싱글톤 컨테이너

웹 애플리케이션과 싱글톤

싱글톤 패턴

private static final

static으로 가짐 → class 레벨로 존재하기 때문에 딱 하나만 존재

same ==

equal equals 메소드

- appconfig를 다 싱글톤으로 바꿔서 해야 하나?

  → ㄴㄴ 스프링 컨테이너에서 기본적으로 객체를 다 싱글톤으로 만들어서 관리해 줌

-

싱글톤 컨테이너

37강 싱글톤 방식의 주의점

싱글톤 객체는 상태를 유지하게 설계하면 안 됨

- 특정 클라이언트에 의존적인 필드 X
- 특정 클라이언트가 값을 변경할 수 있는 필드가 있으면 안 됨
- 가급적 읽기만 가능하도록
- 스프링 빈의 필드에 공유 값을 설정하면 정말 큰 장애가 발생할 수 있음!

이거 제대로 안 잡으면

다른 사람 아이디 로그인 내역이 보이고 함...

스프링 빈은 항상 무상태로 설계하자!

38강 @Configuration과 싱글톤

@Configuration은 싱글톤을 위해 존재

// @Bean memberSerivce → new MemoryMemberRepository()

// @Bean orderService → new MemoryMemberRepository()

39강 @Configuration과 바이트코드 조작의 마법

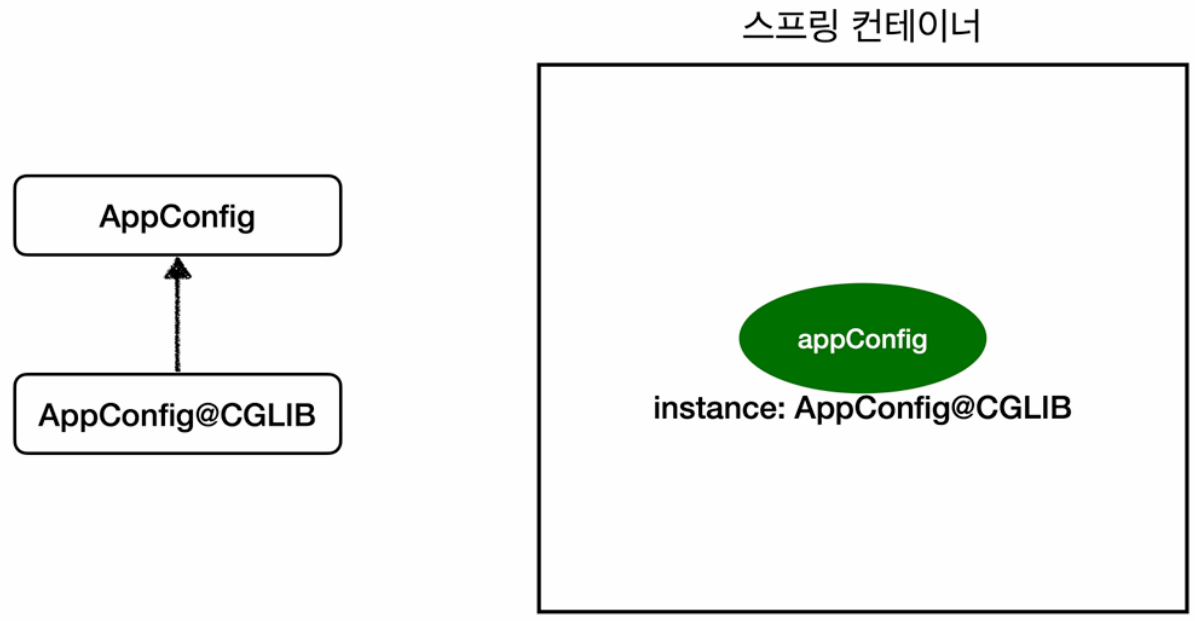
순수한 클래스라면 다음과 같이 출력되어야 한다.

```
class hello.core.AppConfig
```

그런데 예상과는 다르게 클래스 명에 xxxCGLIB가 붙으면서 상당히 복잡해진 것을 볼 수 있다.

이것은 내가 만든 클래스가 아니라 스프링이 CGLIB라는 바이트코드 조작 라이브러리를 사용해서 AppConfig 클래스를 상속받은 임의의 다른 클래스를 만들고, 그 다른 클래스를 스프링 빈으로 등록한 것이다!

그림



- @Bean이 붙은 메서드마다 이미 스프링 빈이 존재하면 존재하는 빈을 반환하고, 스프링 빈이 없으면 생성해서 스프링 빈으로 등록하고 반환하는 코드가 동적으로 만들어진다.
- 덕분에 싱글톤이 보장되는 것이다.

정리

@Bean 만 사용해도 스프링 빈으로 등록되지만, 싱글톤을 보장하지 않는다.

memberRepository() 처럼 의존관계 주입이 필요해서 메서드를 직접 호출할 때 싱글톤을 보장하지 않
크게 고민할 것이 없다. 스프링 설정 정보는 항상
`@Configuration`
을 사용하자.