+ spring

: Enterprise급 application 개발을 위한 framework

1. Rod Johnson이 창시
2. 애플리케이션 개발의 복잡성을 줄여 주기 위한 목적
3. EJB 사용으로 수행되었던 모든 기능을 POJO를 사용해서 가능하게 함

* EJB (Enterprise JavaBeans)

: 기업 환경을 구축하기 위한 서버측 컴포넌트 모델

: 애플리케이션 업무 로직을 가지고 있는 서버 애플리케이션

* POJO (Plain Old Java Object)

: getter/setter를 가진 단순한 자바 오브젝트

1. IOC (Inversion Of Control) : 제어의 역전

* 어플리케이션 동작에 대한 제어권이 개발자에게 있는 게 아니라 프레임워크의 컨테이너에게 제어권이 있다

DI (Dependency Injection) : 의존성 주입

* 컨테이너에 있는 정보를 스프링 프레임워크가 run time시 불러와서 동작

1. AOP (Aspect Oriented Programming)
2. POJO (Plain Old Java Project) 지원
3. 경량 컨테이너
4. 모듈화 되어 있어 필요한 모듈만 추가 가능

+ 스프링은 모듈화 되어있어 모듈을 가져다 build 후 사용하면 됨.

그러나 이 모듈을 release에 따라서 계속 바꿔서 추가해줘야 하는 불편함이 생김

메이븐은 빌드를 위한 모듈 제공, 배포관리

maven : 빌드 툴(build tool) + 오픈소스 버전관리

+ AOP(Aspect Oriented Programming)

