Mindtree

Welcome to possible

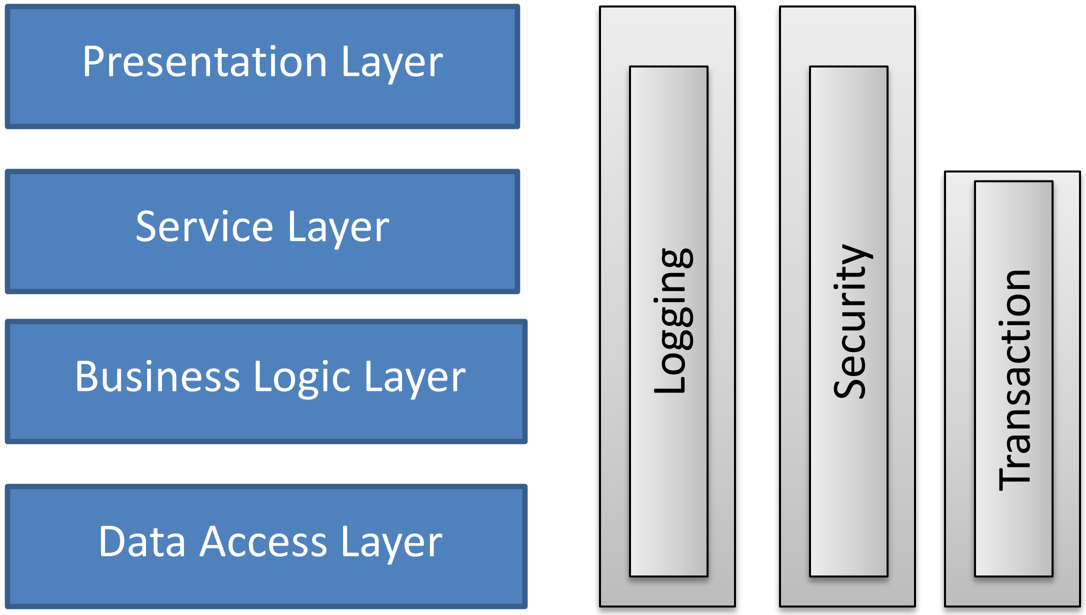
Introduction to AOP

Objectives

- Define Aspect Oriented Programming with spring
- How code tangling and code scattering happens with cross cutting concerns?

Aspect Oriented Programming

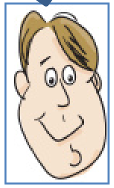
- Cross-cutting concerns are aspects of a program which affect other concerns.
- Cross-cutting concerns often cannot be cleanly decomposed from the rest of the system in both the design and implementation, and can result in either *scattering* (code duplication), *tangling* (significant dependencies between systems), or both.
- Examples of some cross cutting concerns (Logging, Security, Transaction)

