Profile

다른 환경(test환경, production 환경)에서 **각각 다른 설정** (데이터베이스 설정, 암호화, 시스템 프로퍼티 설정 등)을 해야할 때 환경에 따라 서로 다른 `Bean`을 등록할 수 있다. 이를 `프로파일`이라 한다.

`@Profile`은 **클래스**와 **메서드**에 정의할 수 있다.

Profile.java

```java
@Target({ElementType.TYPE, ElementType.METHOD})
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Conditional({ProfileCondition.class})
public @interface Profile {
    String[] value();
}
```

MyConfig.java

```java
package com.megait.profile;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Profile;

@Configuration
public class MyConfig {
    @Bean
    @Profile("catlover")
    public String cat(){
        return "cat";
    }

    @Bean
    @Profile("doglover")
    public String dog(){
        return "dog";
    }
}
```

application.properties

```
spring.profiles.active=doglover
```

`doglover` 프로파일을 활성화한다.

ProfileApplicationTests .class (Test)

```
package com.megait.profile;

import org.assertj.core.api.Assertions;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
class ProfileApplicationTests {

    @Autowired
    private MyConfig myConfig;

    @Test
    void contextLoads() {
        String pet1 = myConfig.cat();
        Assertions.assertThat(pet1).isNotNull();


        String pet2 = myConfig.dog();
        Assertions.assertThat(pet2).isNotNull();
    }

}
```

`@Profile("doglover")` 로 지정되었던 `dog()` 은 assert 를 통과했지만
`@Profile("catlover")` 로 지정되었던 `cat()` 은 assert 를 통과하지 못했다.

```
org.springframework.beans.factory.NoSuchBeanDefinitionException: No bean named 'cat' available
```

반대로 `application.properties` 의 `spring.profiles.active=catlover` 로 수정하고 다시 테스트를 실행해보자.

이번엔 반대로 dog() 의 값을 받아오지 못했다.

```
org.springframework.beans.factory.NoSuchBeanDefinitionException: No bean named 'dog' available
```

관행적으로 `@Profile("test"), @Profile("debug")` (Test 레벨) `prod` (실제 서비스 레벨)로 구분하여 빈을 생성한다.

각 프로파일에 따른 `.properties` 설정

`application-{profile}.properties` 형태로 프로퍼티 파일을 만들면 된다.

application-catlover.properties

```
favorite.snack=Churu
favorite.toy=Cat Fishing Rod
```

application-doglover.properties

```
favorite.snack=Jerky
favorite.toy=Disk
```

ProfileApplicationTests .class (Test)

```
package com.megait.profile;

import org.assertj.core.api.Assertions;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
class ProfileApplicationTests {

    @Autowired
    private MyConfig myConfig;

    // @Test (이 메서드는 테스트에 포함시키지 말자. 필연적으로 Exception 이 발생하기 때
문)
    void contextLoads() {
        String pet1 = myConfig.cat();
        Assertions.assertThat(pet1).isNotNull();


        String pet2 = myConfig.dog();
        Assertions.assertThat(pet2).isNotNull();
    }

    @Test
    void toyAndSnack(
            @Value("${favorite.toy}") String toy,
            @Value("${favorite.snack}") String snack){
        System.out.println("toy : " + toy);
        System.out.println("snack : " + snack);
    }
```

```
}
```

> **Quiz1)** `maven` **대신** `gradle`**로 바꾸어 실행해보기**

> **Quiz2)** `.properies`**를** `.yml`**로 바꾸어 실행해보기**

> **Quiz3) Test 에서 여러가지 설정값 불러오기 방법들을 활용하여 다양하게 프로퍼티 값 불러오기 (최소 2가지)**

> **Quiz4) Runner에서 여러가지 설정값 불러오기 방법들을 활용하여 다양하게 프로퍼티 값 불러오기 (최소 2가지)**

> **Quiz5) application.properties (main)의** `spring.profiles.active`**는** `catlover`**로 설정되어있다고 가정했을 때,**
>
> **@Test 에서 cat() 대신 dog() 빈을 얻어올 수 있는 방법은 무엇이 있을까? 단, MyConfig.java는 수정하지 않는다. (최소 2가지)**