- OOP를 더 잘 할 수 있도록 도와주는 보조 프로그램

- 원래 업무들에서 공통된 업무들을 찾고 그 업무들을 AOP로 만들어 놓고

원래 업무 수행 시 사이 사이에 끼워넣는 형태로 구현

ex) 인터넷 뱅킹

핵심 업무(기능) : 이체, 송금, 출금, 대출, 조회

공통 업무(기능) : 로그인, 출금한도, 기록, 권한검사, 감사

> AOP 적용하는 방법 3가지

1. 컴파일 시점에 코드에 기능을 추가하는 방법

2. 클래스를 로딩하는 시점에 바이트 코드에 공통 기능을 추가하는 방법

(1~2) 은 AspectJ로 지원

3. 런타임 시에 Proxy 객체를 생성해서 공통 기능을 추가하는 방법

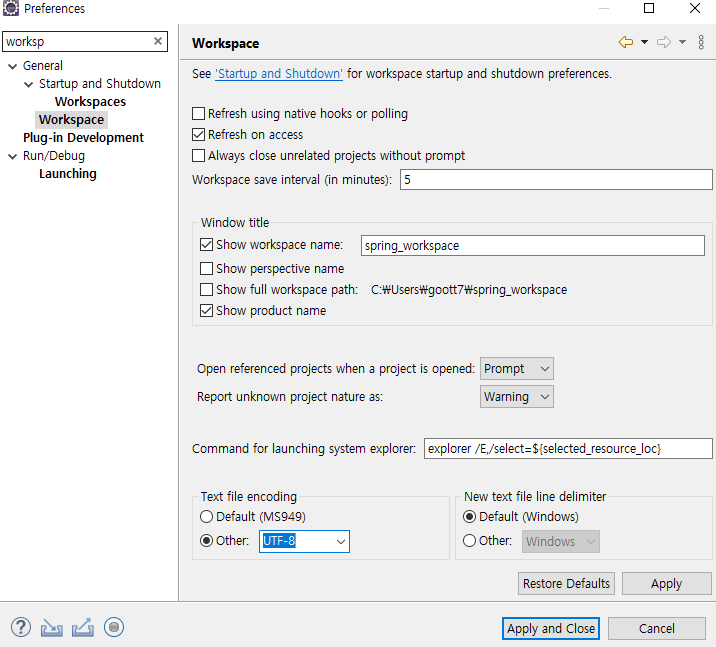
(3)은 spring-aop 지원

스프링 설치

<https://spring.io/> Boot, FrameWork, Security, REST Docs등이 쓰임

C:\Users\goott7\spring\_workspace 새로운 워크스페이스 생성

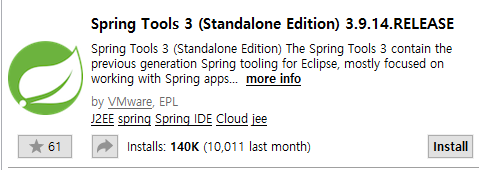
Window-preferences

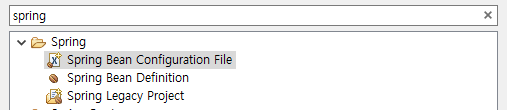
 utf-8처리

Spring을 쓰기 위해 sts가 필요

홈페이지 Spring.io-Projects-Spring Tools 4

3으로 구조를 배운 후에 4를 쓰라 함  
이클립스 마켓플레이스에서 sts검색





이 XML은 3에서만 있고 4에서는 지원하지 않음 3 설치 확인

Spring legacy project는 4기반, 일반 자바 프로젝트를 생성하고 설정하는 것부터

메이븐 설치

<https://maven.apache.org/>

그냥 프로젝트의 properties에서 변경 가능

<https://start.spring.io/> 여기서 만들 수도 있음

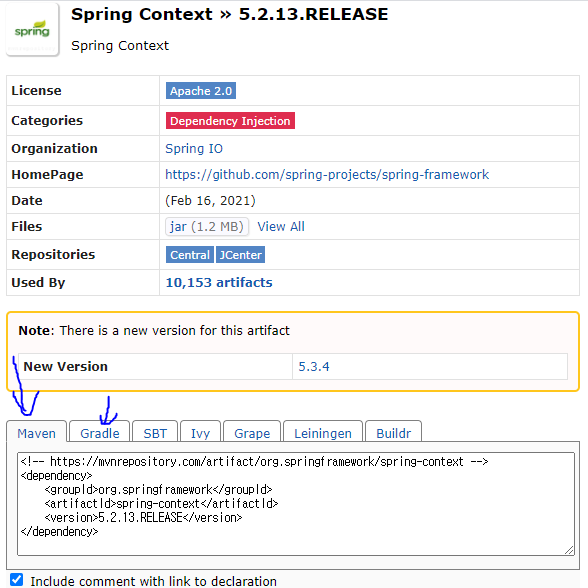
<https://mvnrepository.com/> 사용가능한 모듈들이 있음

<https://mvnrepository.com/artifact/org.springframework/spring-context>

spring-context를 찾아서 Central

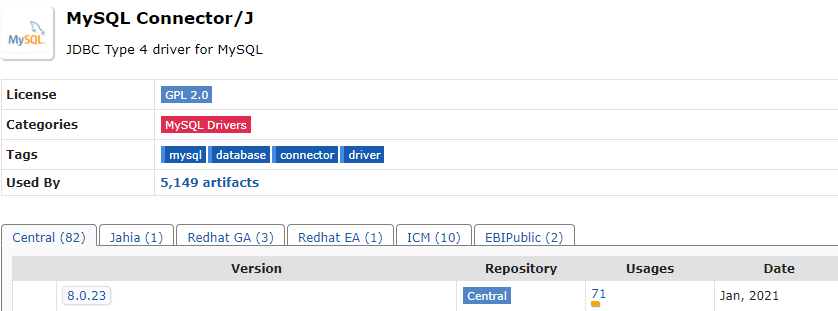
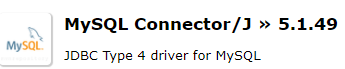
스프링 사이트에서 현제 안정버전을 볼 수 있음

 하지만 이전 안정 버전을 쓸 것이다.

