

# 스프링 프레임워크 입문

## 스프링이 뭔데?

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

### 수업 목표

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

### 수업 계획

1. 강의 소개
  - a. 강의 소개
  - b. 프로젝트 세팅
  - c. 프로젝트 살펴보기
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. AOP 적용 예제
4. Portable Service Abstraction
  - a. PSA 소개
  - b. 스프링 트랜잭션
  - c. 캐시
  - d. 웹 MVC
5. 강의 마무리

### 참고

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

# 프로젝트 세팅

## Spring-PetClinic

JDK 버전: 1.8 (9랑 10은 노노)

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

IDE: 인텔리J (커뮤니티 버전도 괜찮습니다.)

주의 할 점: **wro4j 메이븐 플러그인이 현재 Java 9 이상을 지원하지 않습니다.**

실행 방법:

- mvn spring-boot:run
- IDE에서 메인 애플리케이션 실행

프로젝트 구조 설명

- 일반적인 메이븐 프로젝트
- 스프링 부트 기반 프로젝트

# Inversion of Control

제어가 뒤바뀌었다고? 뭔 소리야?

“내가 쓸 놈은 내가 만들어 쓸게...” (일반적인 의존성에 대한 제어권)

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

“내가 쓸 놈은 이 놈인데... 누군가 알아서 주겠지...” (IoC)

- 내가 쓸 놈의 타입만 맞으면 어떤거든 상관없지 뭐..
- 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);  
    }  
}
```

# 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.

# 빈 (Bean)

스프링 IoC 컨테이너가 관리하는 객체

## 어떻게 등록하지?

- Component Scanning
  - @Component
    - @Repository
    - @Service
    - @Controller
- 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

# AOP 소개

흩어진 코드를 한 곳으로 모아

## 흩어진 AAAA 와 BBBB

```
class A {
    method a () {
        AAAA
        오늘은 7월 4일 미국 독립 기념일이래요.
        BBBB
    }

    method b () {
        AAAA
        저는 아침에 운동을 다녀와서 밥먹고 빨래를 했습니다.
        BBBB
    }
}

class B {
    method c() {
        AAAA
        점심은 이거 짭니다 못먹었는데 저녁엔 제육볶음을 먹고 싶네요.
        BBBB
    }
}
```

## 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
    }
}
```

} }



# 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

# 스프링 트랜잭션

PlatformTransactionManager

나의 코드

@Transactional

[PlatformTransactionManager](#)

JpaTransacionManager | DatasourceTransactionManager | HibernateTransactionManager

# 스프링 캐시

CacheManager

나의 코드

@Cacheable | @CacheEvict | ...

[CacheManager](#)

JCacheManager | ConcurrentMapCacheManager | EhCacheCacheManager | ...

# 스프링 웹 MVC

@Controller 와 @RequestMapping

나의 코드

@Controller | @RequestMapping | ...

Servlet | Reactive

톰캣, 제티, 네티, 언더토우

스프링 프레임워크 입문 강좌를 마쳤습니다.

감사합니다.

-백기선-