**11/4 김민식 OJT Day 03**

**Spring FrameWork 기본**

스프링 03 – DI(Dependency Injection)-I

참고영상 - [신입SW인력을 위한 실전 자바(Java) 스프링(Spring) 동영상과정 제 03강 DI(Dependency Injection)-I - YouTube](https://www.youtube.com/watch?v=bZjUUJM_ZqA&list=PLPvokKzUkaLeiLUDKpqRdP8jtLU8lLKgR&index=3)

3-2. 스프링 설정 파일의 이해

* 클래스 내, 필드들을 세터로 꺼내어 xml에서의 property name으로 값을 설정할 수 있다.
* MainClass.java에서 XML파일을 파싱(parsing) 하는 법

**package** com.java.ex;

**import** org.springframework.context.support.AbstractApplicationContext;

**import** org.springframework.context.support.GenericXmlApplicationContext;

**public** **class** MainClass {

**public** **static** **void** main(String[] args) {

String configLoc = "classpath:applicationCTX.xml";

AbstractApplicationContext ctx = **new** GenericXmlApplicationContext(configLoc);

MyCalculator myCalculator = ctx.getBean("myCalculator",MyCalculator.**class**);

}

}

스프링 04 – DI(Dependency Injection)-II

참고영상 - [신입SW인력을 위한 실전 자바(Java) 스프링(Spring) 동영상과정 제 04강 DI(Dependency Injection)-II - YouTube](https://www.youtube.com/watch?v=Jwns7YqAg4I&list=PLPvokKzUkaLeiLUDKpqRdP8jtLU8lLKgR&index=4)

* 목차
  + 스프링 프로퍼티 설정
  + 스프링 컨테이너의 이해

4-1. 스프링 프로퍼티 설정

* 기초데이터(필드) -> <property name = “” value=””/>
* List Type -> <property name = “list”> <list> <value></value></list></property>
* 다른 빈객체 참조 -> <property name = “Class”><ref bean=”\*\*Class”/></property>
* Setter가 있어야 참조 및 가져오기 가능

4-2. 스프링 컨테이너의 이해

**package** com.javalec.ex;

**import** org.springframework.context.support.AbstractApplicationContext;

**import** org.springframework.context.support.GenericXmlApplicationContext;

**public** **class** MainClass {

**public** **static** **void** main(String[] args) {

String configLoc = "classpath:appllicationCTX.xml";

AbstractApplicationContext ctx = **new** GenericXmlApplicationContext(configLoc); //스프링 컨테이너 생성

MyInfo myInfo = ctx.getBean("myInfo",MyInfo.**class**); //스프링 컨테이너에서 컴포넌트 가져옮

myInfo.getinfo();

ctx.close();

}

}

스프링 05 – DI활용

참고영상 - [신입SW인력을 위한 실전 자바(Java) 스프링(Spring) 동영상과정 제 05강 DI활용 - YouTube](https://www.youtube.com/watch?v=wqHBAmIZcvg&list=PLPvokKzUkaLeiLUDKpqRdP8jtLU8lLKgR&index=5)

* 목차
  + 의존 관계
  + DI 사용에 따른 장점

5-1. 의존 관계

* DI는 의존 주입
* 객체에 의존을 주입하는 방법
  + 생성자 이용 -> <constructor-arg><value></value></constructor-arg>
  + Setter를 이용

5-2. DI사용에 따른 장점

* 유지보수에 커다란 장점
* 규모가 큰 프로젝트에서 DI를 이용한 개발의 장점이 발휘
* Java파일이 수정없이 스프링 설정 파일만을 수정하여 부품들을 생성/조립 가능
  + Interface사용이 많음

스프링 06 – DI설정 방법

참고영상 - [신입SW인력을 위한 실전 자바(Java) 스프링(Spring) 동영상과정 제 06강 DI설정 방법 - YouTube](https://www.youtube.com/watch?v=MabPKw0HapE&list=PLPvokKzUkaLeiLUDKpqRdP8jtLU8lLKgR&index=6)