 이중 maven과 gradle이 가장 많이 쓰임

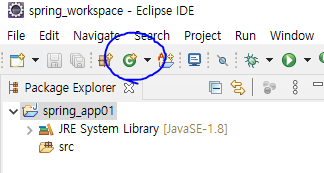
복사를 하고 dependencies를 먼저 열고 그 안에 추가하면 됨

mysql도 검색해서

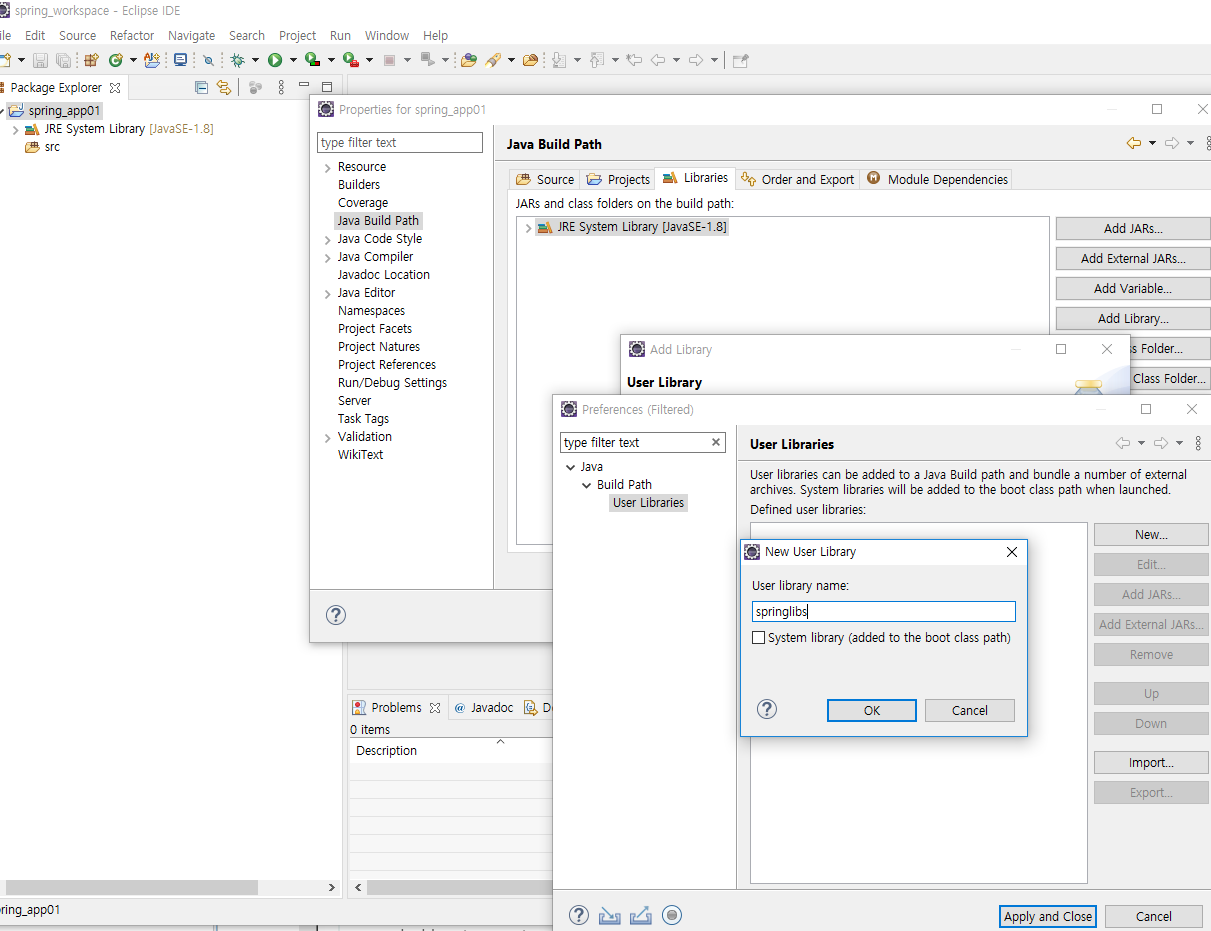
5버전인놈으로 받아서 쓰면 됨, 오라클은 안줌

spring\_app01로 일반 자바 프로젝트 생성 + sts + libarary 넣어서 스프링 프로젝트로

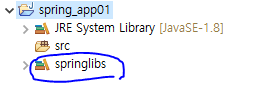
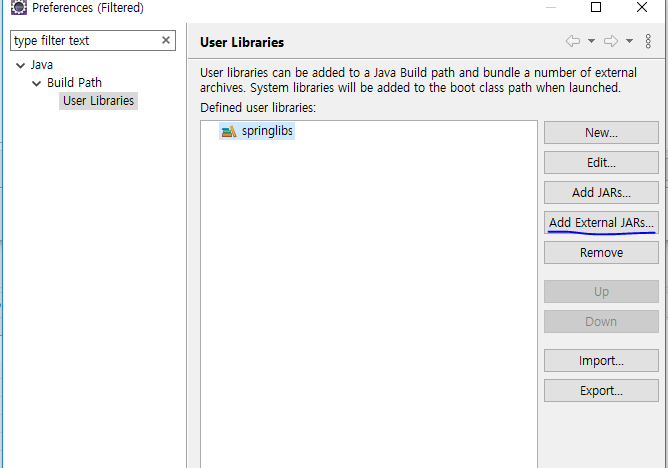
sts가 깔리면 이 버튼이 생김

