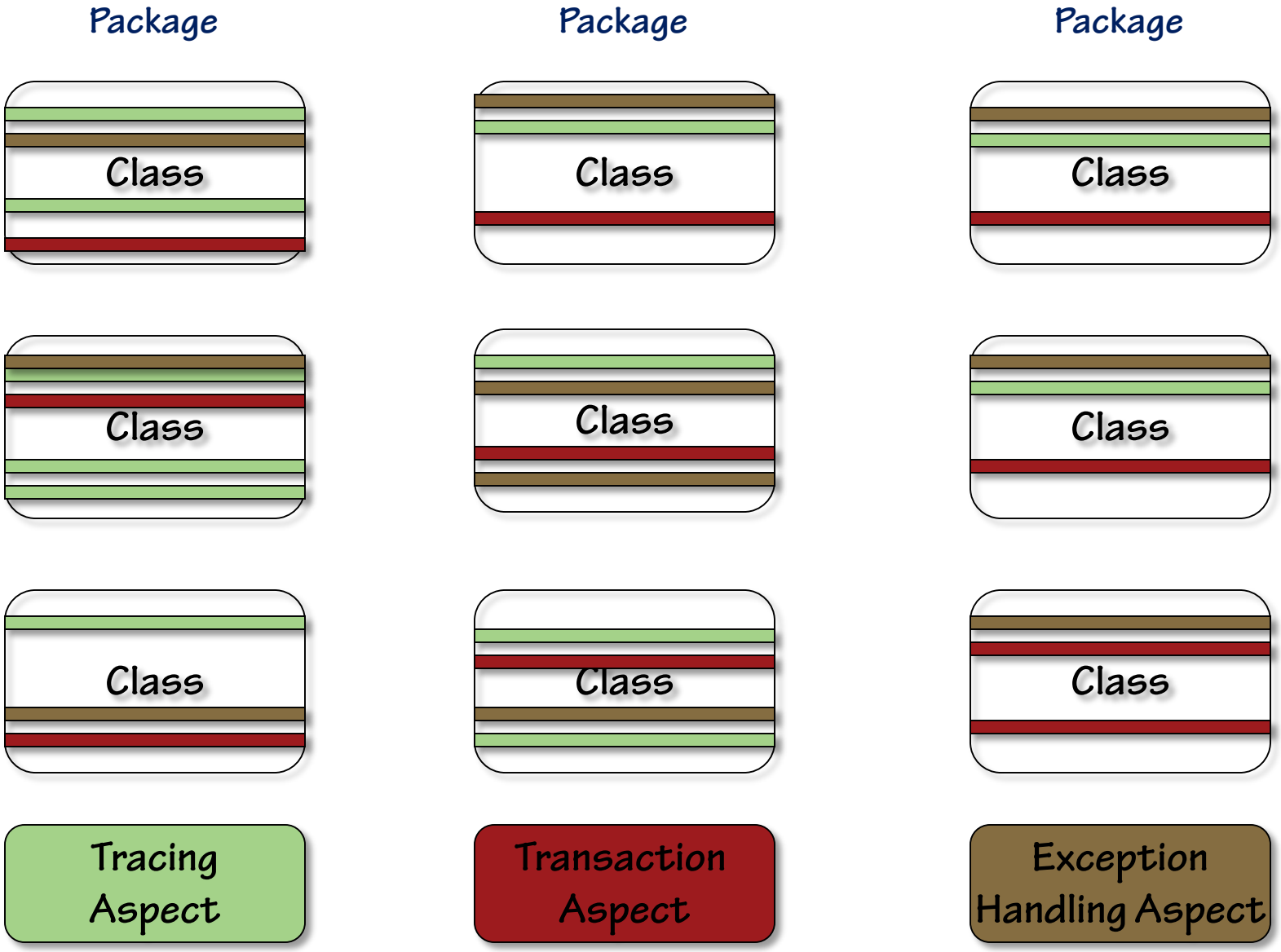
# Simplify Code Using AOP

```
public void doSomething() {
  final String METHODNAME = "doSomething";
  logger.trace("entering " + CLASSNAME + "." + METHODNAME);
  TransactionStatus tx = transactionManager.getTransaction(
    new DefaultTransactionDefinition());
  try {
    // Business Logic
  } catch (RuntimeException ex) {
    logger.error("exception in "+CLASSNAME+"."+METHODNAME, ex);
    tx.setRollbackOnly();
    throw ex;
  } finally {
    transactionManager.commit(tx);
    logger.trace("exiting " + CLASSNAME + "." + METHODNAME);
  }
}
```

# How AOP Works



# How AOP Works



# Cross-Cutting Concerns

- Tracing, Exception Handling and Transactions are cross-cutting concerns
- i.e. a lot of classes and methods must implement them

- Can't be implemented in a single place
- …if you use object-oriented programming only

- Aspect-oriented programming allows centralized implementation of cross-cutting concerns

# Developing with AOP

- Implement business logic

- Write Aspects for Cross Cutting Concerns
- …or use Spring's Aspect Library
- "Compose" your own infrastructure

- Use Spring AOP or AspectJ to weave aspects into the application
- Most commonly used AOP implementations
- Most powerful Java AOP implementations

# Summary

- **Without AOP cross-cutting concerns are implemented in many methods**

- **Mostly technical**
  - Tracing
  - Exception handling
  - Transactions
  - Security

- **Result**
  - Code tangling: Multiple concerns in each piece of code
  - Code scattering:  Aspects are not implemented in one place

- **Aspect-oriented programming resolves these issues**