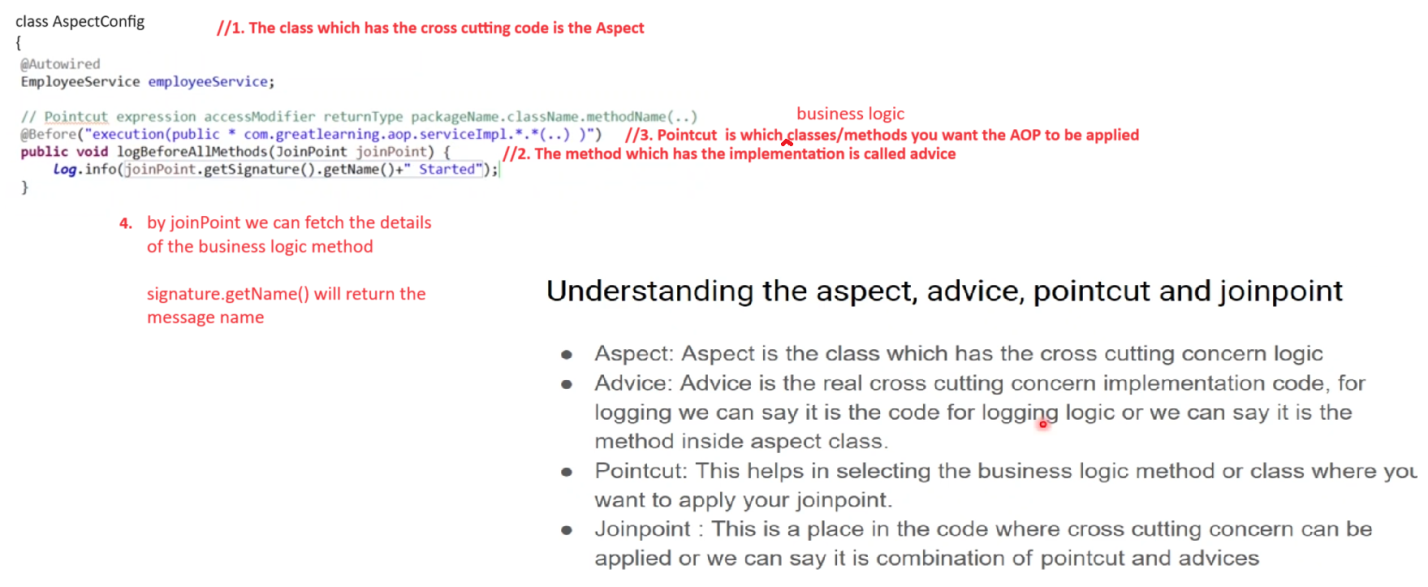
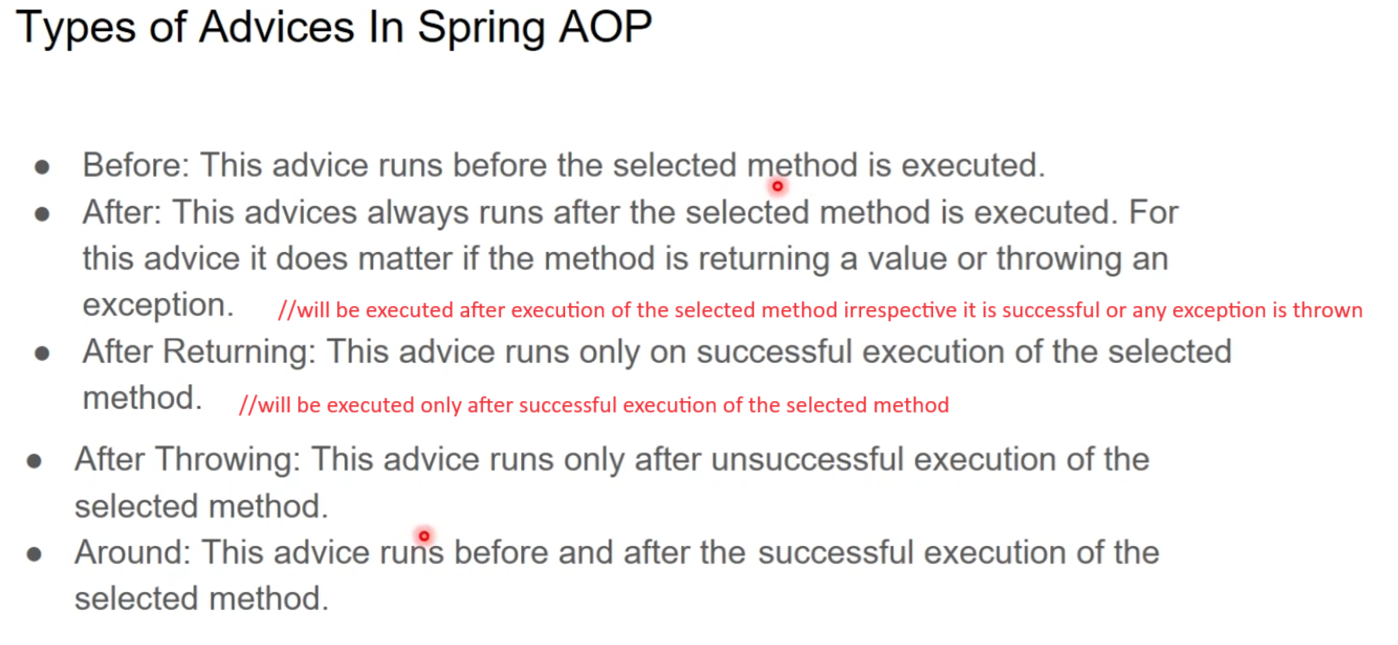
AOP