spring\_libs를 zip풀어서 C에 놓기

프로젝트 build path-configure build path-유저라이브러리 추가->springlibs



Add External JARs로 받은 모든 라이브러리 넣기

생김

Hello.java interface, 이름 받아서 출력용

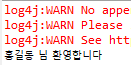
HelloKr.java 한글로 출력

TestMain .java HelloKr 가져와서 한글로 출력

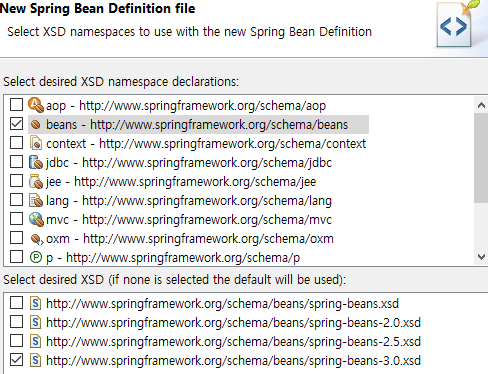
-일반 자바

-스프링 빈 설정파일 사용





Src에 spring bean 설정파일 생성-app.xml, next가서 버전 선택



Xsd는 규약, dtd임

app.xml 스프링 컨테이너가 사용할 객체의 정보를 이곳에 등록

ctrl+space 눌러서 있는거 불러올 수 있음



일본어, 중국어 환영 인사 추가 후 테스트 완료

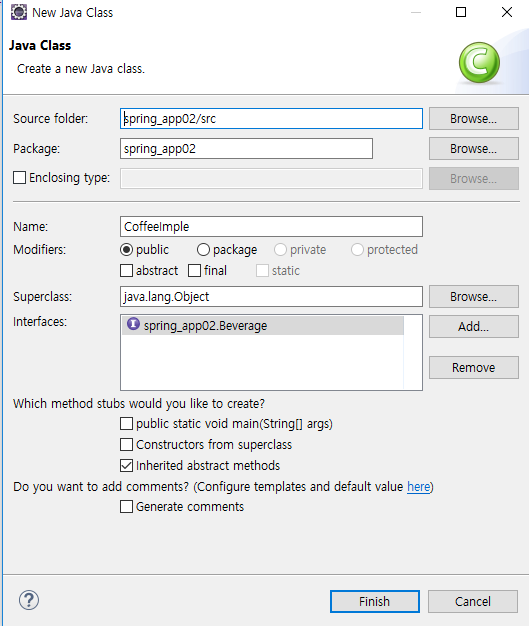
spring\_app02 자바 프로젝트 추가, spring bean 설정 파일을 추가하면 스프링 프로젝트됨

Beverage.java interface, 기본 생성자에 냉수 설정됨

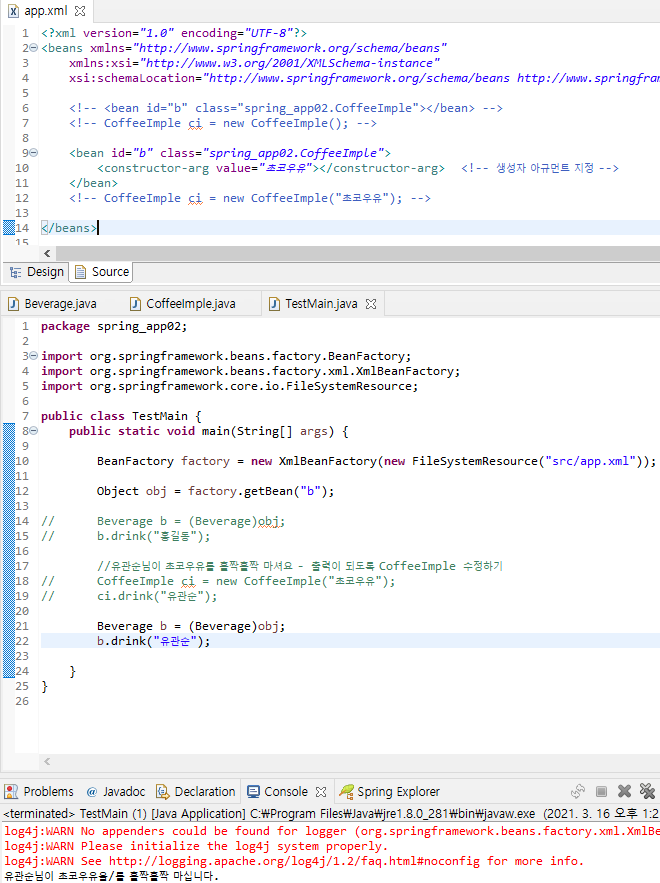
CoffeeImple.java Imple 받아 온다는거 이름에 써놓음, 구현객체

