

Aspect-Oriented Programming (AOP)

AOP and Spring MVC

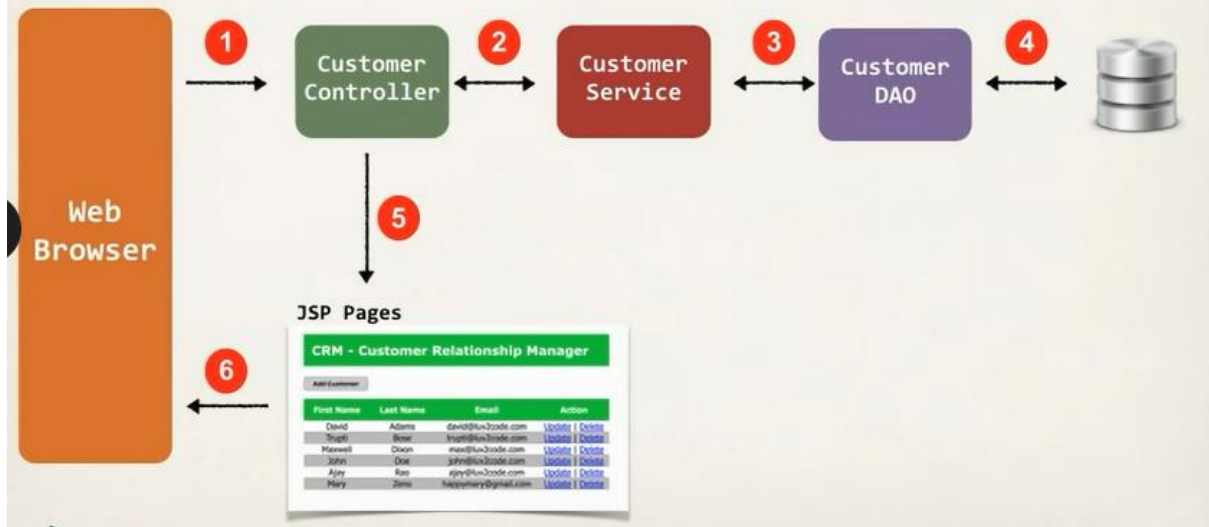


Goal

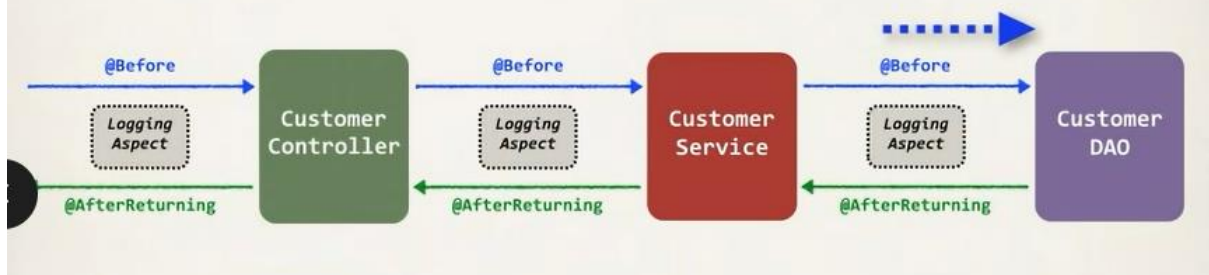
- Add AOP Logging support to our Spring MVC app (CRM Web App)

CRM - Customer Relationship Manager			
Add Customer			
First Name	Last Name	Email	Action
David	Adams	david@luv2code.com	Update Delete
Trupti	Bose	trupti@luv2code.com	Update Delete
Maxwell	Dixon	max@luv2code.com	Update Delete
John	Doe	john@luv2code.com	Update Delete
Ajay	Rao	ajay@luv2code.com	Update Delete
Mary	Zeno	happymary@gmail.com	Update Delete

