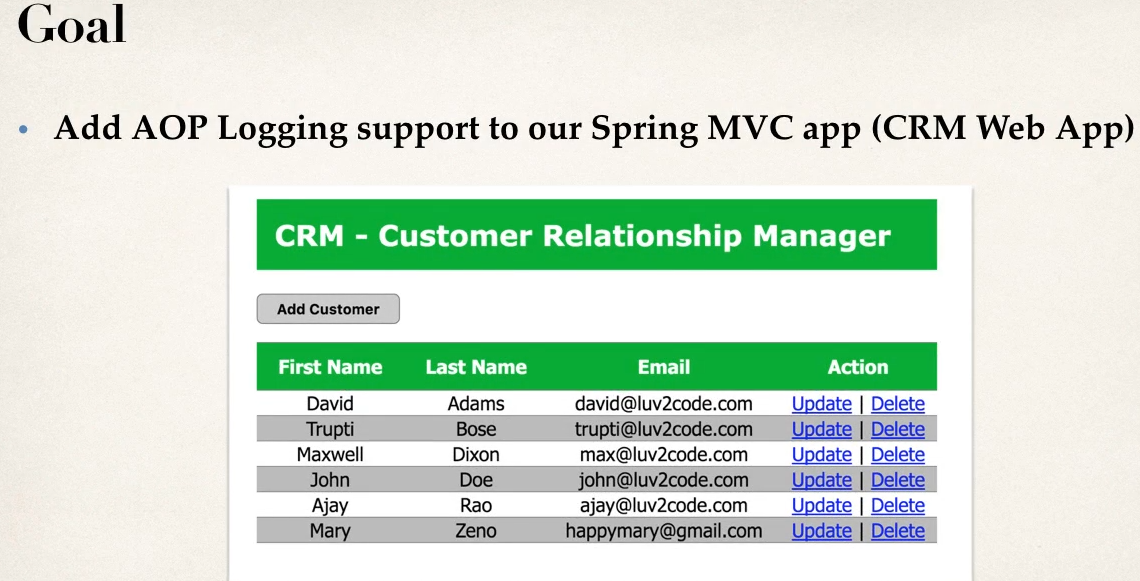
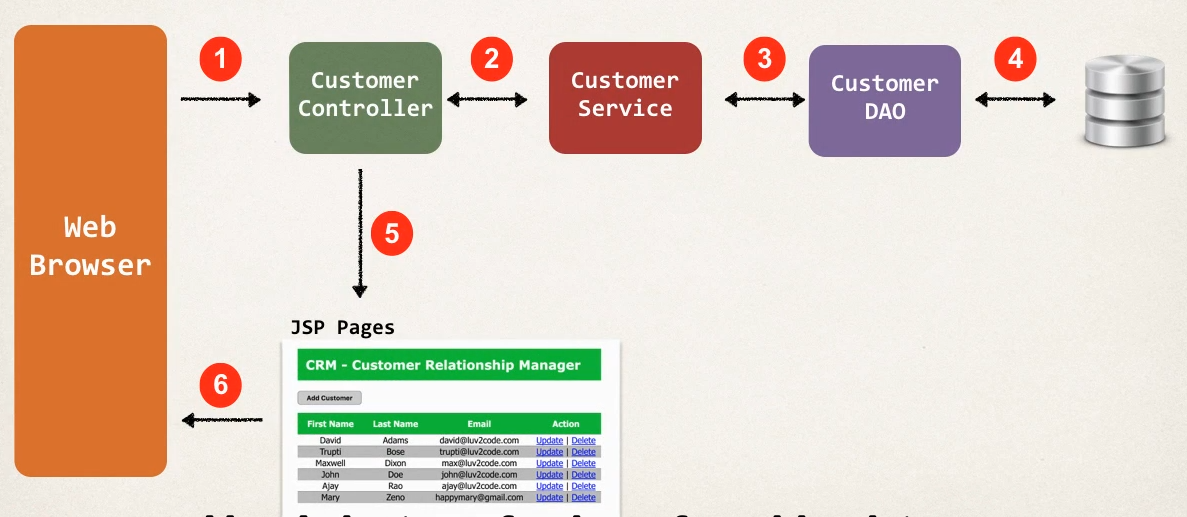
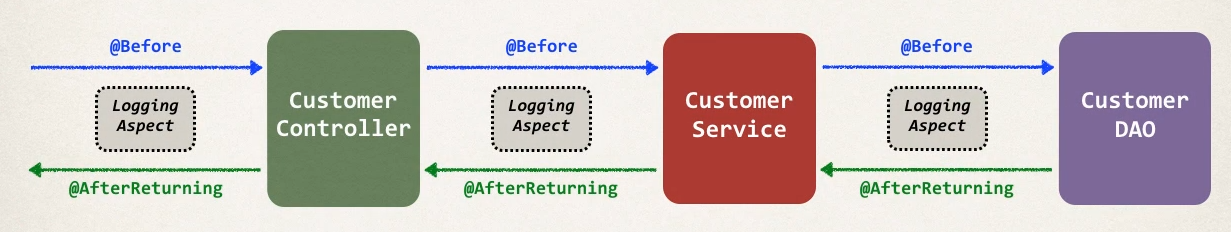
SPRING AOP with SPRING MVC