생성자 오버로딩 가능하게 하고 음료 종류바꿈

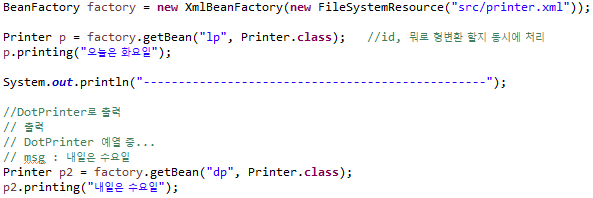
DI (Dependency Injection) : 의존관계에 있어서 xml에 있는 정보를 스프링 프레임워크가 run time시 호출해서 동작

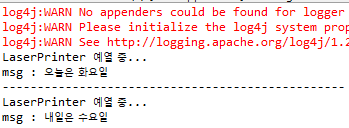


TestMain.java 테스트용

 생성자 bean파일에서 지정해서 출력하게함

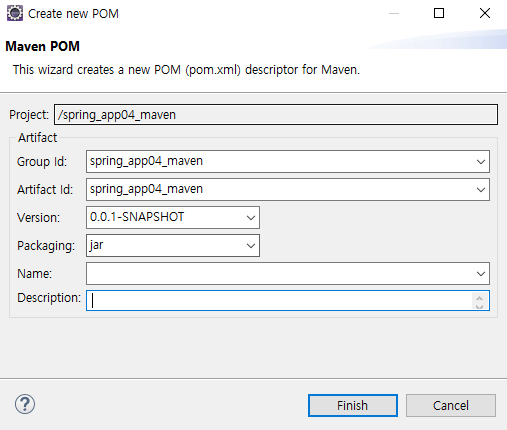
spring\_app03에서는 factory.getBean(id, 변환 클래스)를 통해 한번에 처리하는거 나옴





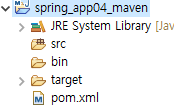
spring\_app04\_maven

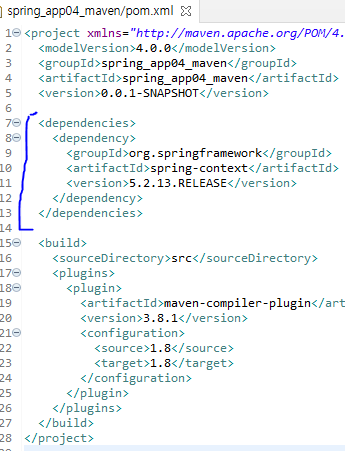
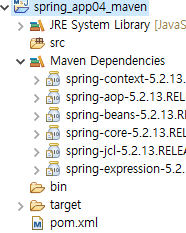
properties-configure-convert to maven project



Packaging 🡪 일반 자바는 jar, 웹 프로젝트는 war

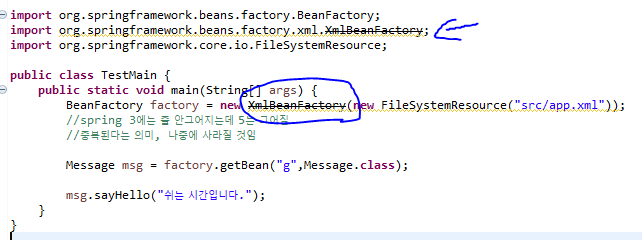
변경이 되면 pom.xml이 생긴다



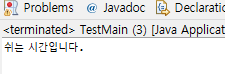
 메이븐 모듈 받아와서 넣기 

dependencies먼저 열고 그 안에 dependency를 넣어야 함, 그럼 라이브러리에 다음과 같이 추가되는 것 확인

spring bean 설정 파일도 버전이 4.3까지 추가됨, 4.3 사용



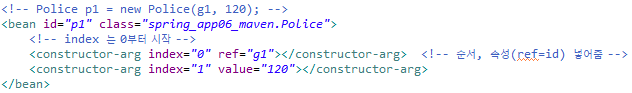
중복되어 사라질 코드가 있음, 이전의 빨간 오류 표기들이 다 사라짐을 볼 수 있다.



