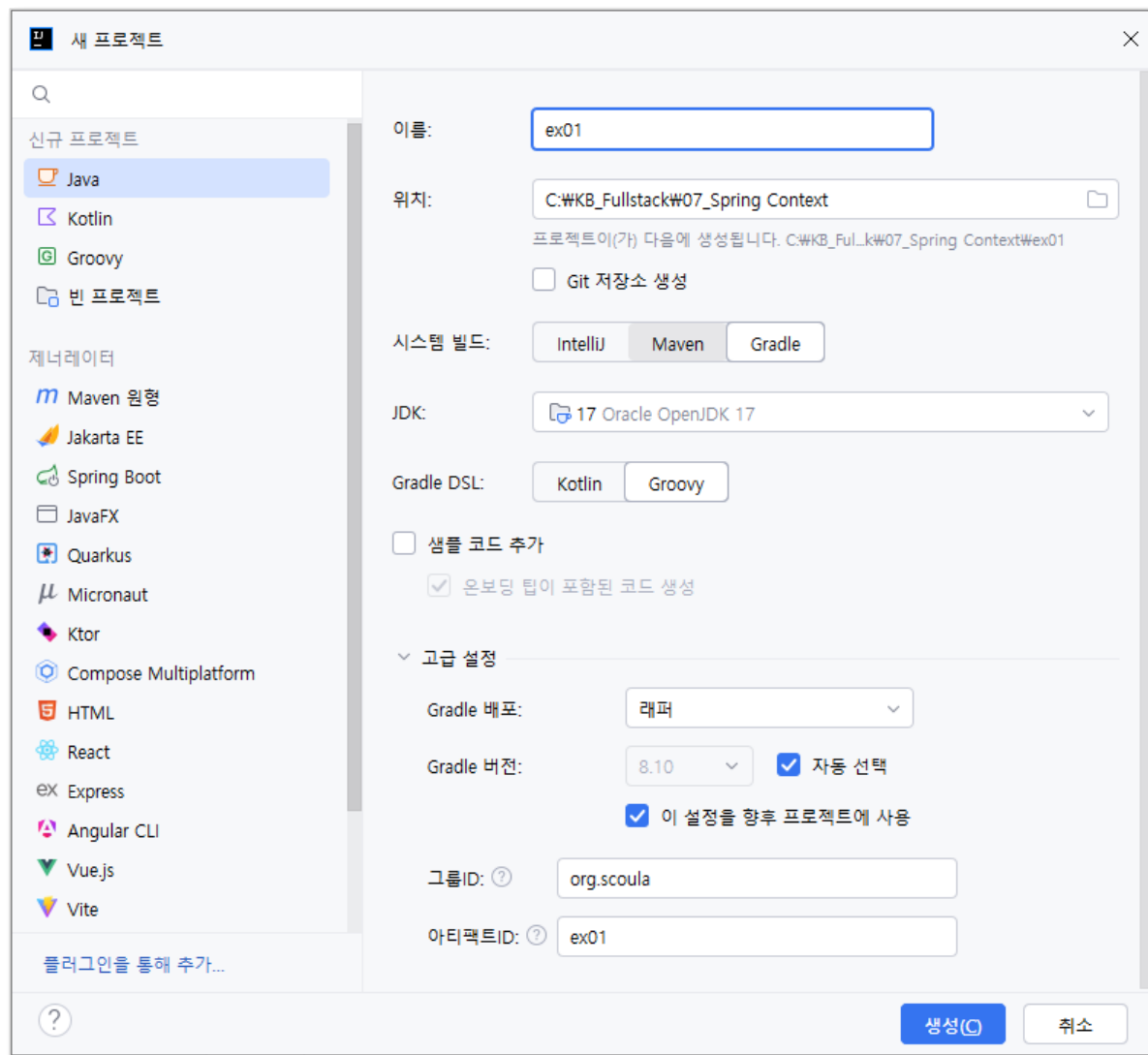


2025년 상반기 K-디지털 트레이닝

스프링 컨텍스트- 빈 정의

[KB] IT's Your Life

✓ 프로젝트 생성



새 프로젝트

신규 프로젝트

- Java
- Kotlin
- Groovy
- 빈 프로젝트

제너레이터

- Maven 원형
- Jakarta EE
- Spring Boot
- JavaFX
- Quarkus
- Micronaut
- Ktor
- Compose Multiplatform
- HTML
- React
- Express
- Angular CLI
- Vue.js
- Vite

플러그인을 통해 추가...