Big Picture



Logging Aspect



Development Process

Step-By-Step

1. Add AspectJ JAR file to web project
2. Enable AspectJ Auto Proxy
3. Create Aspect
 1. Add logging support
 2. Setup pointcut declarations
 3. Add @Before advice
 4. Add @AfterReturning advice

Step 1: Add AspectJ JAR file to web project


- All Spring projects require AspectJ JAR file for AOP support
- Can download from
 - <http://www.luv2code.com/download-aspectjweaver>
- Add JAR file to your web project: **WEB-INF/lib** directory

Step 2: Enable AspectJ Auto Proxy (XML)

```
<beans xmlns="http://www.springframework.org/schema/beans"
...
xmlns:aop="http://www.springframework.org/schema/aop"
xsi:schemaLocation="
...
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop.xsd">

<!-- Add AspectJ autoproxy support for AOP -->
<aop:aspectj-autoproxy />

...
</beans>
```



For processing
@Aspect classes

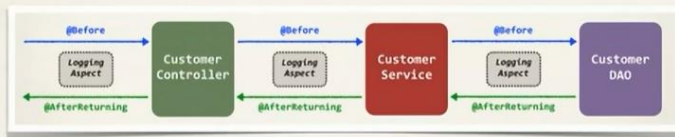
Step 2: Enable AspectJ Auto Proxy (Java)

- If you want to use pure Java configuration then use the annotation

@EnableAspectJAutoProxy

For processing
@Aspect classes

Pointcut Declarations



src

- com.luv2code.springdemo.controller
- com.luv2code.springdemo.dao
- com.luv2code.springdemo.entity
- com.luv2code.springdemo.service
- com.luv2code.testdb

Only match on these three packages



Goal

✓ Add AOP Logging support to our Spring MVC app (CRM Web App)

CRM - Customer Relationship Manager

Add Customer

First Name	Last Name	Email	Action
David	Adams	david@luv2code.com	Update Delete
Trupti	Bose	trupti@luv2code.com	Update Delete
Maxwell	Dixon	max@luv2code.com	Update Delete
John	Doe	john@luv2code.com	Update Delete
Ajay	Rao		
Mary	Zeno		




```
@Aspect
@Component
public class CRMLoggingAspect {
    // Setup logger
    4 usages
    private static final Logger log = LogManager.getLogger(CRMLoggingAspect.class);

    // Setup pointcut declarations
    no usages
    @Pointcut("execution(* com.zemoso.springdemo.controller.*(..))")
    private void forControllerPackage() {
    }

    no usages
    @Pointcut("execution(* com.zemoso.springdemo.service.*(..))")
    private void forServicePackage() {
    }

    no usages
    @Pointcut("execution(* com.zemoso.springdemo.dao.*(..))")
    private void forDaoPackage() {
    }
}
```

```
@Pointcut("forControllerPackage() || forServicePackage() || forDaoPackage()")
private void forAppFlow() {
}

// Add @Before advice
no usages
@Before("forAppFlow()")
public void before(JoinPoint theJoinPoint) {
    // Display the method we are calling
    String theMethodName = theJoinPoint.getSignature().toShortString();
    log.info("≡>>> in @Before: calling method " + theMethodName);

    // Display the arguments to the method
    // Get the arguments
    Object[] args = theJoinPoint.getArgs();

    // Loop through and display the arguments
    for (Object arg : args) {
        log.info("≡>>> argument: " + arg);
    }
}
```

```
// Add @AfterReturning advice
no usages
@AfterReturning(
    pointcut = "forAppFlow()",
    returning = "theResult")
public void afterReturning(JoinPoint theJoinPoint, Object theResult) {
    // Display the method we are calling
    String theMethodName = theJoinPoint.getSignature().toShortString();
    log.info("≡>>> in @AfterReturning: from method " + theMethodName);

    // Display data returned
    log.info("≡>>> result: " + theResult);
}

}
```