KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

스프링 컨텍스트- 빈 작성

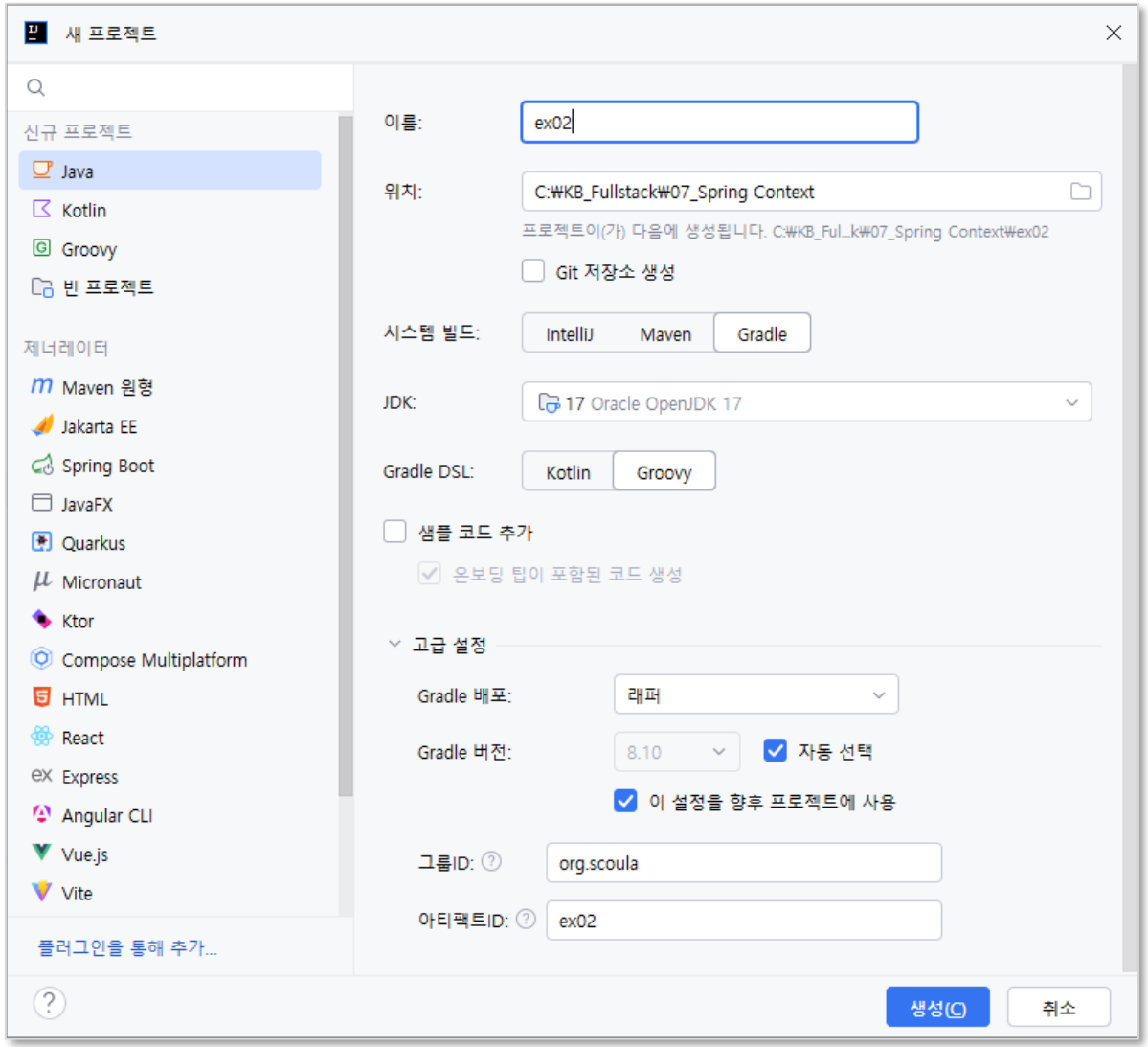
[KB] IT's Your Life

KB 국민은행

구성 파일에서 정의된 빈 간 관계 구현

프로젝트 생성

- 프로젝트명: ex02



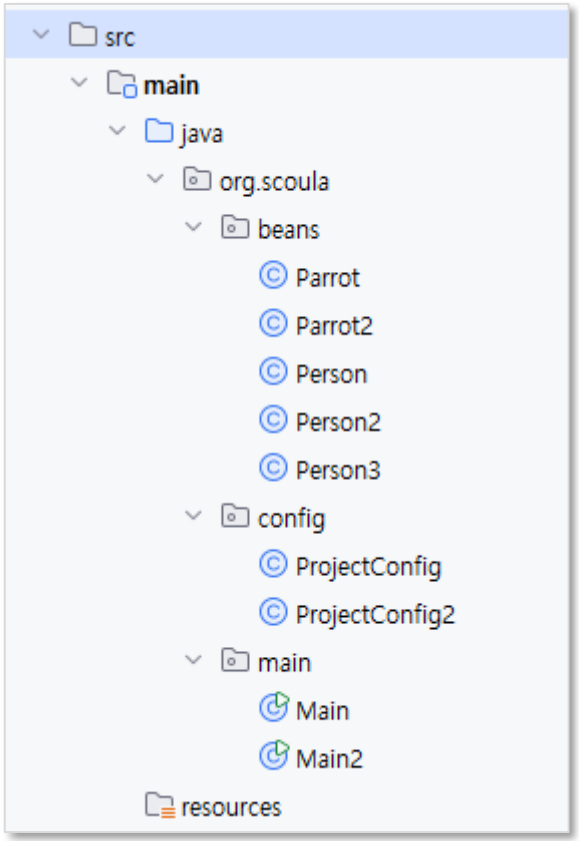
구성 파일에서 정의된 빈 간 관계 구현

build.gradle

```
plugins {
    id 'java'
}

group = 'org.scoula'
version = '1.0-SNAPSHOT'

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework:spring-context:5.3.37'
    implementation 'javax.annotation:javax.annotation-api:1.3.2'

    testImplementation platform('org.junit:junit-bom:5.10.0')
    testImplementation 'org.junit.jupiter:junit-jupiter'
}

test {
    useJUnitPlatform()
}
```