이름: ex01

위치: C:\KB_Fullstack\07_Spring Context
프로젝트 이름(가) 다음에 생성됩니다. C:\KB_Fullstack\07_Spring Context\ex01

☐ Git 저장소 생성

시스템 빌드: IntelliJ Maven Gradle

JDK: 17 Oracle OpenJDK 17

Gradle DSL: Kotlin Groovy

☐ 샘플 코드 추가
☒ 온보딩 팁이 포함된 코드 생성

고급 설정

Gradle 배포: 래퍼

Gradle 버전: 8.10 ☒ 자동 선택
☒ 이 설정을 향후 프로젝트에 사용

그룹ID: org.scoula

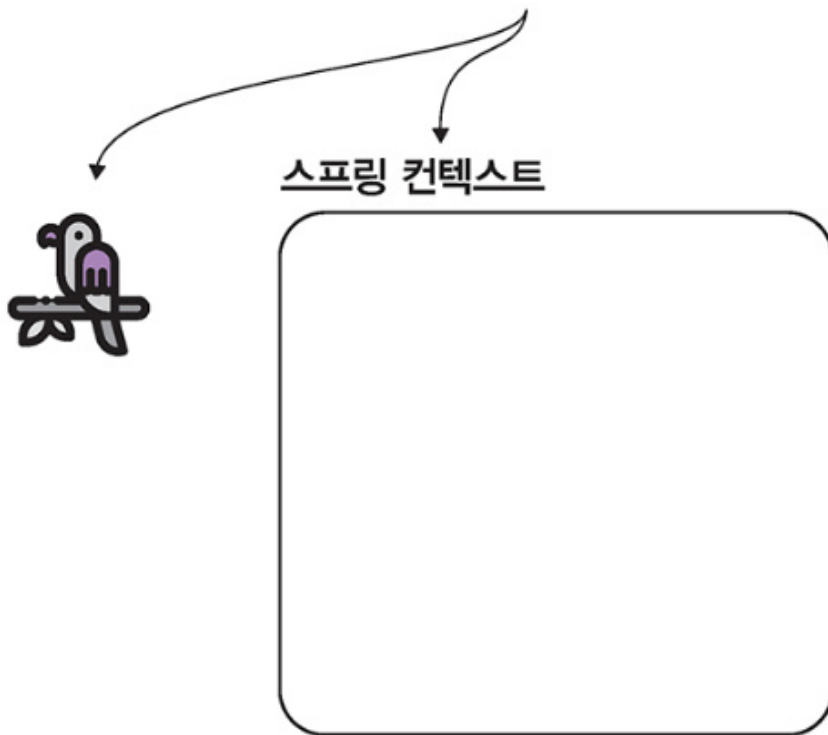
아티팩트ID: ex01

생성(C) 취소

build.gradle

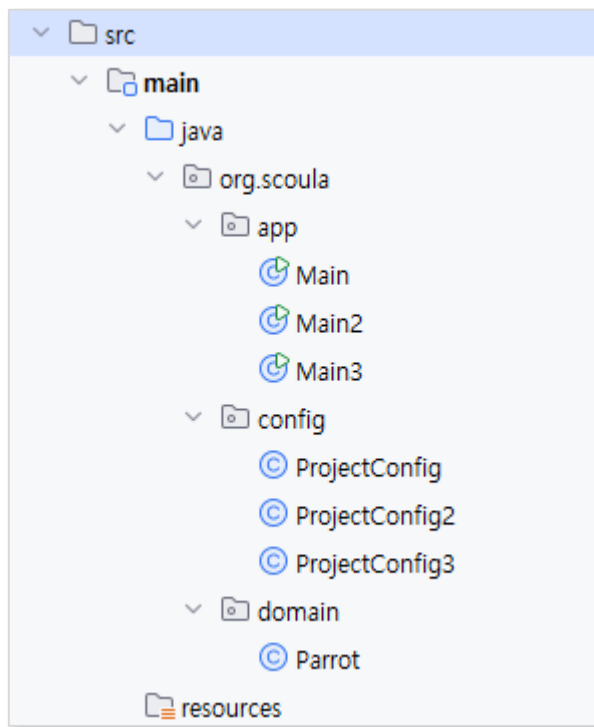
```
implementation 'org.springframework:spring-context:5.3.37'
```

먼저 Parrot 타입의 객체와 스프링 컨텍스트를
독립적으로 만드는 것으로 시작한다.



처음에는 스프링 컨텍스트가 비어 있다. 나중에
Parrot 인스턴스를 컨텍스트로 이동하여 스프링에
인스턴스를 알리고 이를 관리할 수 있도록 한다.

✓ 실습 환경



domain.Parrot.java

```
package org.scoula.domain;

public class Parrot {
    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

Main.java

```
package org.scoula.app;

import org.scoula.domain.Parrot;