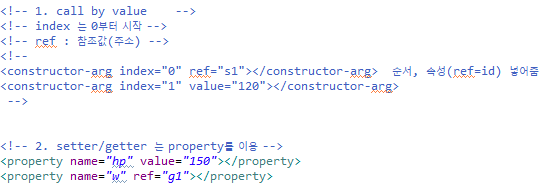
spring\_app05\_maven DB 연결 실험, 싱글톤

spring\_app06\_maven 게임(?)

app.xml에서 index 사용



setter/getter를 dao에서 선언해놓고 그걸 사용해서 제어



spring\_app07\_anno 현재 시간을 다른 객체에 만들고

그 객체를 xml 설정(컨테이너)에서 받아서 쓰도록 함

불러와서 쓰는 방법

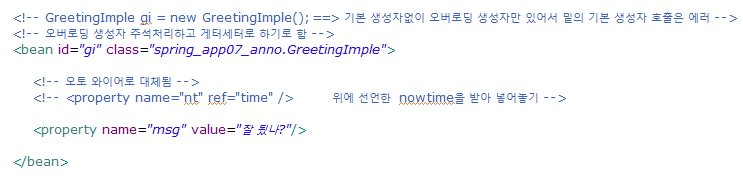




오토와이어로 대체







spring\_app08\_anno 좀더 간단하게 오토와이어를 사용하는 거만 해놓음

property로 세터에 값 넣는부분 없이 오토

단점: 한 개만 됨, 2개 이상을 받아와야 하면 지정해서 사용해야 한다.

spring\_app09\_anno 오토 와이어 여러 개 사용할 때

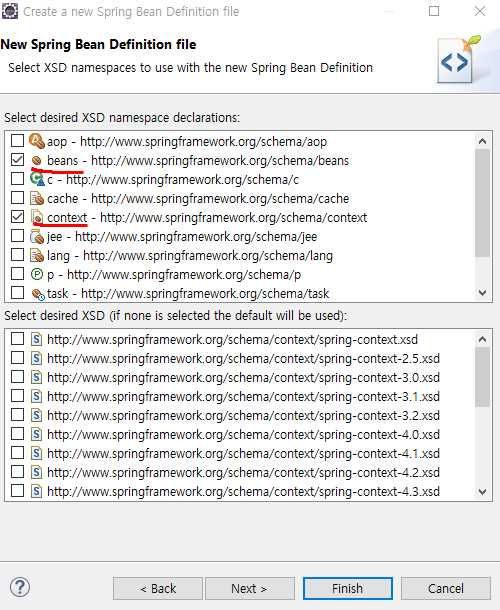
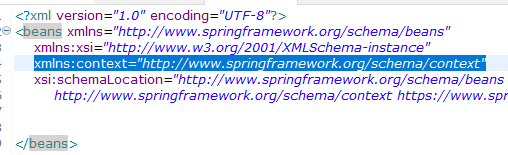
DI 객체가 2개 이상이면 자동으로 의존주입이 안됨

해결방법:

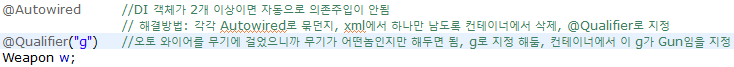
각각 Autowired로 묶던지

xml에서 하나만 남도록 컨테이너에서 삭제

@Qualifier로 지정

context 4.3도 필요



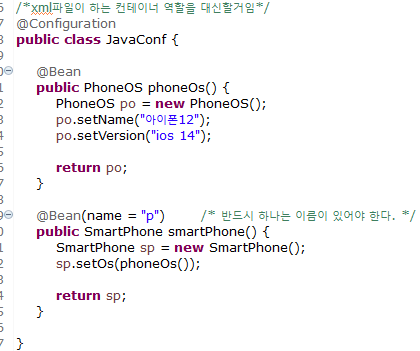


spring\_app10\_anno 일반 데이터 세팅해줌

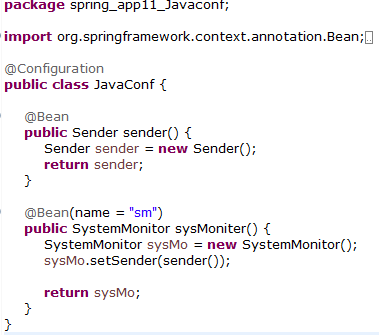
spring\_app10\_anno2 오토 와이어, 근데 오류남

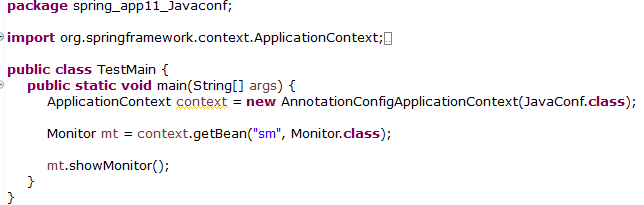
xml대신 자바 파일로 컨테이너 역할을 하도록 설정 해봄

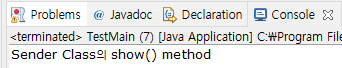
원래의 xml파일은 꽤 밑에 있어서 자바 파일로 컨테이너 설정을 하도록 하는 방법 구현함



