SpringBoot 3일차 - AOP

AOP(Aspect Oriented Programming)
언제 왜 쓰나? 만 알면 좀 쉬움..

- 모든 메소드의 호출 시간을 측정하고 싶다.
  - 메소드 1000개 되면 다 해봐야됨 ㅠㅠㅠ

핵심 관심 사항이 아닌걸 해야할때, 공통 관심 사항이 있을 때, 핵심 비즈니스 로직과 별로안중요한게 섞일때, 어떤 로직을 별도의 공통 로직으로 만들기 어려울때, 모든 로직 다 찾아서 변경할때

공통 관심 사항(cross-cutting concern) vs 핵심 관심 사항(core concern) 분리 => AOP!!

The bean 'timeTraceAop', defined in class path resource [hello/hellospring/SpringConfig.class], could not be registered. A bean with that name has already been defined in file [/Users/sehwan/Desktop/inflearn_spring/hello-spring/out/production/classes/hello/hellospring/aop/TimeTraceAop.class] and overriding is disabled.
=> 만약 springconfig에 @Bean으로 TimeTracAop를 등록해놓았다면, @Component나 @Bean 둘중에 하나는 지워야한다
왜냐) @Component라는 어노테이션이 붙어있으면 스프링 빈에 자동으로 등록되는데 springconfig에서 또 등록한 것임. 근데 스프링 bean은 한 클래스당 하나의 bean만 할당해서 등록해주므로 안됨!
근데 여기서 springconfig에 있는걸 안지우고 aop에 있는 component를 지우면 에러가 나는데 아래와같다

```
Description:

The dependencies of some of the beans in the application context form a cycle:

   memberController defined in file [/Users/sehwan/Desktop/inflearn_spring/hello-spring/out/production/classes/hello/hellospring/controller/MemberController.class]
      ↓
   memberService defined in class path resource [hello/hellospring/SpringConfig.class]
┌─────┐
|  timeTraceAop defined in class path resource [hello/hellospring/SpringConfig.class]
└─────┘


Action:

Relying upon circular references is discouraged and they are prohibited by default. Update your application to remove the dependency cycle between beans. As a last r

Process finished with exit code 0
```

싸이클이 계속 돈다고 하는거같은데 뭔진 잘 모르겠음

에러 고치고 나서 회원 목록 누르니까

```
2023-05-26T02:39:53.725+09:00  INFO 19806 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 7 ms
START: execution(String hello.hellospring.controller.HomeController.home())
END: execution(String hello.hellospring.controller.HomeController.home()) 0ms
START: execution(String hello.hellospring.controller.MemberController.list(Model))
START: execution(List hello.hellospring.service.MemberService.findMembers())
START: execution(List org.springframework.data.repository.ListCrudRepository.findAll())
Hibernate: select m1_0.id,m1_0.name from member m1_0
END: execution(List org.springframework.data.repository.ListCrudRepository.findAll()) 98ms
END: execution(List hello.hellospring.service.MemberService.findMembers()) 101ms
END: execution(String hello.hellospring.controller.MemberController.list(Model)) 101ms
```

이렇게 뜨는데 db 볼때 98ms 서비스에서 101ms 컨트롤러에서 101ms

execution....이렇게 뜨는게 Around안에 들어가는 내용임.
Around : aop를 지정하고 어디에 적용할지를 정하는거임. 그니까 우리 코드에서는 hello.hellospring 패키지의 하위 폴더를 얘기하는듯.

그렇다면 실제로 어떻게 동작하나?

스프링은 어떻게 동작함? -> 강의록 그림 참조
적용 전에는 의존관계로 호출함.
aop 적용 후에는 가짜 멤버서비스인 프록시를 만들어냄. (**프록시기술**)
그러면서 스프링 컨테이너는 스프링 빈을 등록할때 가짜 스프링 빈을 등록시키고 joinPoint.proceed()로 넘어갈때 실제 멤버서비스로 가서 aop를 적용함.

그럼 프록시 기술은 어디서 확인할수있나? 눈으로 확인해보자
MemberController에서 memberService가 injection될때
sout memberservice.getClass()로 볼 수 있다.
=> memberService = class hello.hellospring.service.MemberService$$SpringCGLIB$$0 이런식으로 뒤에 이상한걸 붙여서 가짜를 만듦.

강의록을 참조하자! 그림으로 보면 좀 쉽던뎅

⭐jointPoint.proceed()와 pointcut의 차이?
조인 포인트는 조금 추상적인 개념이고, 어드바이스가 적용될 수 있는 위치를 말합니다. 쉽게 이야기해서 모든 메서드의 실행 부분이 조인 포인트가 될 수 있습니다.
포인트 컷은 수 많은 조인 포인트 중에서 실제 어드바이스를 적용할 조인 포인트를 선별하는 작업이라 생각하시면 됩니다. "execution ... *(..)"의 모든 부분이 여기에서 지정한 포인트 컷 룰입니다. 이 조건을 만족하는 모든 조인 포인트에 어드바이스가 적용됩니다.