public class Main {

    public static void main(String[] args) {

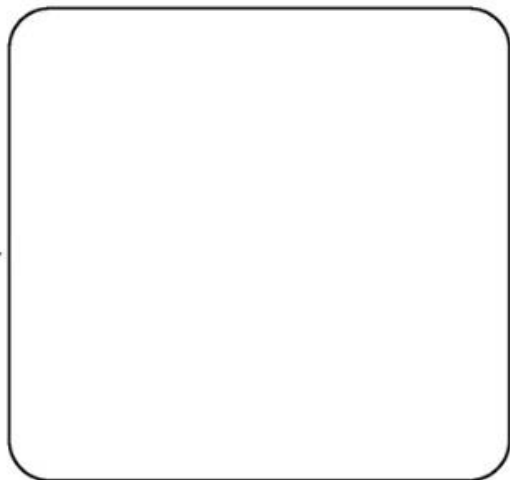
        Parrot p = new Parrot();
    }
}
```

당신이 한 것

Parrot 인스턴스를 생성했지만
스프링 컨텍스트 외부에 있다.



스프링 컨텍스트

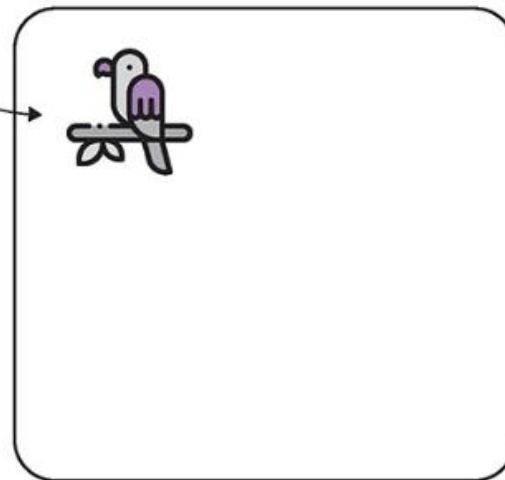


스프링 컨텍스트를 정의했지만 현재 비어 있다.

당신이 하려는 것

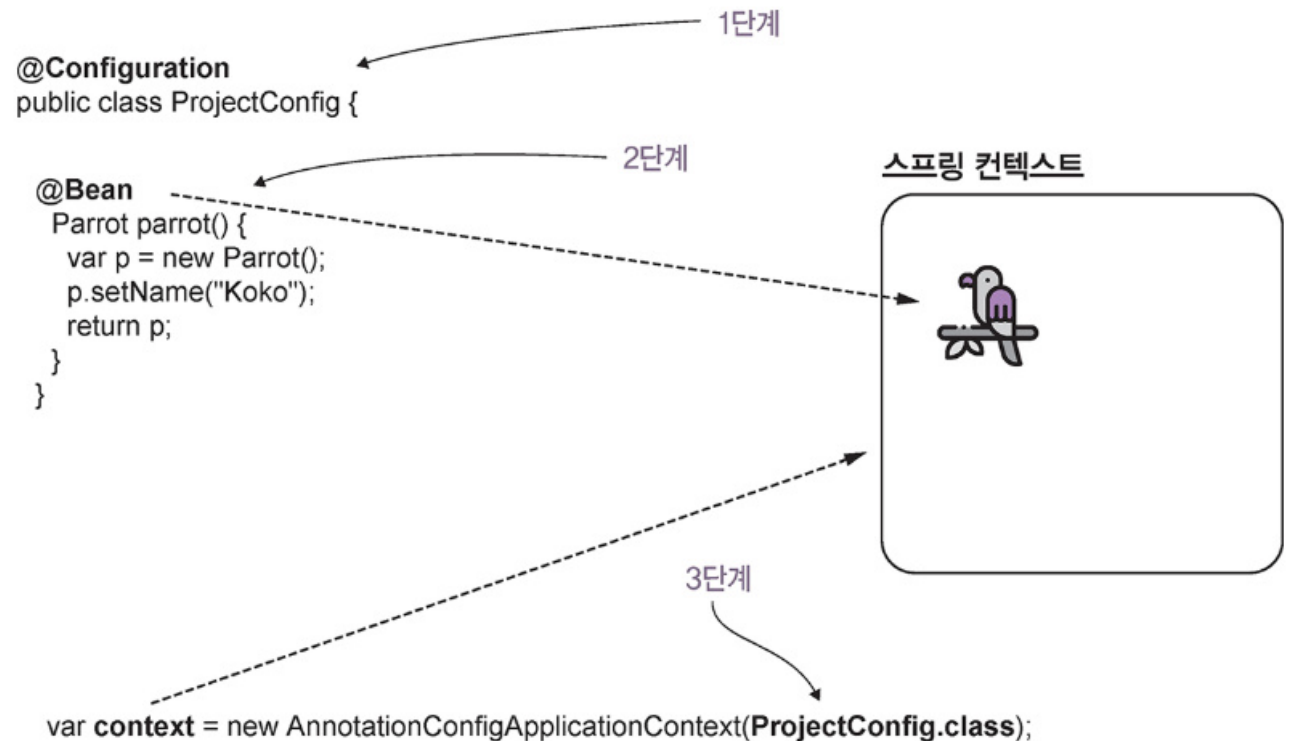
스프링 컨텍스트에 Parrot 인스턴스를 추가하면
스프링이 해당 인스턴스를 볼 수 있다.

스프링 컨텍스트



✓ @Bean 애너테이션을 사용하여 스프링 컨텍스트에 빈 추가

- 스프링은 Bean으로 등록된 객체만 관리할 수 있음
- 스프링 컨텍스트에 빈을 추가하는 단계
 1. @Configuration으로 구성 클래스 정의
 - 스프링 컨텍스트 구성 시 사용
 2. 컨텍스트에 추가하려는 객체 인스턴스를 반환하는 메서드를 구성하는 클래스에 추가, 해당 메서드에 @Bean 애너테이션 추가
 3. 스프링이 1에서 정의한 구성 클래스 사용



✓ 1단계: 프로젝트에서 구성 클래스 정의하기

 config.ProjectConfig.java

```
package org.scoula.config;

import org.springframework.context.annotation.Configuration;

@Configuration
public class ProjectConfig {

}
```

✓ 2단계: 빈을 반환하는 메서드를 생성하고 Bean 애너테이션을 메서드에 추가하기

config.ProjectConfig.java

```
package org.scoula.config;

import org.scoula.domain.Parrot;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class ProjectConfig {
    @Bean
    Parrot parrot() {
        var p = new Parrot();
        p.setName("Koko");
        return p;
    }
}
```

✓ 3단계: 새로 생성된 구성 클래스로 스프링이 컨텍스트를 초기화하도록 만들기

Main.java