* 목차
  + XML파일을 이용한 DI설정 방법
  + JAVA를 이용한 DI 설정 방법
  + XML과 JAVA를 같이 사용

6-1. XML파일을 이용한 DI설정 방법

* Namespace이용가능 <bean id=”\*\*” class=”com.aa.bb.cc” c:papaName=”ddd” p:sisterName=”fff”/> \* 상단의 beans 태그에 네임스페이스 사용여부 작성
* Config파일(CTX.xml)을 통해 같은 객체 사용 및 여러 개의 config파일 사용 가능

6-2. JAVA를 이용한 DI 설정 방법

* @Configuration – 이 클래스는 스프링 설정에 사용되는 클래스입니다. 라고 명시
* @Bean – 객체 생성

package com.javalec.ex;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class MainClass {

public static void main(String[] args) {

AnnotationConfigApplicationContext ctx = new AnnotationConfigApplicationContext(ApplicationConfig.class);

Student student1 = ctx.getBean("student1",Student.class);

System.***out***.println(student1.getName());

System.***out***.println(student1.getAge());

System.***out***.println(student1.getHobbys());

Student student2 = ctx.getBean("student2",Student.class);

System.***out***.println(student2.getName());

System.***out***.println(student2.getAge());

System.***out***.println(student2.getHobbys());

ctx.close();

}

}

6-3. XML과 JAVA를 같이 사용

* XML파일에 JAVA파일을 포함시켜 사용 하는 방법

<context:annotation-config /> - config anotation이 된 클래스를 가져온다.

<bean class=”com.javalec.ex.ApplicationConfig”/>

* JAVA파일에 XML파일을 포함시켜 사용 하는 방법

@Configurable

@ImportReasource

스프링 07 – 생명 주기(life cycle) 와 범위(scope)

참고영상 - [신입SW인력을 위한 실전 자바(Java) 스프링(Spring) 동영상과정 제 07강 생명주기(life cycle) - YouTube](https://www.youtube.com/watch?v=zKI4GWT5Xzs&list=PLPvokKzUkaLeiLUDKpqRdP8jtLU8lLKgR&index=7)

* 목차
  + 스프링 컨테이너 생명 주기
  + 스프링 빈 생명 주기
  + 스프링 빈 범위

7-1. 스프링 컨테이너 생명 주기

1. 스프링 컨테이너 생성 -> GenericXmlApplicationContext ctx = new GenericXmlApplicationContext();
2. 스프링 컨테이너 설정 -> ctx.load(“classpath:applicationCTX.xml”);   
    ctx.refresh();
3. 스프링 컨테이너 사용 -> Student student = ctx.getBean(“student”,Student.class);
4. 스프링 컨테이너 종료 -> ctx.close();

7-2. 스프링 빈 생명 주기

1. Implements InitializingBean(초기화 과정) , DisposableBean (소멸 과정) – 인터페이스를 활용한 방법
   1. afterPropertiesSet() – 빈 초기화 과정에서 호출
   2. destroy() – 빈 소멸 과정에서 생성
2. @PostConstruct, @PreDestroy – 어노테이션을 이용한 방법
   1. @PostConstruct – 빈 초기화 과정에서 호출
   2. @PreDestroy – 소멸과정에서 호출

7-3. 스프링 빈 범위(scope)

* 해당하는 객체가 어디까지 영향을 미치는지 결정하는 것
* Scope=”singleton” : Default 값 , 하나

스프링 08 – 외부 파일을 이용한 설정

참고영상 - [신입SW인력을 위한 실전 자바(Java) 스프링(Spring) 동영상과정 제 08강 외부파일을 이용한 설정 - YouTube](https://www.youtube.com/watch?v=9cDHggH0ENA&list=PLPvokKzUkaLeiLUDKpqRdP8jtLU8lLKgR&index=8)