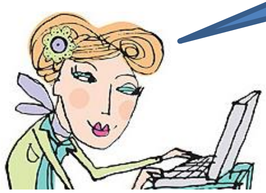


Aspect Oriented Programming

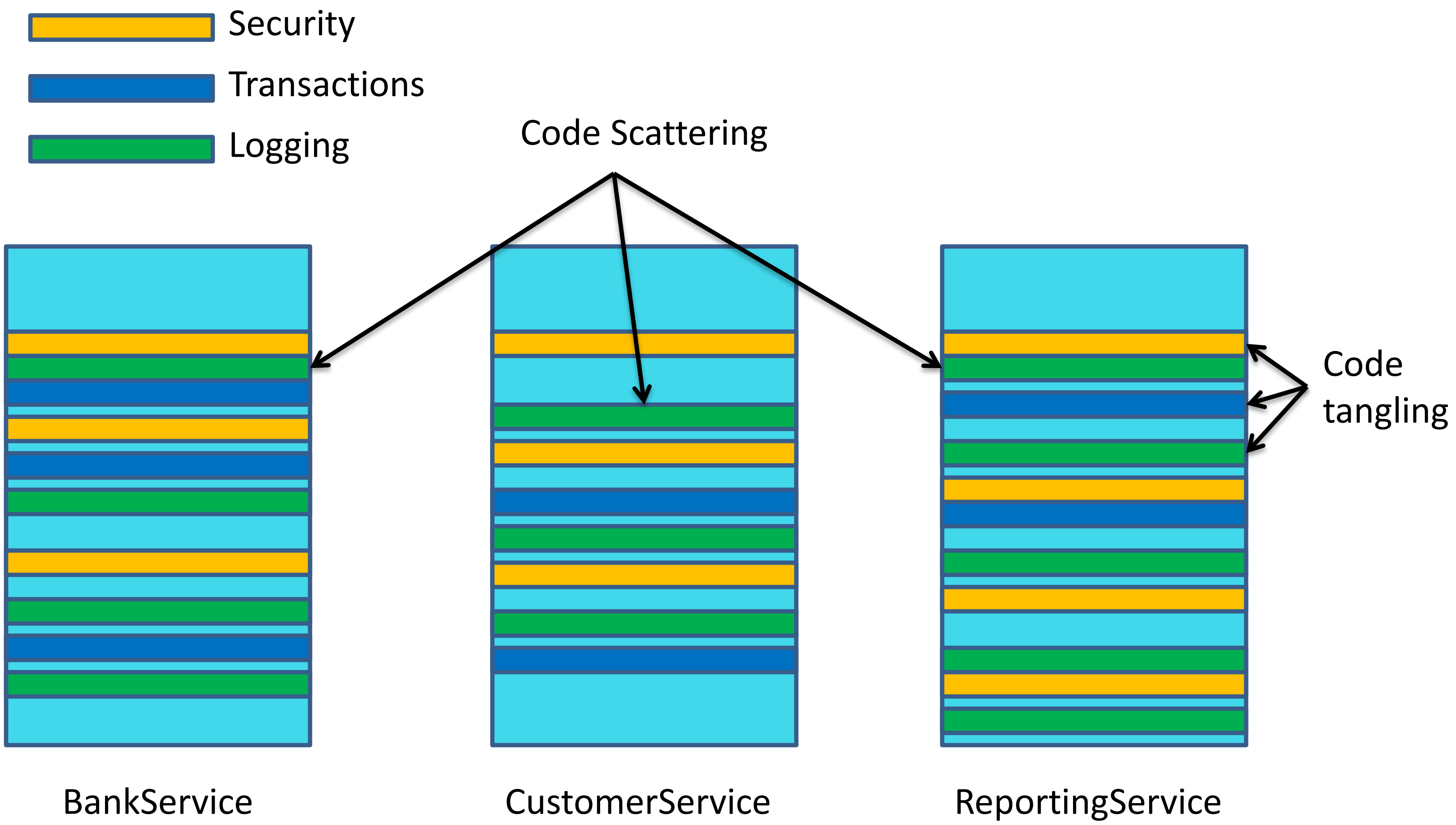
Code tangled with logging and transaction cross cutting concerns

```
private static Logger logger = Logger.getLogger(EmployeeDaoHibernateImpl.class);
public void insertEmployee(Employee employee) throws ApplicationException {
    logger.debug("Adding employee to DB: " + employee);
    Session session = null;
    Transaction tx = null;
    try {
        session = HibernateUtil.getSessionFactory().openSession();
        tx = session.beginTransaction();
        Serializable employeeId = session.save(employee);
        logger.debug("Employee Persisted with ID: " + employeeId);
        tx.commit();
    } catch (HibernateException ex) {
        tx.rollback();
        logger.debug("Exception occured : " + ex);
        throw new ApplicationException(ex.getMessage(), ex);
    } catch (ApplicationException e) {
        logger.debug("Exception occured : " + e);
        throw new ApplicationException(e.getMessage(), e);
    } finally {
        session.close();
    }
}
```





