1. IoC(Inversion of Control)

의존성에 대한 제어권은 원래 자기 자신에게 있었지만 스프링에서는 자기 자신 이외의 누군가가 객체를 주입해준다. (`누군가 내 생성자에 주입해 주겠지` 하는 생각)

1. IoC 컨테이너 – ApplicationContext(BeanFactory)

직접 쓸 일은 많지 않다. 실제로 보통 코드에서 보이지 않는다.

빈(객체)을 만들고 엮어주며 제공해준다.

오로지 빈만 관리할 수 있다. 빈의 의존성을 관리해 준다.

예) 리파지토리로 생성된 객체(빈)을 찾아서 컨트롤러의 생성자에 주입해준다.

I OC 컨테이너에서 빈을 꺼내서 사용하는게 @Autowired 이다.

애노테이션으로 컨트롤러, 컴포넌트, 서비스, 리파지토리가 붙어 있으면 빈이다.

1. 빈(Bean)

스프링 IoC 컨테이너가 관리하는 객체

* 등록 방법
  + Component Scanning
    - @Component
      * @Repository
      * @Service
      * @Controller

애노테이션 자체는 주석과 같다. 기능이 없다.

애노테이션을 마커로 사용해서 애노테이션을 처리하는 프로세서들이 있는 것이다.

@Component Scan을 처리하는 프로세서가 있다.

스프링 프로젝트의 기본 클래스가 속한 패키지 아래의 컴포넌트만 찾아서 빈으로 등록해 준다.

직접 등록 : @Bean 애노테이션 지정, @Configuration 애노테이션에 속한 곳에서 등록 가능.

예)

@Bean

public String keesun() {

return “keesun”;

}

빈 꺼내 쓰는 방법 : @Autowired String keesun; 이렇게 쓰면 IoC 컨테이너에서 알아서 넣어준다.

1. 의존성 주입 (DI)

@Autowired / @Inject 를 붙여주면 된다.

* 생성자 : 1순위 - 객체에 반드시 필요한 의존성일 때 생성자에 붙인다.
* 필드 : 3순위 - 세터가 없으면 필드에 붙인다.
* Setter : 2순위 - Setter가 있다면 Setter에 붙인다.

빈이 되는 클래스에 생성자가 하나만 있고, 생성자의 매개변수 타입이 빈으로 등록되어 있으면 @Autowired가 없더라도 자동으로 빈을 주입해 준다.

1. AOP (Aspect Oriented Programming) – 관점 지향 프로그래밍

single responsibility principle 객체 지향 원칙대로 코딩하도록 하는 코딩 기법

흩어진 코드를 한 곳으로 모으는 프로그래밍 기법

예) 비슷한 메서드를 템플릿화해서 다른 부분을 템플릿에 끼워넣는 방식

프로그래밍 방식

* 바이트 코드 조작 (.class 수정)
* 프록시 패턴 : 상속을 통함.

1. PSA (Portable Service Abstraction) - 이식 가능한 서비스 추상화

예) 스프링 트랜잭션, 캐시, 웹 MVC

잘 만든 인터페이스

스프링 api는 90% 이상 잘 만든 인터페이스. 추상화된 api. 구현체가 바뀌더라도 추상화된 인터페이스 코드는 수정하지 않아도 됨.

스프링의 3가지 큰 특징

* IoC
* AOP
* PSA