Development Process

1 Add AspectJ JAr to web project

aspectjweaver-1.9.5.jar

2 Enable AspectJ Auto Proxy

Config **namespaces**

xmlns:aop="http://www.springframework.org/schema/aop"

Config **Schema**

xsi:schemaLocation = "http://www.springframework.org/schema/aop

[http://www.springframework.org/schema/aop/spring-aop.xsd"](http://www.springframework.org/schema/aop/spring-aop.xsd\")

<!-- Adding AspectJ autoproxy Support for AOP // for processing @Aspect classes-->

<aop:aspectj-autoproxy/>

If in case pure core java configuration then use Annotaion place it on java configuration file

@EnableAspectJAutoProxy

3 Create Aspect

A add logging support

B Setup PointCut Decalrations

C add @Before Advice

D add @AfterReturning advice

Before 3 step

Create PointCutDecalrations for all pacakge and merge it as PointCut

Only include controller,service,and dao not entity

Create a new package to write Aspect related Codes

@Aspect

@Component

public class CRMLoggingAspect {

// setup Logger

private Logger myLogger=Logger.getLogger(getClass().getName());

// setup Pointcut declaration

// Any Class any method any Number of Arguments

// Controller

@Pointcut("execution(\* com.crm.controller.\*.\*(..))")

private void forControllerpkg() {}

// Sevice

@Pointcut("execution(\* com.crm.service.\*.\*(..))")

private void forServicepkg() {}

// Daos

@Pointcut("execution(\* com.crm.dao.\*.\*(..))")

private void forDaopkg() {}

@Pointcut(" forControllerpkg() || forServicepkg() || forDaopkg() ")

private void forAppFlow() {}

// add @Before Advice

@Before("forAppFlow()")

public void beforeLoggs(JoinPoint joinPoint)

{

// display Method whcih are calling

String method=joinPoint.getSignature().toShortString();

myLogger.info("======> in @Before :Calling Method "+ method);

// display argumnets to method

//Get the Arguments

Object[] args=joinPoint.getArgs();

//loop through and display args

for (Object object : args) {

myLogger.info("======> Arguments "+ object);

}

}

// add @AfterReturning advice

@AfterReturning(pointcut="forAppFlow()",returning="result")

public void afterRetruning(JoinPoint joinPoint,Object result)

{

// display method we are returning from

String method=joinPoint.getSignature().toShortString();

myLogger.info("======> in @AfterReturning :Calling Method "+ method);

// display data returned

myLogger.info("======> Results "+ result);

}

}