JoinPoint is the connection between the business logic & the advice method

PointCut : on which methods u need the AOP





In the config package create a class

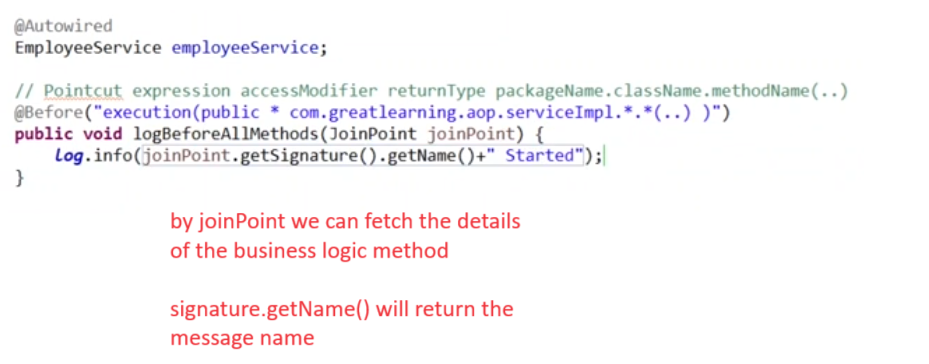
@Configuration

@Aspect

@Slf4j

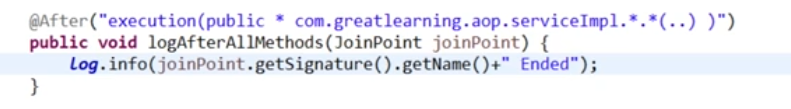
Class AspectConfig

{

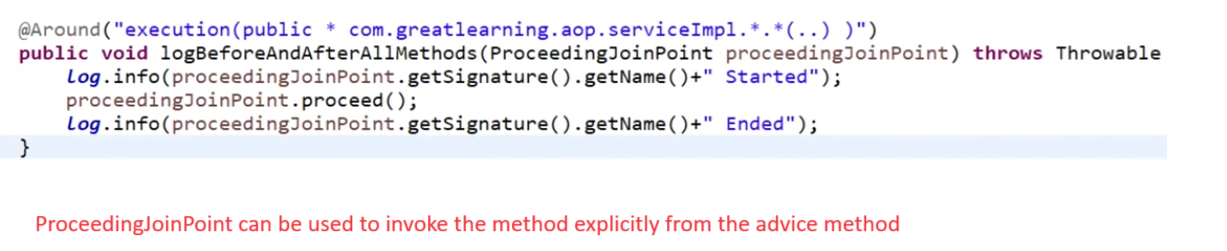


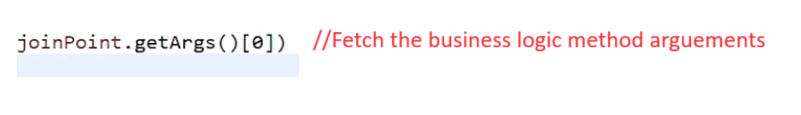
}

**@After**

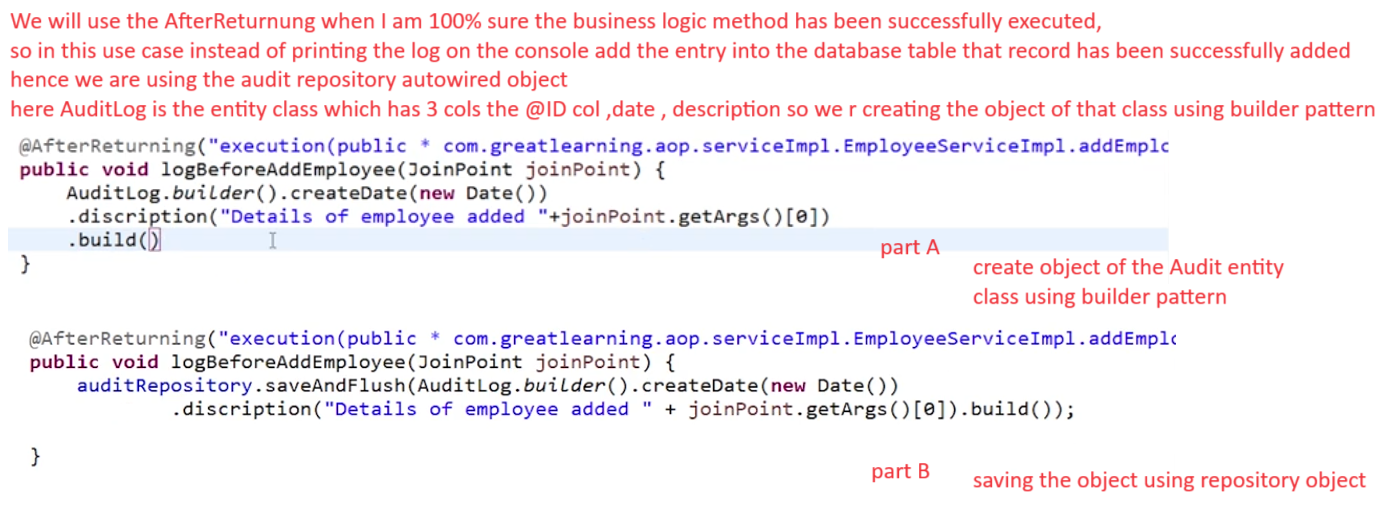


**@Around**

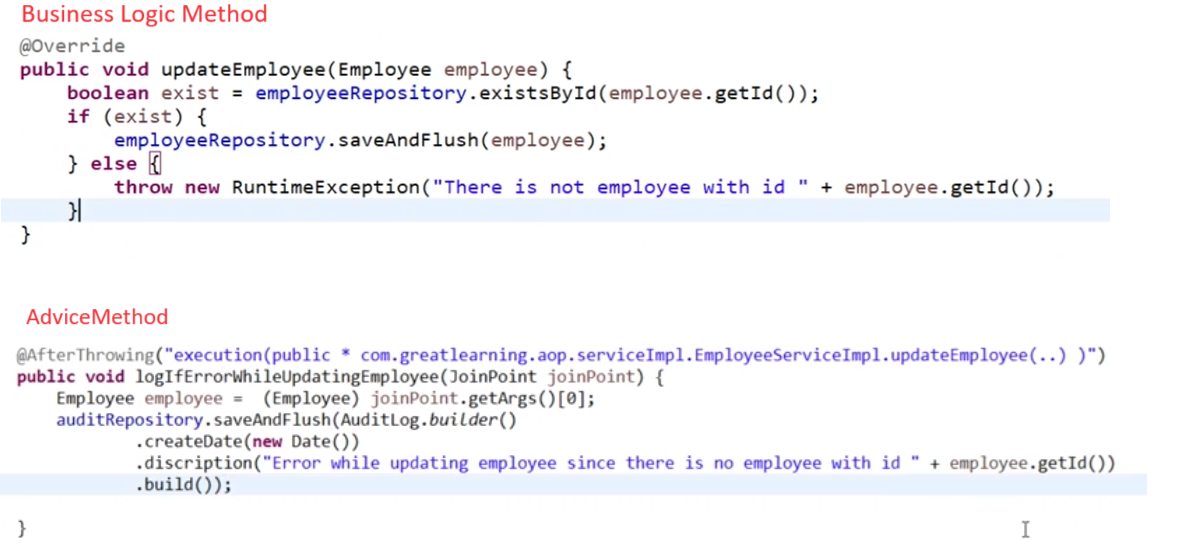


**NOTE :** 

**@AfterReturning**



**@AfterThrowing**



package com.excelr.aop.config;

import java.util.Date;

import org.aspectj.lang.JoinPoint;

import org.aspectj.lang.ProceedingJoinPoint;

import org.aspectj.lang.annotation.After;

import org.aspectj.lang.annotation.AfterReturning;

import org.aspectj.lang.annotation.AfterThrowing;

import org.aspectj.lang.annotation.Around;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Before;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Configuration;

