

예제로 배우는 스프링 프레임워크 입문

(ver 2019.02)

스프링이 뭔데?

스프링 프레임워크(이하, 스프링)를 사용한 예제 코드를 보며 스프링의 주요 철학과 기능을 빠르게 학습합니다.

강좌 목표

- 실제 코드를 보며 스프링 프레임워크에 대해 소개합니다.
- 스프링 프레임워크가 개발자에게 주는 가치를 이해합니다.
- 스프링 프레임워크 주요 기능을 짧은 시간 안에 간략하게 이해하는 것을 목표로 합니다.

강좌 계획

1. 강의 소개
 - a. 프로젝트 세팅
 - b. 프로젝트 살펴보기
 - c. 프로젝트 과제 풀이
2. Inversion of Control
 - a. IoC 소개
 - b. IoC (Inversion of Control) 컨테이너
 - c. 빈 (Bean)
 - d. 의존성 주입 (Dependency Injection)
3. Aspect Oriented Programming
 - a. AOP 소개
 - b. 프록시 패턴
 - c. 스프링 @AOP 예제
4. Portable Service Abstraction
 - a. PSA 소개
 - i. 웹 MVC
 - ii. 스프링 트랜잭션
 - iii. 캐시
5. 강의 마무리

참고

- [스프링 학습 방법](#)
- [인프런 스프링 강좌 수강 방법](#)
- [토비의 스프링](#)
- [스프링 프레임워크 레퍼런스](#)
- [PetClinic](#)
- [Youtube/백기선](#)

프로젝트 준비

Spring-PetClinic

JDK 버전: 11

소스 코드: <https://github.com/spring-projects/spring-petclinic>

IDE: 인텔리J (커뮤니티 버전도 괜찮습니다.)

실행 방법:

- `./mvnw package`
- `java -jar target/*.jar`
- IDE에서 메인 애플리케이션 실행

프로젝트 살펴보기

프로젝트 구조 설명

- 일반적인 메이븐 프로젝트
- src/main/java
- src/main/resources
- src/test/java
- src/test/resources

스프링 부트 기반 프로젝트

- 스프링 부트
- 스프링 데이터 JPA
- DB: HSQLDB
- 뷰: 타임리프
- 캐시: EHCache

코드가 어떻게 흘러가는 걸까?

- 로그로 분석하는 방법
- 디버거로 분석하는 방법

코드를 조금 고쳐볼까?

- LastName이 아니라 FirstName으로 검색해 볼까?
- 정확히 일치하는게 아니라 해당 키워드가 들어있는 걸 찾아볼까?
- Owner에 age 추가

프로젝트 살펴보기 과제 풀이

LastName이 아니라 FirstName으로 검색해 볼까?

- 뷰 변경
- 코드 조금 변경

정확히 일치하는게 아니라 해당 키워드가 들어있는 걸 찾아볼까?

- 쿼리만 변경

Owner에 age 추가

- 모델 변경
- 스키마 변경
- 데이터 변경
- 뷰 변경

Inversion of Control

제어권이 뒤바뀌었다고?

일반적인 (의존성에 대한) 제어권: “내가 사용할 의존성은 내가 만든다.”

```
class OwnerController {  
    private OwnerRepository repository = new OwnerRepository();  
}
```

IoC: “내가 사용할 의존성을 누군가 알아서 주겠지”

- 내가 사용할 의존성의 타입(또는 인터페이스)만 맞으면 어떤거든 상관없다.
- 그래야 내 코드 테스트 하기도 편하지.

```
class OwnerController {  
    private OwnerRepository repo;  
  
    public OwnerController(OwnerRepository repo) {  
        this.repo = repo;  
    }  
}
```

```
    // repo를 사용합니다.  
}
```

```
class OwnerControllerTest {  
    @Test  
    public void create() {  
        OwnerRepository repo = new OwnerRepository();  
        OwnerController controller = new OwnerController(repo);  
    }  
}
```

참고

- <https://martinfowler.com/articles/injection.html>

IoC (Inversion of Control) 컨테이너

ApplicationContext (BeanFactory)

빈(bean)을 만들고 역어주며 제공해준다.

빈 설정

- 이름 또는 ID
- 타입
- 스코프

아이러니하게도 컨테이너를 직접 쓸 일은 많지 않다.

참고

- <https://github.com/spring-guides/understanding/tree/master/application-context>
- <https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/context/ApplicationContext.html>
- <https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/beans/factory/BeanFactory.html>

빈 (Bean)

스프링 IoC 컨테이너가 관리하는 객체

어떻게 등록하지?

