

스프링 프레임워크 ver.1

스프링 프레임워크 = 스프링 3버전

스프링 부트 = 스프링 4버전 이라 생각하면 편하다.

※프레임워크 : 추상적으로 개발을 하려는것을 정의해 놓은 틀(Frame).

-개발자는 구현을 하기만 하면 된다.

-프레임워크가 없다면 개발이 외에 모든 것을 개발자가 행해야 하므로 일률이 떨어질 수밖에 없다.

★스프링 프레임워크의 장점★

- 복잡함에 반기를 들어서 만들어진 프레임워크
- 프로젝트 전체 구조를 설계할 때 유용한 프레임워크
- 다른 프레임워크들의 포용(여러 프레임워크를 혼용해서 사용가능)
- 개발 생산성과 개발도구의 지원

스프링 프레임워크 특징

- POJO 기반의 구성
- 의존성 주입(DI)을 통한 객체 간의 관계 구성
- AOP (Aspect - Oriented Programming) 지원
- 편리한 MVC 구조
- WAS의 종속적이지 않은 개발 환경

★★특징

★POJO(Plain Old Java Object) 기반의 구성★

=> 오래되고 간단한 자바 객체 기반의 구성 => 자바 사용 가능!(간단하게)

오래된 방식의 간단한 자바 객체라는 의미이며, JAVA 코드에서 일반적으로 객체를 구성하는 방식을 스프링 프레임워크에서 그대로 사용할 수 있다는 말이다.

★의존성 주입(DI)을 통한 객체 간의 관계 구성★

의존성의 예

예 1)

(레스토랑) 이 (쉐프)한테 의존한다. 레스토랑은 셰프가 필수다. 객체라고 보면 레스토랑 객체에 셰프가 무조건 존재.

레스토랑 클래스 내에서 셰프 객체를 생성하면 레스토랑 클래스는 해당 객체 밖에 사용하지 못한다. (의존성을 가진다.)

레스토랑 클래스 안에 멤버 셰프 new를 사용해 객체를 생성 하였을 때 단단하게 의존하게 되어 있다.

-> 단단하면 유연성이 떨어진다. (유연한 개발이 불가능)

※느슨하게 해야하는데 이걸 이렇게 만들어 주는 것을 의존성 주입이라 한다.

예 2)

칼과 요리를 생각 (요리)는 (칼)이 필요하다. 요리 속(new 칼) => 그러면 단단하다.

필요한 종류의 칼을 받아올 수 있어야 한다 -> 느슨할 필요가 있다

※ 느슨하게 만들 필요가 있다. 의존성 주입이 필요하다.

★의존성 주입(DI)★

의존성

의존성(**Dependency**)이란 하나의 객체가 다른 객체 없이 제대로 된 역할을 할 수 없다는 것을 의미한다. 예를 들어 **A** 객체가 **B** 객체 없이 동작이 불가능한 상황을 '**A**'가 '**B**'에 의존적이다라고 표현한다.

주입

주입(**Injection**)은 말 그대로 외부에서 밀어 넣는 것을 의미한다.

예를 들어 어떤 객체가 필요한 객체를 외부에서 밀어 넣는 것을 의미한다.

주입을 받는 입장에서는 어떤 객체인지 신경 쓸 필요가 없고 어떤 객체에 의존하든 자신의 역할은 변하지 않게 된다.

**의존 직접 생성시

a → → → → → → → → → → b ⇒ new

a 객체에서 b 객체를 직접 생성 = 결합성이 강하다.

**의존성 주입

a → → → → ??? ↔ ↔ ↔ ↔ b

a가 필요로 할 때마다 ???에서 b 객체를 주입

즉 a는 b가 필요하다는 신호만 보내고, b 객체를 주입하는 것은 외부에서 이루어짐
의존성 주입 방식을 사용하기 위해서는 ???라는 존재가 필요하게 된다.

스프링 프레임워크에서는 ApplicationContext가 ???라는 존재이며,

필요한 객체들을 생성하고, 필요한 객체들을 주입해주는 역할을 한다.

따라서 개발자들은 기존의 프로그래밍과 달리 객체와 객체를 분리해서 생성하고.

이러한 객체들을 역(wiring)작업의 형태로 개발하게 된다.

ApplicationContext가 관리하는 객체들을 '빈(Bean)'이라고 부르고,

빈과 빈 사이의 의존 관계를 처리 방식으로는 **XML**방식, 어노테이션 설정, **JAVA** 설정방식을 이용할 수 있다.

DI(Dependency Injection)란?

- 스프링의 의존 주입
 - applicationContext에서 스프링 컨테이너를 만들고 스프링 컨테이너에서 'Bean'을 만들어 낸다.

액체가 모여있는곳 : 컨테이너 =>getBean()=> **Bean**

빈의 범위

싱글톤

스프링 컨테이너에서 생성된 (**Bean**) 객체의 경우 동일한 타입에 대해서는 기본적으로 한 개만 생성이 되며, **getBean()** 메소드로 호출될 때 동일한 객체가 반환된다.