import com.excelr.aop.entity.AuditLog;

import com.excelr.aop.entity.Employee;

import com.excelr.aop.repository.AuditRepository;

import com.excelr.aop.service.EmployeeService;

import lombok.extern.slf4j.Slf4j;

@Slf4j

@Aspect

@Configuration

public class AspectConfig {

@Autowired

AuditRepository auditRepository;

@Autowired

EmployeeService employeeService;

// Pointcut expression accessModifier returnType

// packageName.className.methodName(..)

// @Before("execution(public \* com.excelr.aop.serviceImpl.\*.\*(..) )")

// public void logBeforeAllMethods(JoinPoint joinPoint) {

// log.info(joinPoint.getSignature().getName() + " Started");

// }

//

// @After("execution(public \* com.excelr.aop.serviceImpl.\*.\*(..) )")

// public void logAfterAllMethods(JoinPoint joinPoint) {

// log.info(joinPoint.getSignature().getName() + " Ended");

// }

@Around("execution(public \* com.excelr.aop.serviceImpl.\*.\*(..) )")

public void logBeforeAndAfterAllMethods(ProceedingJoinPoint proceedingJoinPoint) throws Throwable {

log.info(proceedingJoinPoint.getSignature().getName() + " Started");

proceedingJoinPoint.proceed();

log.info(proceedingJoinPoint.getSignature().getName() + " Ended");

}

@AfterReturning("execution(public \* com.excelr.aop.serviceImpl.EmployeeServiceImpl.addEmployee(..) )")

public void logBeforeAddEmployee(JoinPoint joinPoint) {

auditRepository.saveAndFlush(AuditLog.builder().createDate(new Date())

.discription("Details of employee added " + joinPoint.getArgs()[0]).build());

}

@AfterThrowing("execution(public \* com.excelr.aop.serviceImpl.EmployeeServiceImpl.updateEmployee(..) )")

public void logIfErrorWhileUpdatingEmployee(JoinPoint joinPoint) {

Employee employee = (Employee) joinPoint.getArgs()[0];

auditRepository.saveAndFlush(AuditLog.builder().createDate(new Date())

.discription("Error while updating employee since there is no employee with id " + employee.getId())

.build());

}

}