* 목차
  + Environment 객체
  + 프로퍼티 파일을 이용한 설정
  + 프로파일(profile) 속성을 이용한 설정

8-1. Environment 객체

* Context ->(ctx.getEnvironment()) Environment ->(env.getPropertySources()) PropertySourece (프로퍼티 추가 및 추출)
  + 추가 : propertySources.addLast()
  + 추출 : env.getProperty

8-2. 프로퍼티 파일을 이용한 설정

1. Xml파일을 이용한 설정

* .xml파일에 Namespaces에서 context 체크
* <context:property-placeholder location=”classpath:admin~,classpath:sub\_admin~”/>

1. JAVA파일을 이용한 설정

* Public static PropertySourcesPlaceholderConfigurer Properties(){

}

* @Value(${sub\_admin.pw})로 세팅 가능

8-3. 프로파일(profile) 속성을 이용한 설정

1. Xml파일을 이용

* ctx.getEnvironment().setActiveProfiles(“dev”); 라면 .xml 파일에 beans안에 profile=”dev”가 추가된 xml이 실행

1. JAVA파일을 이용

* @Profile(“Run”)으로 설정 가능

스프링 09 – AOP(Aspect Oriented Programming) - I

참고영상 - [신입SW인력을 위한 실전 자바(Java) 스프링(Spring) 동영상과정 제 09강 AOP-I - YouTube](https://www.youtube.com/watch?v=2F8K9BLgvjE&list=PLPvokKzUkaLeiLUDKpqRdP8jtLU8lLKgR&index=9)

* 목차
  + AOP란?
  + XML기반의 AOP구현

9-1. AOP란?

* 관점 지향 프로그래밍
* 핵심 기능과 공통 기능을 분리, 공통 기능을 필요로 하는 핵심 기능들에서 사용하는 방식
* AOP용어
* Aspect : 공통 기능
* Advice : Aspect의 기능 자체
* Joinpoint : Advice를 적용해야 되는 부분( ex, 필드, 메소드 ) (스프링에서는 메소드만 해당)
* Pointcut : Joinpoint의 부분으로 실제로 Advice가 적용된 부분
* Weaving : Advice를 핵심 기능에 적용 하는 행위
* 스프링에서 AOP 구현 방법 : proxy를 이용
* 호출부(Client) -> Proxy(대행) -> Target(핵심기능)
* 스프링에서 AOP 구현 방식
* XML 스키마 기반의 AOP구현
* @Aspect 어노테이션 기반의 AOP 구현 (자바코드)

9-2. XML 기반의 AOP구현

* 작업 순서
  + 1. 의존 설정(pom.xml)
    2. 공통 기능의 클래스 제작 – Advice 역할 클래스
    3. XML설정 파일에 Aspect 설정
* Advice 종류
* <aop:before> : 메소드 실행 전에 advice 실행
* <aop:after-returning> : 정상적으로 메소드 실행 후에 advice 실행
* <aop:after-throwing> : 메소드 실행 중 exception 발생시 advice 실행
* <aop:after-around> : 메소드 실행 중 exception 발생하여도 advice 실행

질문사항

1. ~~IDE(Eclipse), JDK, VScode, Spring Framework 등 설치를 임의로 혹은 정해진 규격 혹은 ver (ex.  JDK 11 or 1.8 등) 이 있는지 ※ 이미 Installed (자바 11ver. 등)~~
2. 일과 후 개인시간에 중점적으로 공부하면 좋을 것들
3. ~~MarkDown으로 강의내용 정리해도 될지.~~
4. 노트북 휴대 가능 여부 (교육 용도)
5. 프로토콜의 직접적인 쓰임새
6. RequestParam을 사용하지 않고 Thymeleaf의 th:href="@{/movielist/detail(id=${m.id})}" 와 같이 처리할 때의 차이점
7. 회사내 사용하는 참고할만한 코드나 분석할만한 코드를 보여주실 수 있을지
8. XML을 이용한 Spring 사용과 JAVA를 이용한 Spring 사용의 비율`