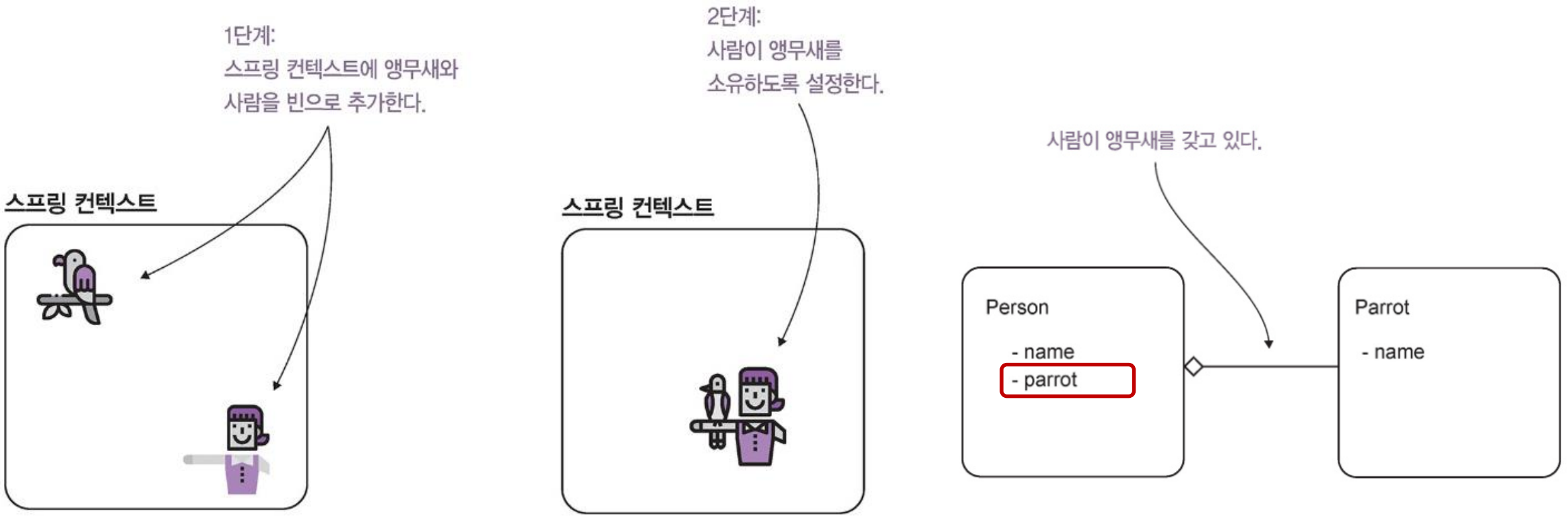
구성 파일에서 정의된 빈 간 관계 구현

실습 환경



구성 파일에서 정의된 빈 간 관계 구현

스프링 빈 간의 관계 설정



구성 파일에서 정의된 빈 간 관계 구현

beans.Parrot.java

```
package org.scoula.beans;

public class Parrot {
    private String name;

    public String getName() {
        return name;
    }

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

    @Override
    public String toString() {
        return "Parrot : " + name;
    }

}
```

구성 파일에서 정의된 빈 간 관계 구현

beans.Person.java

```
package org.scoula.beans;

public class Person {

  private String name;
  private Parrot parrot;

  public String getName() {
    return name;
  }

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

  public Parrot getParrot() {
    return parrot;
  }

  public void setParrot(Parrot parrot) {
    this.parrot = parrot;
  }
}
```

구성 파일에서 정의된 빈 간 관계 구현

config.ProjectConfig.java

```
package org.scoula.config;

import org.scoula.beans.Parrot;
import org.scoula.beans.Person;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class ProjectConfig {

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

    @Bean
    public Person person() {
        Person p = new Person();
        p.setName("Ella");
        return p;
    }
}
```

구성 파일에서 정의된 빈 간 관계 구현

main.Main.java

```
package org.scoula.main;

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

public class Main {

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

        Person person = context.getBean(Person.class);

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

        System.out.println("Person's name: " + person.getName());
        System.out.println("Parrot's name: " + parrot.getName());

        System.out.println("Person's parrot: " + person.getParrot());
    }
}
```

```
Person's name: Ella
Parrot's name: Koko
Person's parrot: null
```

구성 파일에서 정의된 빈 간 관계 구현

- 두 @Bean 메서드 간 직접 메서드를 호출하는 빈 작성

구성 파일에서 정의된 빈 간 관계 구현

config.ProjectConfig.java

```
@Configuration
public class ProjectConfig {

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

  @Bean
  public Person person() {
    Person p = new Person();
    p.setName("Ella");
    p.setParrot(parrot());
    return p;
  }
}
```

```
Person's name: Ella
Parrot's name: Koko
Person's parrot: Parrot : Koko
```

