02_Spring IOC/DI

IOC(Inversion Of Control, 제어 반전)

- IOC란, 프로그램을 구동하는데 필요한 객체에 대한 생성, 변경 등의 관리를 프로그램을 개발하는 사람이 아닌 프로그램을 구동하는 컨테이너에서 직접 관리하는 것을 말함.

- **스프링은 IOC를 통해 구동 시 필요한 객체의 생성부터 생명 주기까지 해당 객체에 대한 관리를 직접 수행한다.**

- 개발자가 직접 객체를 생성할 수 있지만 해당 권한을 컨테이너에 맡김으로써 소스 코드 구현의 시간을 단축할 수 있다.

Bean(빈)

- Spring에서 IOC를 이용해 직접 생성하고 생명 주기 제어를 담당하는 객체

(개발자가 생성하고 관리하는 객체는 Instance)

IOC 관련 어노테이션

- 특정 클래스에 IOC 관련 어노테이션을 작성
  → Bean으로 등록되어 서버 실행 시 Spring Container에 의해 Bean(객체)으로 만들어짐

@Component	-객체(컴포넌트)를 나타내는 일반적인 타입으로 <bean> 태그와 동일한 역할
@Repository	-퍼시스턴스(persistence) 레이어, 영속성을 가지는 속성(파일, 데이터베이스)를 가진 클래스 ex) Data Access Object Class
@Service	-서비스 레이어, 비즈니스 로직을 가진 클래스 ex) Service Class
@Controller	-프리젠테이션 레이어, 웹 애플리케이션에서 View에서 전달된 웹 요청과 응답을 처리하는 클래스 ex) Controller Class

💡 서버 실행 시 Spring Container가 **Component Scan(Bean Scanning)** 을 수행하여

@Component, @Repository, @Service, @Controller 어노테이션이 붙은 클래스를 모두 찾아 **Bean으로 등록(객체로 생성)**

Spring Legacy Project는 <context:component-scan> 태그
Spring Boot Project는 @CompnentScan 어노테이션을 이용해 기능 수행

DI(Dependency Injection, 의존성주입)

- DI란, IOC 구현의 핵심 기술로
  IOC를 통해 Spring이 생성하고 관리하는 Bean(객체)을
  필요한 곳에 주입하는 것
  (객체 생성을 개발자가 아닌 Spring에게
  의존하고
  생성된 객체를 개발자가 필요한 곳에
  주입하는 것**)**

- 이렇게 의존성을 주입 받게 되면 이후 해당 객체를 수정해야 할 상황이 발생했을 때
  소스 코드의 수정을 최소화 할 수 있음

- **객체간의 종속 관계 (결합도)를 약화 시킬 수 있음**

(결합도 **:** 한 클래스에서 필드 객체를 생성할 때 발생하는 두 객체 간의 관계를 말하며,

각 객체간의 내용이 수정될 경우 영향을 미치는 정도를 나타냄)

DI 관련 어노테션

@Autowired	-정밀한 의존 관계 주입(DI)이 필요한 경우에 유용하다. -@Autowired는 필드 변수, setter 메소드, 생성자, 일반 메소드에 적용 가능하다. -의존하는 객체를 주입할 때 주로 Type을 이용하게 된다. -@Autowired 는 <property>, <constructor-arg> 태그와 동일한 역할을 한다.
@Qualifier	@Autowired와 함께 쓰이며, 한 프로젝트 내에 @Autowired로 의존성을 주입하고자 하는 객체가 여러 개 있을 경우, @Qualifier(“name”)를 통해 원하는 객체를 지정하여 주입할 수 있다.