스터디 4주차

2021.09.05

섹션 2: 스프링 핵심 원리 이해1 – 예제 만들기

섹션 3: 스프링 핵심 원리 이해2- 객체 지향 원리 적응

섹션 4: 스프링 컨테이너와 스프링 빈

인프런 김영한 스프링 완전 정복 로드맵 이용

해당 PPT는 강의 내용을 요약, 정리하여 작성하였습니다.

잘못된 내용은 피드백주세요

목 차

SRP

"구체화에 의존 X, 추상화에 의존 O"

구현 객체 생성 및 연결하는 역할

&

실행하는 역할

AppConfig

MemberServiceImpl

OrderServiceImpl

관심사 분리

DIP

"한 클래스는 하나의 책임만 가져야 한다"

사용 영역

```
private final MemberRepository memberRepository = new MemoryMemberRepository();
```

마치 상대배우를 직접 고르는 것과 같다

```
private final MemberRepository memberRepository;

public MemberServiceImpl(MemberRepository memberRepository) {
    this.memberRepository = memberRepository;
}
```

MemberServieImpl 클래스

MemberRepository 인터페이스에만 의존

구성 영역

```
@Bean
public MemberService memberService() {
    return new MemberServiceImpl(memberRepository());
}
```

AppConfig 클래스

구체 클래스 선택

MemoryMemberRepository 객체 생성하여 연결

DI(Dependency Injection)

의존관계 주입

의존관계를 외부(AppConfig)에서 주입해주는 모습

역할에 따른 구현을 한눈에 볼 수 있다.

애플리케이션 전체 구성이 어떻게 되어있는지 빠르게 파악 가능

OCP

"소프트웨어 요소는 확장에는 열려 있으나 변경에는 닫혀 있어야 한다"

FixDiscountPolicy 객체 → RateDiscountPolicy 객체로 변경

사용 영역

```java
private final DiscountPolicy discountPolicy;

public OrderServiceImpl(MemberRepository memberRepository, DiscountPolicy discountPolicy) {
    this.memberRepository = memberRepository;
    this.discountPolicy = discountPolicy;
}
```

OrderServiceImpl 클래스

DIP를 지켰기 때문에

OrderServiceImpl 코드 변경 X

구성 영역

```java
@Bean
public OrderService orderService() {
    return new OrderServiceImpl(memberRepository(), discountPolicy());
}

@Bean
public DiscountPolicy discountPolicy() {
//    return new FixDiscountPolicy();
    return new RateDiscountPolicy();
}
```

객체 변경

AppConfig 클래스

AppConfig 코드만 변경

IoC 컨테이너

IoC와 DI를 해주는 컨테이너

EX. AppConfig, Junit

IoC(Inversion of Control)

제어의 역전

프로그램의 제어 흐름을 직접 제어하는 것이 아닌 외부에서 관리하는 것

DI

정적인 클래스 의존관계

Import 코드만 보고도 쉽게 판단 가능

코드를 실행하지 않아도 파악 가능

```
import hello.core.discount.DiscountPolicy;
import hello.core.member.MemberRepository;
import hello.core.member.Member;

public class OrderServiceImpl implements OrderService {
            . . .
}
```

동적인 클래스 의존관계

코드 실행 시점에서 실제 생성 및 연결된 객체 파악된다.

스프링 컨테이너

스프링 컨테이너의 최상위 클래스인 BeanFactory

직접 사용할 일이 별로 없어서 ApplicationContext로 주로 정의한다.

- BeanFactory의 기능(ex. getBean())을 상속 받아서 Bean 관리기능 + 편리한 부가 기능 제공

  new AnnotationConfigApplicationContext(AppConfig.class)

BeanDefinition(빈 설정 메타정보)를 추상화해서 스프링 빈 생성

→ factoryBeanName(ex. AppConfig)를 통해 스프링 빈 등록

- @Configuration, @Bean 사용
- 어노테이션 방식 말고도 XML로 생성 가능
  - Bean에 관한 정보가 명확하게 들어간다.

    new AnnotationConfigApplicationContext(AppConfig.xml)

```
<bean id="memberRepository" class="hello.core.member.MemoryMemberRepository" />
```

스프링 빈 이름은 메서드 명으로 저장

ApplicationContext.getBean()을 이용하여 저장된 메서드를 찾는다.

이름으로 조회, 이름 없이 타입만으로 조회, 구체 타입으로 조회 가능

컨테이너에 등록된 빈 조회

```java
@Test
@DisplayName("모든 빈 출력하기")
void findAllBean() {
    String[] beanDefinitionNames = ac.getBeanDefinitionNames();
    for (String beanDefinitionName : beanDefinitionNames) {
        Object bean = ac.getBean(beanDefinitionName);
        System.out.println("name = " + beanDefinitionName + " object = " + bean);
    }
}
```

스프링에 등록된 모든 빈의 이름 조회

```java
@Test
@DisplayName("애플리케이션 빈 출력하기“)
void findApplicationBean() {
    String[] beanDefinitionNames = ac.getBeanDefinitionNames();

    for (String beanDefinitionName : beanDefinitionNames) {
        BeanDefinition beanDefinition = ac.getBeanDefinition(beanDefinitionName);

        if (beanDefinition.getRole() == BeanDefinition.ROLE_APPLICATION) {
            Object bean = ac.getBean(beanDefinitionName);
            System.out.println("name = " + beanDefinitionName + " object = " + bean);
        }
    }
}
```