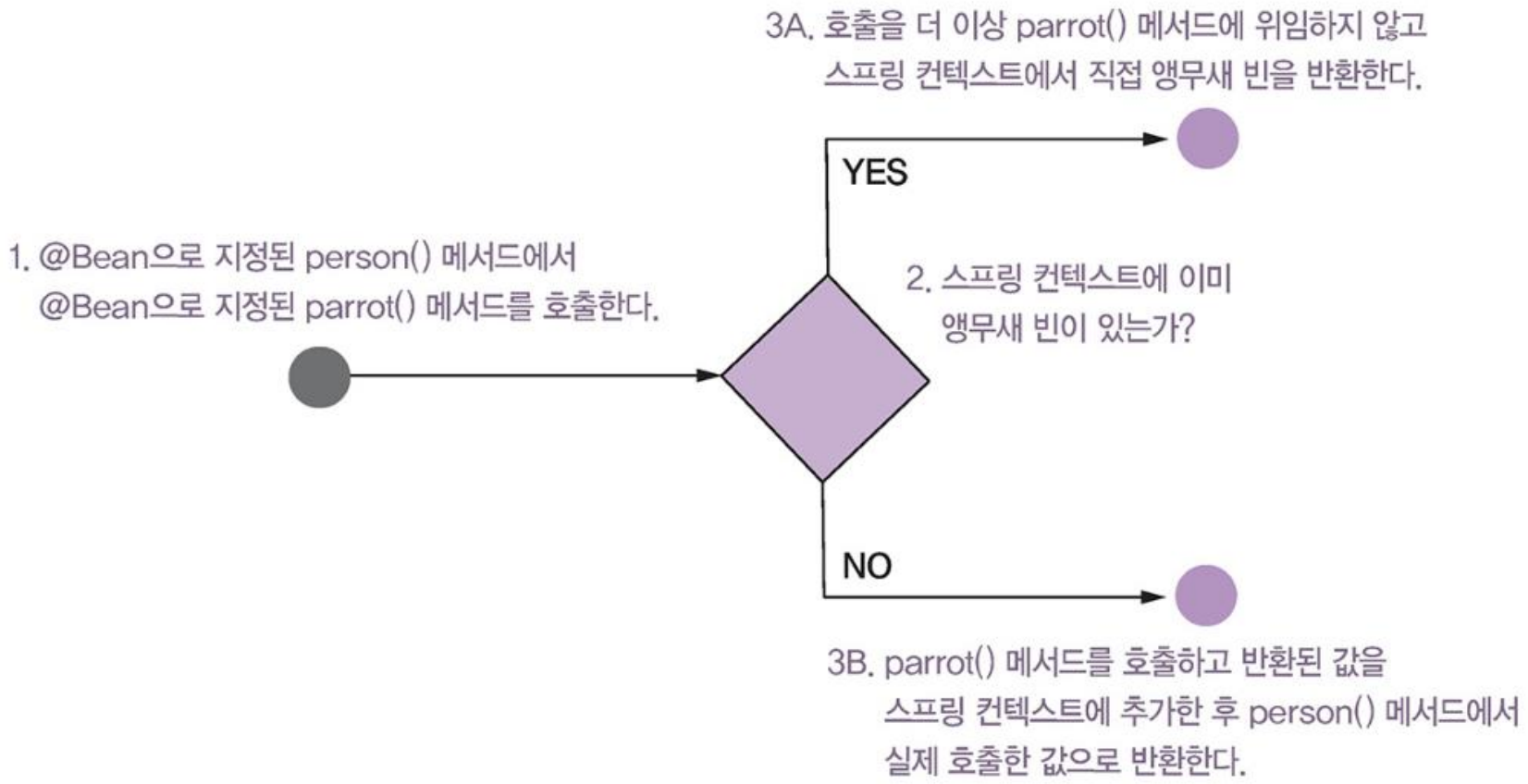
구성 파일에서 정의된 빈 간 관계 구현


```
@Configuration
public class ProjectConfig {

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

  @Bean
  public Person person() {
    Person p = new Person();
    p.setName("Ella");
    p.setParrot(parrot());
    return p;
  }
}
```