spring\_app11\_Javaconf java 파일에서 컨테이너 역할 하도록 설정, app08 기반







spring\_app12\_conn xml 컨테이너로 db 접속 확인

spring\_app13\_jdbc 스프링에서 제공하는 jdbc 기능 사용

커넥션 풀링 방식도 추가됨

<https://mvnrepository.com/artifact/org.springframework/spring-jdbc>



지금 쓰고 있는 스프링과 버전을 맞춰서 5.2.13씀

pom에 다음의 것들 추가 – 스프링 코어, mysql, 스프링 jdbc



RowMapper: 데이터 베이스의 테이블을 구성하는 각 row를 틀에 맞게 담아서 (Mapping)

구현한 객체

- 사용: RowMapper는 인터페이스이므로 구현하지 못함(new X). 구현 객체가 따로 필요함.

-> 현재 코드 구현 상 굳이 따로 구현객체를 호출할 필요는 없으므로 익명 클래스로

구현, 사용

rowmapper를 사용해서 쿼리 실행함, 간단해짐

db 접속에 필요한 data source들을 지정하고 템플릿에 넣어서이 템플릿을 dao에서 사용



connection pooling 방식 사용하기(위의 방식 대신)

pom에

commons-io

<https://mvnrepository.com/artifact/commons-io/commons-io>

apache commons io 2.7버전 사용

commons-dbcp2

<https://mvnrepository.com/artifact/org.apache.commons/commons-dbcp2>

apache commons dbcp 2.7버전 사용

commons-pool2

<https://mvnrepository.com/artifact/org.apache.commons/commons-pool2>

apache commons pool 2.7버전 사용



트래픽 대비 커넥션을 늘려주는 것, 커넥션을 미리 만들어 놓고 커넥션이 늘어날 때마다 하나씩 할당해주는 방식



spring\_app15\_mybatis\_jdbc 13에서는 스프링으로 jdbc 접속, 여기서는 mybatis로

mybatis는 기본적으로 pooling 방식 지원됨

mybatis 3.5.5

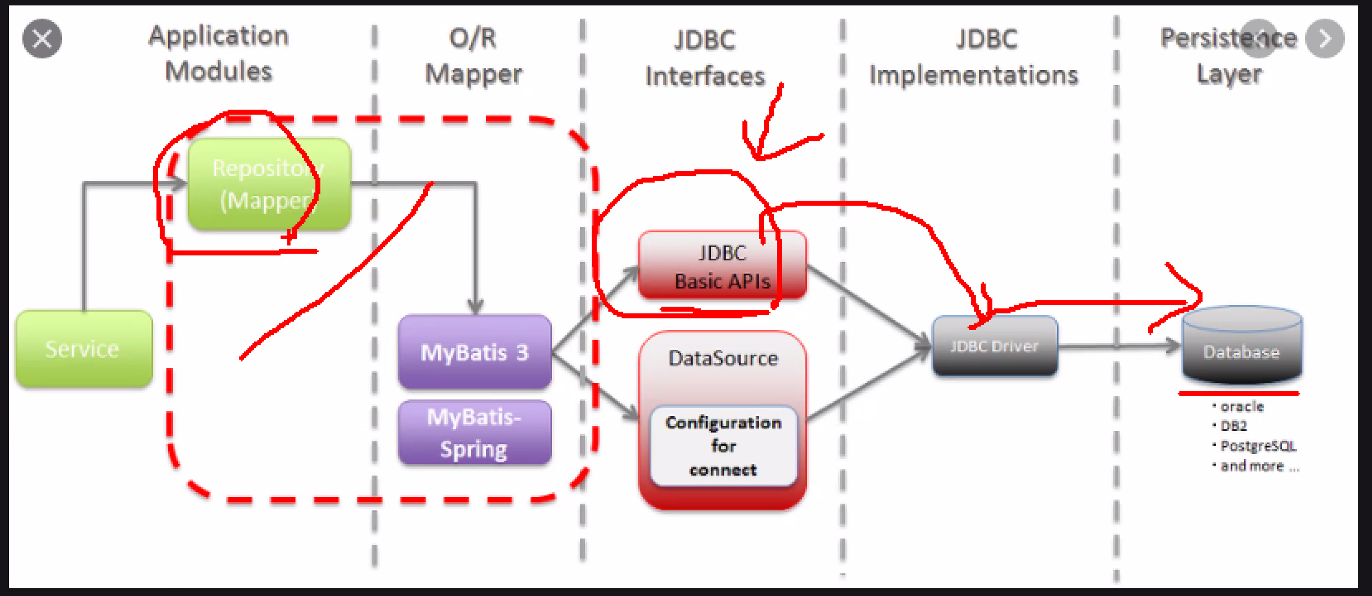
<https://mvnrepository.com/artifact/org.mybatis/mybatis>