- Component Scanning
 - @Component
 - @Repository
 - @Service
 - @Controller
 - @Configuration
- 또는 직접 일일이 XML이나 자바 설정 파일에 등록

어떻게 꺼내쓰지?

- @Autowired 또는 @Inject
- 또는 ApplicationContext에서 getBean()으로 직접 꺼내거나

특징

- 오로지 “빈”들만 의존성 주입을 해줍니다.

의존성 주입 (Dependency Injection)

필요한 의존성을 어떻게 받아올 것인가..

@Autowired / @Inject를 어디에 붙일까?

- 생성자
- 필드
- Setter

과제

- OwnerController에 petRepository 주입하기

AOP 소개

흩어진 코드를 한 곳으로 모아

흩어진 AAAA 와 BBBB

```
class A {  
    method a () {  
        AAAA -> AAA  
        오늘은 7월 4일 미국 독립 기념일이래요.  
        BBBB -> BB  
    }  
  
    method b () {  
        AAAA -> AAA  
        저는 아침에 운동을 다녀와서 밥먹고 빨래를 했습니다.  
        BBBB -> BB  
    }  
}  
  
class B {  
    method c() {  
        AAAA -> AAA  
        점심은 이거 짭니다 못먹었는데 저녁엔 제육볶음을 먹고 싶네요.  
        BBBB -> BB  
    }  
}
```

모아 놓은 AAAA 와 BBBB

```
class A {  
    method a () {  
        오늘은 7월 4일 미국 독립 기념일이래요.  
    }  
  
    method b () {  
        저는 아침에 운동을 다녀와서 밥먹고 빨래를 했습니다.  
    }  
}  
  
class B {  
    method c() {  
        점심은 이거 짭니다 못먹었는데 저녁엔 제육볶음을 먹고 싶네요.  
    }  
}  
  
class AAAABBBB {  
    method aaaabbbb(JoinPoint point) {  
        AAAA  
        point.execute()  
        BBBB  
    }  
}
```

```
}  
}
```

다양한 AOP 구현 방법

- 컴파일 A.java ----(AOP)---> A.class ([AspectJ](#))
- 바이트코드 조작 A.java -> A.class --- (AOP)---> 메모리 (AspectJ)
- 프록시 패턴 (스프링 AOP)

프록시 패턴

- <https://refactoring.guru/design-patterns/proxy>

프록시 패턴

기존 코드 건드리지 않고 새 기능 추가하기

AOP 적용 예제

@LogExecutionTime 으로 메소드 처리 시간 로깅하기

@LogExecutionTime 애노테이션 (어디에 적용할지 표시 해두는 용도)

```
@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
public @interface LogExecutionTime {
}
```

실제 Aspect (@LogExecutionTime 애노테이션 달린곳에 적용)

```
@Component
@Aspect
public class LogAspect {

    Logger logger = LoggerFactory.getLogger(LogAspect.class);

    @Around("@annotation(LogExecutionTime)")
    public Object logExecutionTime(ProceedingJoinPoint joinPoint) throws Throwable {
        Stopwatch stopWatch = new Stopwatch();
        stopWatch.start();

        Object proceed = joinPoint.proceed();

        stopWatch.stop();
        logger.info(stopWatch.prettyPrint());

        return proceed;
    }
}
```

PSA 소개

잘 만든 인터페이스

나의 코드

확장성이 좋지 못한 코드 or 기술에 특화되어 있는 코드

나의 코드

잘 만든 인터페이스 (PSA)

확장성이 좋지 못한 코드 or 기술에 특화되어 있는 코드

Service Abstraction

https://en.wikipedia.org/wiki/Service_abstraction

스프링 웹 MVC

@Controller 와 @RequestMapping

나의 코드

@Controller | @RequestMapping | ...

Servlet | Reactive

톰캣, 제티, 네티, 언더토우

스프링 트랜잭션

PlatformTransactionManager

나의 코드

@Transactional

[PlatformTransactionManager](#)

JpaTransacionManager | DatasourceTransactionManager | HibernateTransactionManager

스프링 캐시

CacheManager

나의 코드

@Cacheable | @CacheEvict | ...

[CacheManager](#)

JCacheManager | ConcurrentMapCacheManager | EhCacheCacheManager | ...

스프링 프레임워크 입문 강좌를 마쳤습니다.

감사합니다.

-백기선-