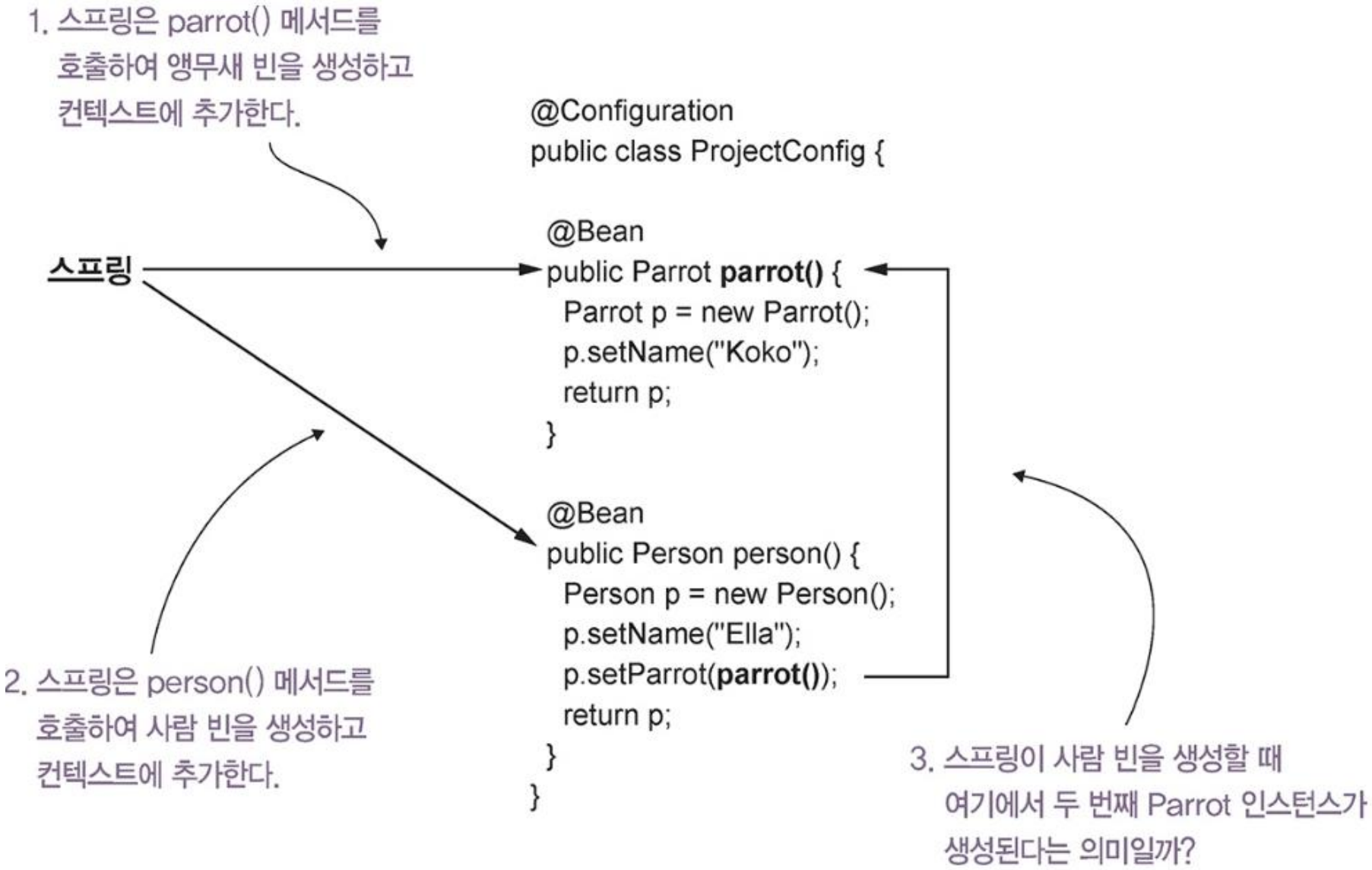
설정하려는 빈을 반환하는 메서드를 직접 호출하여 사람 빈과 앵무새 빈 간 관계를 정의한다.

그 결과 두 빈 사이에 has-A 관계가 생성되었다. 사람은 앵무새를 소유하고 있다.