프로토타입(Prototype)

싱글톤 범위와 반대의 개념 프로토 타입의 경우 개발자가

별도 생성 **scope="prototype"**을 주면된다.

의존객체 자동주입

스프링 설정 파일에서 constructor property 태그로 의존대상을 명시하지 않아도
스프링 컨테이너가 찾아 자동으로 객체를 주입해주는 기능이다.

@Autowired

주입하려고 하는 객체의 타입이 일치하는 객체를 자동으로 주입한다. **property**설정, 의존 설정을 할때 자바코드에 **@Autowired**를 사용하면스프링 컨테이너에서 **Bean** 객체중에 객체 타입 **A-> A, B-> B, C-> C** 해당하는 객체를 찾아서자동으로 주입해준다.

@Resource

주입하려고 하는 객체의 이름이 일치하는 객체를 가져와서 사용한다. 생성자에는 쓰지 못하고 .
Property 나 Method 에 사용 할 수 있다.

의존객체 선택

다수의 빈(Bean)객체 중 의존 객체의 대상이 되는 객체를 선택

@Inject

inject은 Autowired와 거의 같지만 required 속성을 지원하지 않는다.
@Named(value=id명)를 사용하면 해당 Id를 찾아간다

@Qualifier("객체명") 자동 주입시 찾아가는 이름

-Lombok 어노테이션

@AllArgsConstructor

전체 필드를 자동으로 주입해주는 어노테이션

@RequiredArgsConstructor => final이거나 nonNull인 어노테이션을 가진 녀석들만 초기화

★AOP의 지원★

관점 지향 프로그래밍.

예) 결제 관련해서 프로그래밍을 할 때 결제 관련 내용에만 신경쓰게 프로그래밍을 하는
프로그래밍 기법

만약 결제 - 마이페이지 - 포인트 의 업무가 있다고 할때 그 업무에만 집중하고 싶기
때문에 시점을 정해서 AOP객체가 원하는 필요한 매소드나 로그를 사용해준다.

결제 마이 포인트등 = 종단 관심사 => DB나 로그등을 횡단관심사 라고 한다. 즉
종단관심사에 집중해서 관점을 잡아 프로그래밍한다

관점 지향 프로그래밍이란

좋은 개발환경에서는 개발자가 비지니스 로직에만 집중할 수 있게 한다.

스프링 프레임워크는 반복적인 코드를 제거해줌으로써 핵심 비지니스 로직에만
집중할 수 있는 방법을 제공한다.

보안이나 로그, 트랜잭션, 예외처리와 같이 비지니스 로직은 아니지만, 반드시 처리가
필요한 부분을 횡단 관심사(**cross-concern**)이라고 한다.

스프링 프레임워크는 이러한 횡단 관심사를 분리해서 제작하는 것이 가능하고
횡단 관심사를 모듈로 분리하는 프로그래밍을 **AOP**라고 한다.
이를 통해서 **3가지의** 이점이 생긴다.

1) 핵심 비지니스 로직에만 집중하여 코드 개발

2) 각 프로젝트마다 다른 관심사 적용시 코드 수정 최소화

3) 원하는 관심사의 유지보수가 수월한 코드 구성 가능

★트랜잭션의 지원★

DB작업 시 트랜잭션관리를 매번 상황에 맞게 코드로 작성하지 않고,

어노테이션이나 **XML**로 트랜잭션 관리를 설정할 수 있다.

★단위 테스트★

전체 **Application**을 실행하지 않아도 기능별 단위 테스트가 용이하기 때문에

버그를 줄이고 개발 시간을 단축할 수 있다.

스프링 프로젝트 생성

- 1. 프로젝트 생성
 - Group Id : 전체적인 그룹
 - Artifact Id : 부분적 그룹
 - 스프링은 모듈로 구성되어 있다. 모듈 하나하나 프로젝트들은 모두 artifact id에 속하고 그 모듈들은 스프링 버전에 맞는 group id에 둑여 있다.

프로젝트 폴더 -> src -> main/test -> java/resources

- java 폴더 : 자바 파일들
- resources 폴더 : 자원을 관리하는 폴더 스프링 설정파일(XML)도 포함

프로젝트 기본 경로

- 1) src/main/java : 서버단 JAVA 파일
- 2) src/test/java : 단위 테스트를 위한 JAVA 파일
- 3) src/main/resources : src/main/java 관련 설정 파일
- 4) src/test/resources : src/main/test 관련 설정 파일
- 5) src/main/webapp/WEB-INF/views : jsp, html 파일 경로
- 6) pom.xml : 라이브러리 의존성 관리
- 7) src/main/webapp/WEB-INF/spring/appServlet/servlet-context.xml : 웹과 관련된 스프링 설정 파일
- 8) src/main/webapp/WEB-INF/spring/root-context.xml : 스프링 객체 관련 설정 파일

pom.xml

필요한 모듈을 가져오기 위한 파일 역할 : 스프링은 모듈을 지원해주는 프레임 워크

원격 repository에서 필요한 모듈(라이브러리)를 사용하기 위해서 pom.xml 파일에 명시만 해 놓으면 자동으로 다운로드해서 쉽게 사용할 수 있게 해준다.