Aspect Oriented Programming



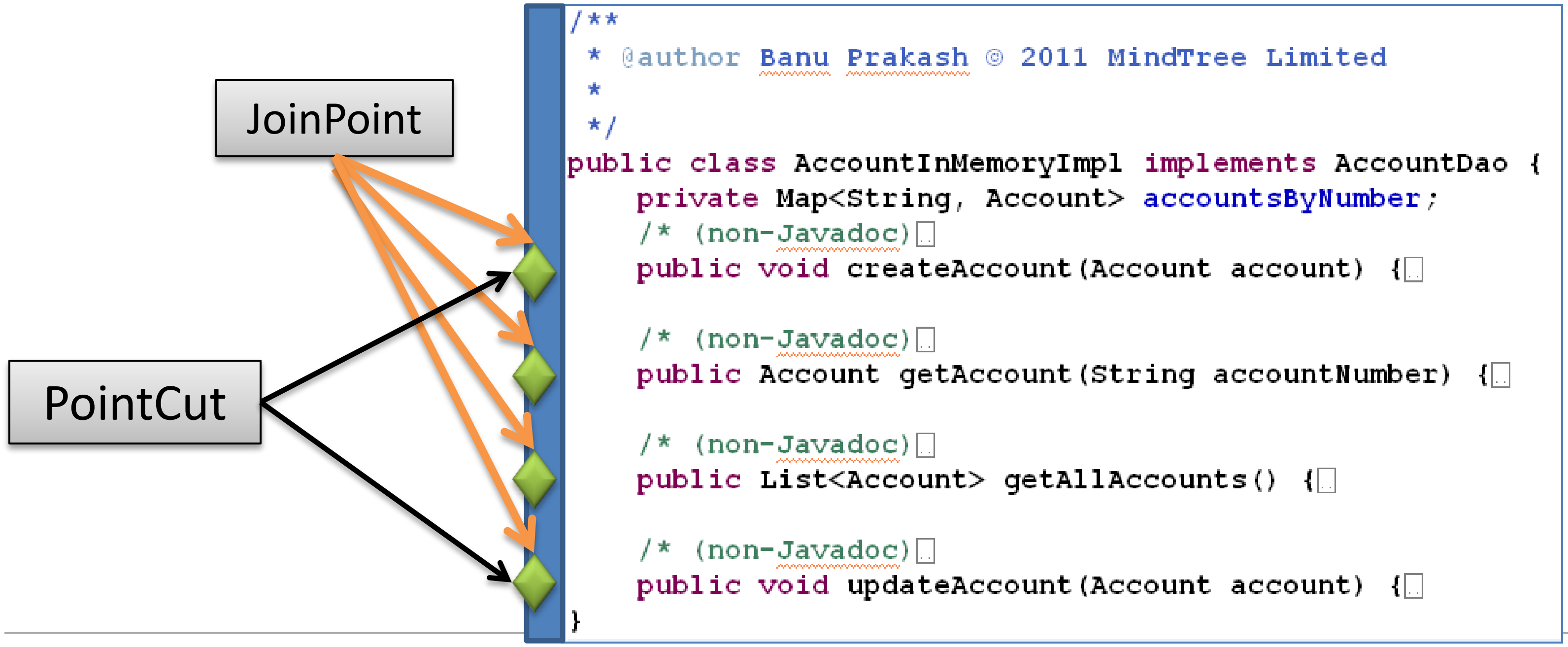
Mindtree

Aspect Oriented Programming

- Aspect Oriented Programming(**AOP**) is a programming paradigm which aims to increase modularity by allowing the separation of cross-cutting concerns.

Aspect Oriented Programming

- JoinPoint, Advice and PointCut
  - JoinPoint is a point during a program execution, such as a method executing or an exception being handled.
  - Advice is an information about when an aspect is to be executed with respect to the join point. Example: Before, after, around, etc.
  - PointCut is a point of control where an advice is applied.



```
/**
 * @author Banu Prakash © 2011 MindTree Limited
 *
 */
public class AccountInMemoryImpl implements AccountDao {
    private Map<String, Account> accountsByNumber;
    /* (non-Javadoc)
    public void createAccount(Account account) {

    /* (non-Javadoc)
    public Account getAccount(String accountNumber) {

    /* (non-Javadoc)
    public List<Account> getAllAccounts() {

    /* (non-Javadoc)
    public void updateAccount(Account account) {
}
```

Mindtree

Aspect Oriented Programming

- Spring AOP advice comes in different forms that let you choose when advice is executed relative to a JoinPoint.

Advice Type	interface
Before	org.springframework.aop.MethodBeforeAdvice
After-returning	org.springframework.aop.AfterReturningAdvice
After-throwing	org.springframework.aop.ThrowsAdvice
Around	org.aopalliance.intercept.MethodInterceptor

Aspect Oriented Programming

- Concerns, Advice and JoinPoint type in a simple banking application.

Concern	Advice	JoinPoint type	Description
Authenticati ng	BeforeAdvice	Method	Validate user
Integrity	BeforeAdvice	Method	Avoid adding duplicate items to the database
Auditing	AfterAdvice, AfterReturningAdvice	Method	Record operations performed by clerks for auditing
Logging	BeforeAdvice, AfterAdvice, AfterReturningAdvice, ThrowsAdvice	Method, Exception	Log operations performed by user and log exceptions.
Transaction	AroundAdvice	Method, Exception	Apply transaction around fund transfer method, rollback if any exception occurs
Profiling	AroundAdvice	Method	Profile how much time does it take to execute the business method

Welcome to possible

India | USA | UK | Germany | Sweden | Belgium | France | Switzerland | UAE | Singapore | Australia | Japan | China