```
package org.scoula.app;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class Main {

    public static void main(String[] args) {
        var context = new AnnotationConfigApplicationContext(ProjectConfig.class);
    }
}
```

○ AnnotationConfigApplicationContext(구성클래스);

- 구성 클래스로 컨텍스트를 만들도록 하는 클래스
- 구성 클래스의 class를 매개변수로 지정

✓ 컨텍스트에서 원하는 빈 객체 추출하기

 app.Main.java

```
package org.scoula.app;

import org.scoula.config.ProjectConfig;
import org.scoula.domain.Parrot;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class Main {

    public static void main(String[] args) {
        var context = new AnnotationConfigApplicationContext(ProjectConfig.class);

        Parrot p = context.getBean(Parrot.class);
        System.out.println(p.getName());
    }
}
```

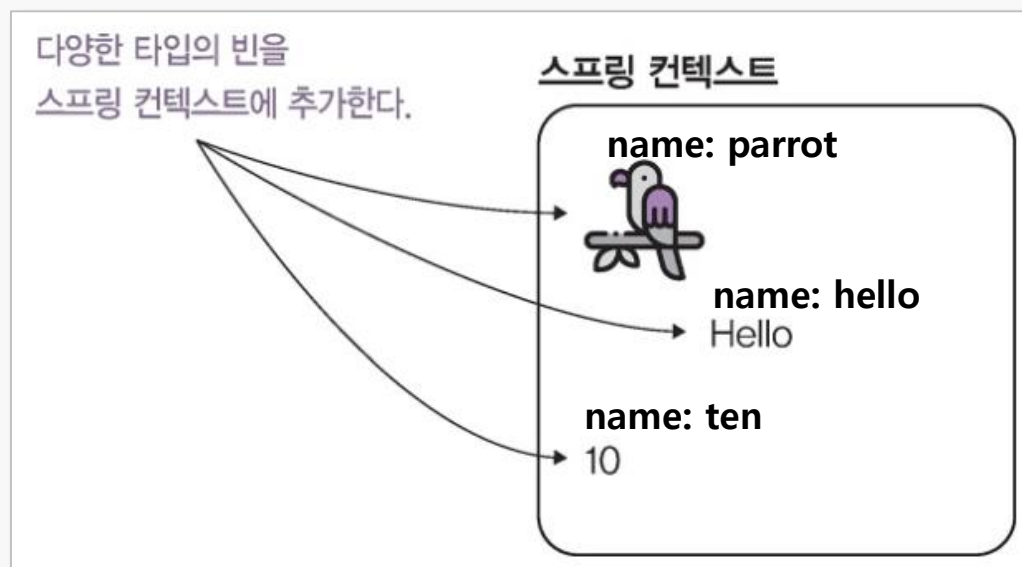
config.ProjectConfig.java

```
@Configuration
public class ProjectConfig {

    @Bean
    Parrot parrot() {
        var p = new Parrot();
        p.setName("Koko");
        return p;
    }

    @Bean
    String hello() {
        return "Hello";
    }

    @Bean
    Integer ten() {
        return 10;
    }
}
```



Main.java

```
public class Main {  
  
    public static void main(String[] args) {  
        var context = new AnnotationConfigApplicationContext(ProjectConfig.class);  
  
        Parrot p = context.getBean(Parrot.class);  
  
        System.out.println(p.getName());  
  
        String s = context.getBean(String.class);  
  
        System.out.println(s);  
  
        Integer n = context.getBean(Integer.class);  
  
        System.out.println(n);  
    }  
}
```

```
Koko  
Hello  
10
```

- 동일한 타입에 대해서는 1개의 Bean만 추출할 수 있음

✓ 스프링 컨텍스트에 동일한 타입의 빈 여러 개 추가하기

config.ProjectConfig2.java

