

**- spring은 자바 언어를 사용하는 어플리케이션을 개발하는 기술이다.**

**- spring은 일정한 틀(프로그램 규칙) 내에서 어플리케이션을 개발한다.**

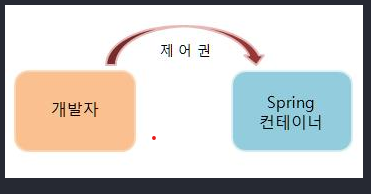
**- spring은 어플리케이션 내부에서 필요한 객체를 직접 생성하지 않고 spring 에서 제공하는 객체를 사용하도록 한다.**

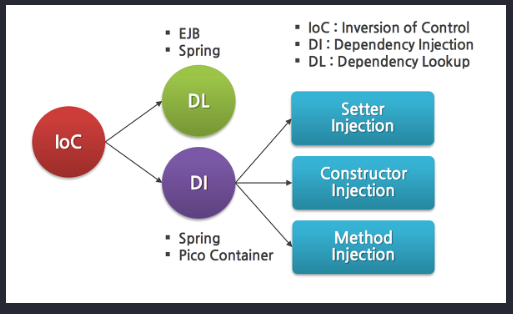
**- spring은 객체를 생성 – 관리 – 소멸하여 객체의 라이프 사이클을 관리한다.**

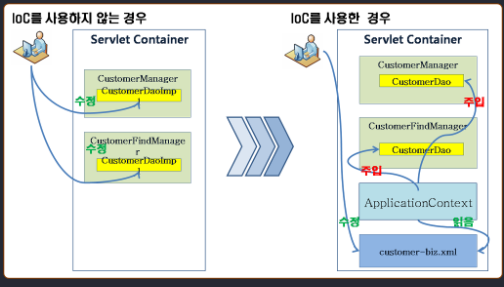
**- 객체간의 의존성(dependency)을 표현할 때 결합도(coupling)을 낮춘다.**

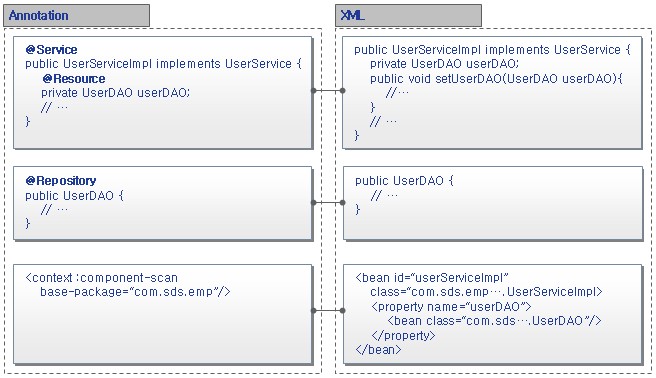
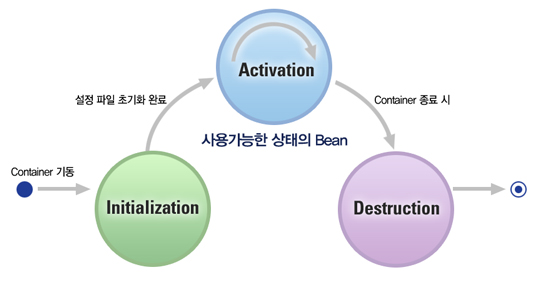
**- IOC / DI**