mybatis spring 2.0.5

<https://mvnrepository.com/artifact/org.mybatis/mybatis-spring>

일반 스프링 프로젝트는 패키지가 3개 이상 내려가는 것이 권장임

kr.co.goott.app.mybatis로 패키지를 안에다가 집어넣기로 함



spring\_aop01 AOP 관련 프로젝트, spring-aop 사용

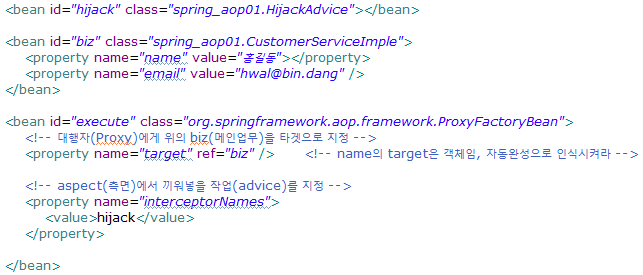
한객체의 모든 실행함수에 껴드는 단점 확인

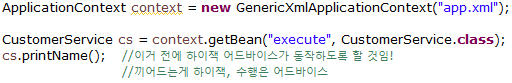
기초 부분은 위의 app 프로젝트와 똑같음

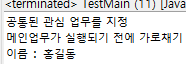
Hijack이 끼어드는 기능, Advice가 수행

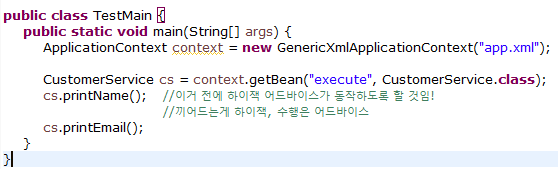
***public******class*** *HijackAdvice* ***implements*** *MethodBeforeAdvice 임플된 놈이 이전 실행임*

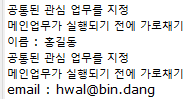
execute로 인터셉트 지정







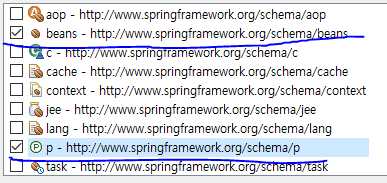


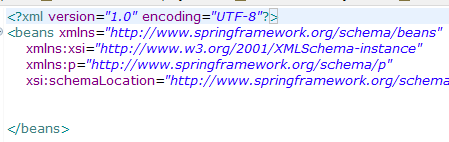


각각의 메인 로직이 시작하기 전에 끼어들고 있는 걸 볼 수 있다.

어떤 객체를 타겟으로 삼으로 모든 로직에 껴든다. 한가지만 끼어들게는 불가능

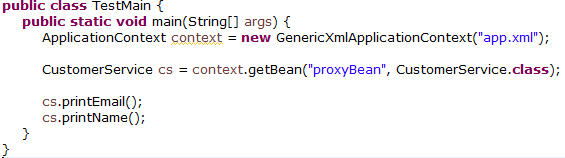
spring\_aop02\_after AspectJ, 뒤에 인터셉트

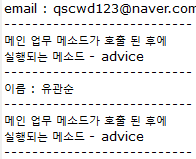
 컨테이너에 p 추가



***public******class*** *AfterInterceptorAdvice* ***implements*** *AfterReturningAdvice*

*임플된 놈이 뒤 실행임*



 매 실행 뒤 인터셉트

p를 사용해서 한번에 값을 집어넣는 기능 사용해봄, 똑같이 동작함 확인