```
@Configuration
public class ProjectConfig2 {

    @Bean
    Parrot parrot1() {
        var p = new Parrot();
        p.setName("Koko");
        return p;
    }

    @Bean
    Parrot parrot2() {
        var p = new Parrot();
        p.setName("Miki");
        return p;
    }

    @Bean
    Parrot parrot3() {
        var p = new Parrot();
        p.setName("Riki");
        return p;
    }
}
```


✓ 스프링 컨텍스트에 동일한 타입의 빈 추출하기

📄 Main2.java

```
public class Main2 {  
  
    public static void main(String[] args) {  
        var context = new AnnotationConfigApplicationContext(ProjectConfig2.class);  
  
        Parrot p = context.getBean(Parrot.class); // 예외 발생 !!!  
        System.out.println(p.getName());  
  
    }  
}
```

- Parrot 타입으로 인스턴스가 3개 등록되어 있음 → 3개 중 어느 것을 참조할지 결정할 수 없어 예외 발생

○ 타입 대신 빈의 이름으로 선택해야 함

- @Bean 등록 시 사용한 메서드명이 빈의 기본 이름으로 등록됨
- @Bean(name="") 또는 @Bean(value="")를 사용하여 이름 지정 가능

config.ProjectConfig2.java

```
@Configuration
public class ProjectConfig2 {

    @Bean
    Parrot parrot1() {
        var p = new Parrot();
        p.setName("Koko");
        return p;
    }

    @Bean(name = "miki") // 빈의 이름 등록 @Bean(value="miki"), @Bean("miki")
    Parrot parrot2() {
        var p = new Parrot();
        p.setName("Miki");
        return p;
    }

    @Bean
    Parrot parrot3() {
        var p = new Parrot();
        p.setName("Riki");
        return p;
    }
}
```

✓ 스프링 컨텍스트에 동일한 타입의 빈 추출하기

Main2.java

```
public class Main2 {  
  
    public static void main(String[] args) {  
        var context = new AnnotationConfigApplicationContext(ProjectConfig2.class);  
  
        Parrot p = context.getBean("miki", Parrot.class);  
        System.out.println(p.getName());  
  