**- XML 태그 설정 / 자바 ANNOTATION 설정**









**SPRING AOP**

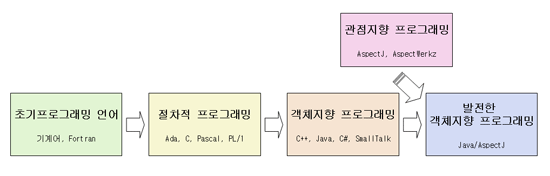
**OOP : OBJECT ORIENTED PROGRAMMING**

**:객체 지향 프로그래밍 (자바, C++, C#, ....)**

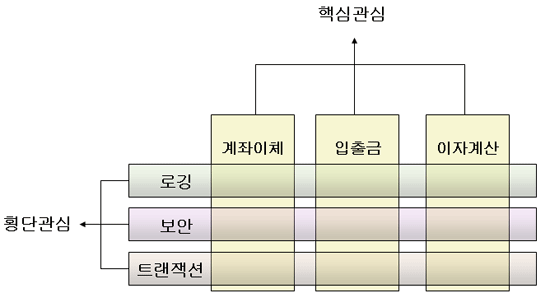
**AOP : ASPECT ORIENTED PROGRAMMING**

**aspect : 모든 스프링 클래스 공통 구현 사항들**

**(반복적 코딩 필요 – 1번 정의 – 필요한 클래스에 연결 / 해제 )**

[](http://c2down.cyworld.nate.com/download?fid=64221081a8b7f600b9ab11966874f805&name=f1.gif)

<프로그래밍 패러다임의 변화>

[](http://c2down.cyworld.nate.com/download?fid=64221081a8b7f6006a7c11966a29f205&name=f2.gif)

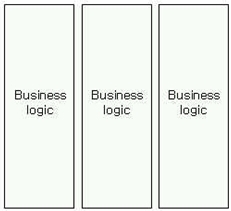
<횡단 관심 (crosscutting concerns)>

(Code) (Advice)

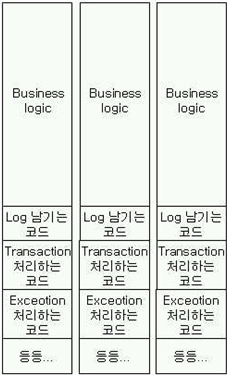
Point-Cut

----🡪Weaving

<위빙 (weaving)>

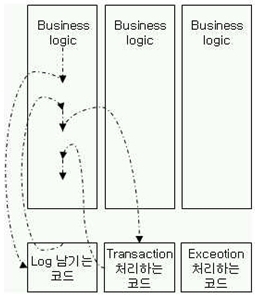


- 코딩하다 보니까 자꾸 비슷한 코드들이 따라 붙더라...

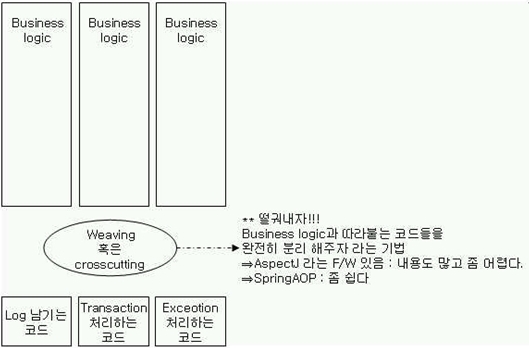


그래서 그런 놈들을 별도의 클래스로 작성하고 나니 코드 지저분.

- 커플링(COUPLING)도 밀접하게 달라 붙더라 (유지보수 떨어져!!!)



- 그래서 생긴게 weaving!!! => business logic과 나머지 관심사 분리하자



Member

login(){

}

insert(){

db 연결

db 해제

}

Product

sell(){

예외처리}

buy()