빈에 대한 메타 정보 찾기

등록된 빈 중 사용자가 정의한 빈 찾기

출력모습

```
name = appConfig object = hello.core.AppConfig$$EnhancerBySpringCGLIB$$a7e90ddf@2ceb80a1
name = memberService object = hello.core.member.MemberServiceImpl@4b45dcb8
name = memberRepository object = hello.core.member.MemoryMemberRepository@7216fb24
name = orderService object = hello.core.order.OrderServiceImpl@2072acb2
name = discountPolicy object = hello.core.discount.RateDiscountPolicy@50ecde95
```

스프링 빈 조회 – 동일한 타입이 둘 이상

```java
@Test
@DisplayName("타입으로 조회시 같은 타입이 둘 이상 있으면, 중복 오류가 발생한다.")
void findBeanByTypeDuplicate() {

    //검증
    assertThrows(NoUniqueBeanDefinitionException.class, () -> ac.getBean(MemberRepository.class));
}
```

타입이 중복되어 있음을 알려주는 오류

```java
@Test
@DisplayName("타입으로 조회시 같은 타입이 둘 이상 있으면, 빈 이름을 지정하면 된다.")
void findBeanByName() {
    MemberRepository memberRepository = ac.getBean("memberRepository1", MemberRepository.class);
    assertThat(memberRepository).isInstanceOf(MemberRepository.class);
}
```

```java
@Test
@DisplayName("특정 타입을 모두 조회하기")
void findAllBeanByType() {
    Map<String, MemberRepository> beansOfType = ac.getBeansOfType(MemberRepository.class);
    for (String key : beansOfType.keySet()) {
        System.out.println("key = " + key + " value = " + beansOfType.get(key));
    }

    System.out.println("beansOfType = " + beansOfType);

    //검증 - beansOfType의 크기가 2 여야 한다.
    assertThat(beansOfType.size()).isEqualTo(2);
}
```

해당 타입의 모든 빈 찾기

스프링 빈 조회 - 상속관계

부모 타입으로 조회하면 자식 타입도 함께 조회

모든 자바 객체의 최상위 클래스 Object로 조회하면 모든 스프링 빈 조회 가능

```java
@Test
@DisplayName("부모 타입으로 모두 조회하기 - Object")
void findAllBeanByObjectType() {
    Map<String, Object> beansOfType = ac.getBeansOfType(Object.class);
    for (String key : beansOfType.keySet()) {
        System.out.println("key = " + key + " value = " + beansOfType.get(key));
        //스프링에 있는 여러가지 모든 bean들이 출력된다!
    }
}
```

ApplicationContextExtendsFindTest 클래스

출력모습

```
key = org.springframework.context.annotation.internalConfigurationAnnotationProcessor value = org.springframework.context.annotation.ConfigurationClassPostProcessor@1b765a2c
key = org.springframework.context.annotation.internalAutowiredAnnotationProcessor value = org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor@2e8e8225
key = org.springframework.context.annotation.internalCommonAnnotationProcessor value = org.springframework.context.annotation.CommonAnnotationBeanPostProcessor@6ebf0f36
key = org.springframework.context.event.internalEventListenerProcessor value = org.springframework.context.event.EventListenerMethodProcessor@18920cc
key = org.springframework.context.event.internalEventListenerFactory value = org.springframework.context.event.DefaultEventListenerFactory@2807bdeb
key = applicationContextExtendsFindTest.TestConfig value = hello.core.beanfind.ApplicationContextExtendsFindTest$TestConfig$$EnhancerBySpringCGLIB$$6ebf8c66@72c28d64
key = rateDiscountPolicy value = hello.core.discount.RateDiscountPolicy@6492fab5
key = fixDiscountPolicy value = hello.core.discount.FixDiscountPolicy@2c5529ab
key = environment value = StandardEnvironment {activeProfiles=[], defaultProfiles=[default], propertySources=[PropertiesPropertySource@967323951 {name='systemProperties', properties={sun.desktop=wi
, java.specification.name=Java Platform API Specification, java.vm.specification.vendor=Oracle Corporation, java.awt.graphicsenv=sun.awt.Win32GraphicsEnvironment, user.script=, sun.management.compi
key = systemProperties value = {sun.desktop=windows, awt.toolkit=sun.awt.windows.WToolkit, java.specification.version=11, sun.cpu.isalist=amd64, sun.jnu.encoding=MS949, java.class.path=C:\Program F
, java.specification.name=Java Platform API Specification, java.vm.specification.vendor=Oracle Corporation, java.awt.graphicsenv=sun.awt.Win32GraphicsEnvironment, user.script=, sun.management.compi
key = systemEnvironment value = {USERDOMAIN_ROAMINGPROFILE=DESKTOP-SGCS8F9, LOCALAPPDATA=C:\Users\min\AppData\Local, PROCESSOR_LEVEL=6, USERDOMAIN=DESKTOP-SGCS8F9, FPS_BROWSER_APP_PROFILE_STRING=In
key = applicationStartup value = org.springframework.core.metrics.DefaultApplicationStartup@2c532cd8
key = org.springframework.context.annotation.ConfigurationClassPostProcessor.importRegistry value = []
key = messageSource value = Empty MessageSource
key = applicationEventMulticaster value = org.springframework.context.event.SimpleApplicationEventMulticaster@294e5088
key = lifecycleProcessor value = org.springframework.context.support.DefaultLifecycleProcessor@51972dc7
```