    }  
}
```

Miki

- context.getBean(빈이름, 타입.class);
 - 타입.class : 리턴 타입으로 사용할 클래스의 class

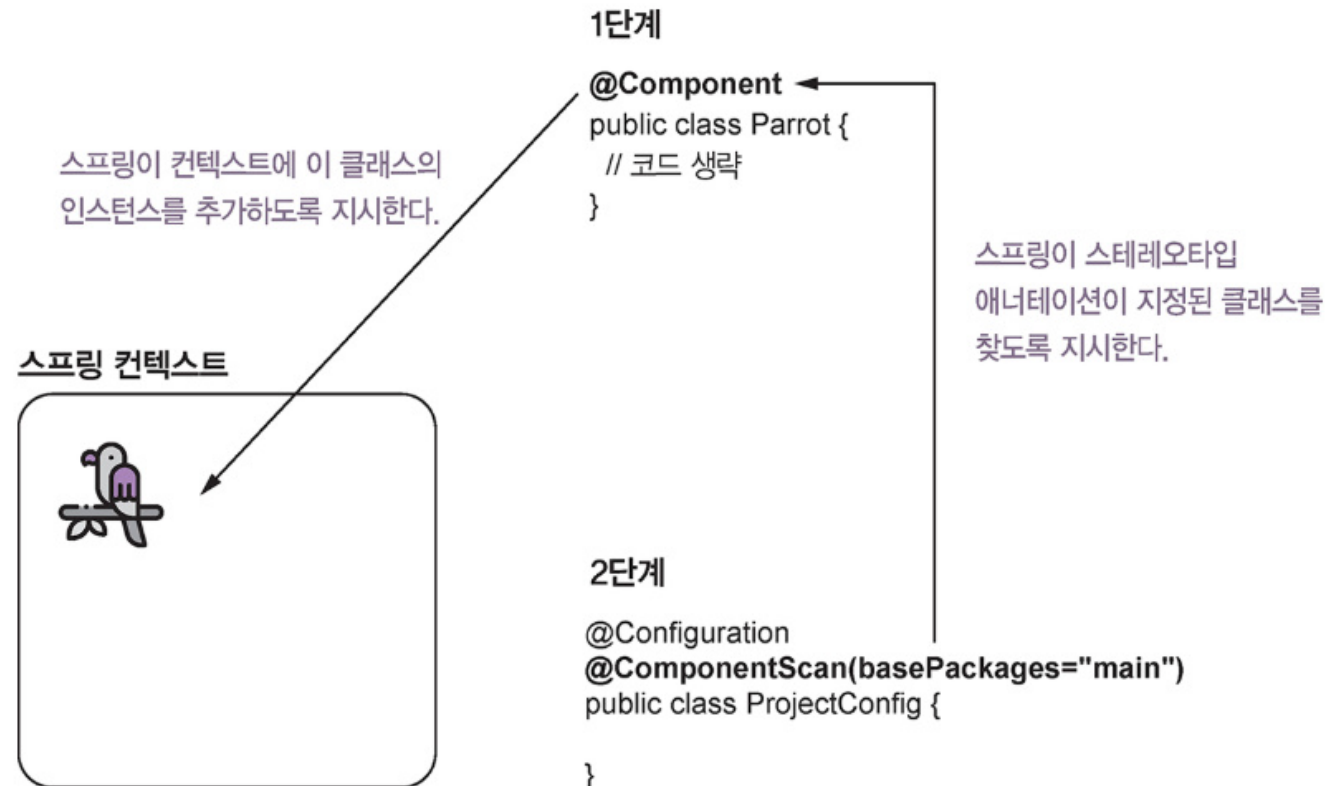
✓ 스테레오타입 애너테이션으로 스프링 컨텍스트에 빈 추가

1. @Component 애너테이션

- 스프링이 컨텍스트에 인스턴스를 추가할 클래스를 표시

2. 구성 클래스 위에 @ComponentScan

- 애너테이션으로 표시한 클래스를 어디에서 찾을 수 있는지 스프링에 지시



Parrot.java

```
package org.scoula.domain;

import org.springframework.stereotype.Component;

@Component // 디폴트 컴포넌트의 name: 클래스명의 camelCase - parrot
public class Parrot {

    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

config.ProjectConfig3.java

```
package org.scoula.config;

import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;

@Configuration
@ComponentScan(basePackages = "org.scoula.domain")
public class ProjectConfig3 {

}
```

Main3.java

```
package org.scoula.app;

import org.scoula.config.ProjectConfig3;
import org.scoula.domain.Parrot;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class Main3 {

    public static void main(String[] args) {
        var context = new AnnotationConfigApplicationContext(ProjectConfig3.class);

        Parrot p = context.getBean(Parrot.class);

        System.out.println(p);
        System.out.println(p.getName());
    }
}
```

```
org.scoula.domain.Parrot@d6e7bab
null
```

- ✓ **PostConstruct를 사용하여 인스턴스 생성 후 관리하기**
 - @Bean은 인스턴스 생성 후 후처리 가능
 - @Component는 생성 후 후처리 불가
 - javax.annotation-api에서 정의한 @PostConstruct를 사용하여 후처리 메서드 지정
 - implementation 'javax.annotation:javax.annotation-api:1.3.2'

Parrot.java

```
package org.scoula.domain;

import org.springframework.stereotype.Component;
import javax.annotation.PostConstruct;

@Component
public class Parrot {

    private String name;

    @PostConstruct
    public void init() {
        this.name = "Kiki";
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

Parrot.java

```
public class Main {  
    public static void main(String[] args) {  
        var context = new AnnotationConfigApplicationContext(ProjectConfig3.class);  
  
        Parrot p = context.getBean(Parrot.class);  
  
        System.out.println(p);  
        System.out.println(p.getName());  
    }  
}
```

```
org.scoula.domain.Parrot@223191a6  
Kiki
```