{실행시간}

Board

get(){

db 연결

db 해제

}

regiter(){

예외처리

}

Exception

check(){}

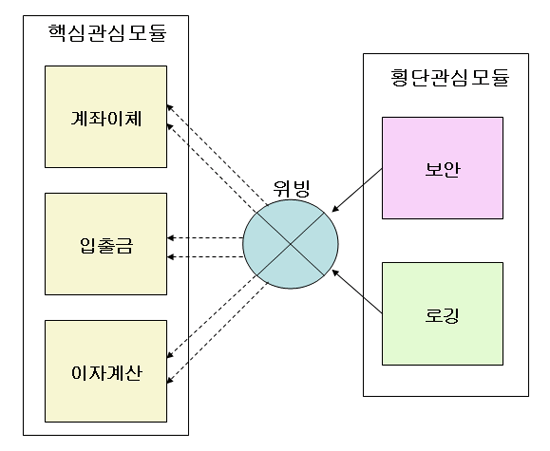
Database

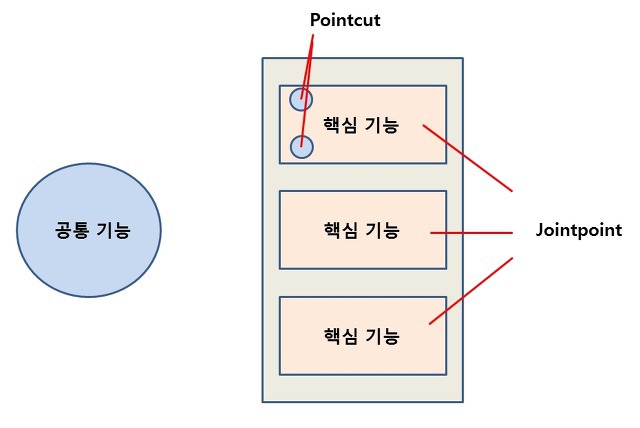
connection(){}

close(){}

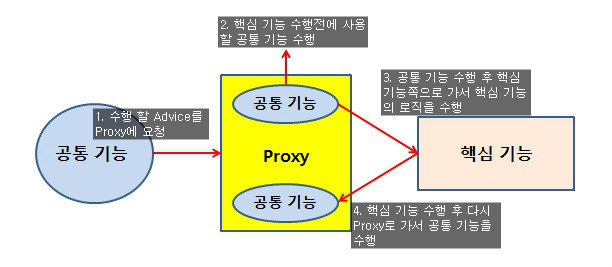
Time

check(){}

[](http://c2down.cyworld.nate.com/download?fid=64221081a8b7f600462d11966ae52d05&name=f3.gif)



**--proxy pattern**



**mvnrepository.com 사이트**

**- 스프링 프로젝트 기본 jar 라이브러리 파일들 다운로드**

**+ aop 관렴 라이브러리 추가 다운로드**

**1> aspectjweaver-xxx.jar**

**2> aspect 클래스 정의 ( 공통관심)**

**3> target 클래스 정의 (핵심관심)**

**4> pointcut 문법**

**\* : 모든   
(..) : 모든 매개변수**

**.. : 하위패키지 포함**

expression=

*"execution(public \* aoptest1..\*.\*(..))"*

*(modifier 리턴타입 패키지명.클래스명.메소드명(..)*

expression=

*"within(aop1.sub.A)" : aop1.sub.A클래스 모든 메소드*

expression=

*"bean(board)|| within(aop1.sub.A) " : board 이름 bean 모든 메소드*

**<aop:config>**

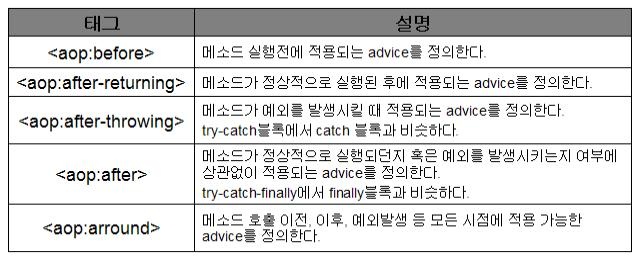
**<aop:pointcut** expression=

*"execution(public \* aoptest1..\*.\*(..))"  
id="p" />*

**<aop:advice ref="공통관심클래스객체" >**

**<aop:aroud method="메소드명 pointcut-ref="p"**

**</aop:config>**



**■ JoinPoint 클래스**

**￭ org.aspectj.lang.JoinPoint**

**￭ 대상 객체 및 호출되는 메소드에 대한 정보, 전달되는 파라미터에 대한 정보가 필요한 경우 사용한다.**

**￭ JointPoint의 주요 메소드**