구성 파일에서 정의된 빈 간 관계 구현



구성 파일에서 정의된 빈 간 관계 구현



구성 파일에서 정의된 빈 간 관계 구현

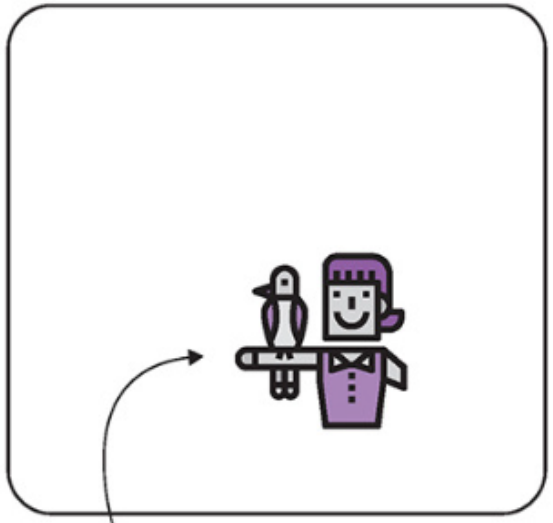
@Bean 메서드의 매개변수로 빈 와이어링하기

```
@Configuration
public class ProjectConfig {

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

  @Bean
  public Person person(Parrot parrot) {
    Person p = new Person();
    p.setName("Ella");
    p.setParrot(parrot);
    return p;
  }
}
```

메서드에 대한 매개변수를 정의하여
스프링 컨텍스트에서 빈을 제공하도록
스프링에 지시한다.

스프링이 전달한 참조로
사람의 속성 값을 설정한다.

스프링 컨텍스트



그 결과 두 빈 사이에 has-A 관계가 생성된다.
사람은 앵무새를 소유하고 있다.

구성 파일에서 정의된 빈 간 관계 구현

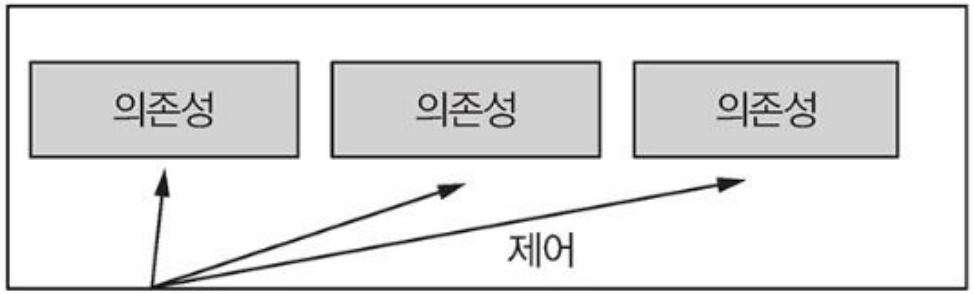
config.ProjectConfig.java

```
@Configuration
public class ProjectConfig {

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

  @Bean
  public Person person(Parrot parrot) {
    Person p = new Person();
    p.setName("Ella");
    p.setParrot(parrot);
    return p;
  }
}
```

```
Person's name: Ella
Parrot's name: Koko
Person's parrot: Parrot : Koko
```

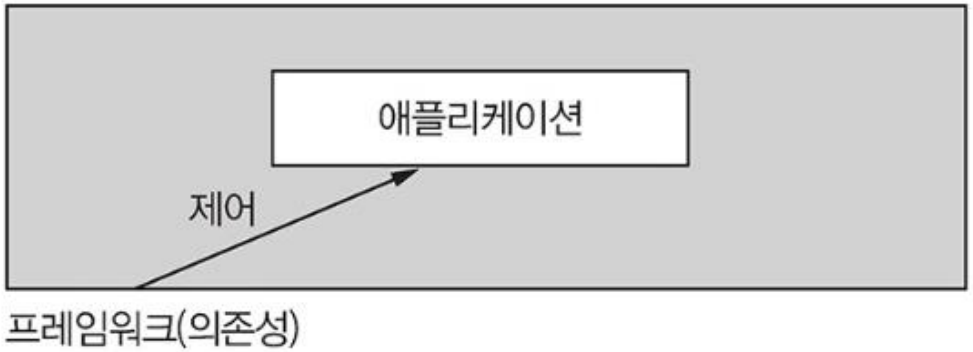
구성 파일에서 정의된 빈 간 관계 구현

IoC를 사용하지 않을 때



애플리케이션은 필요한 의존성을 실행하고 제어(사용)한다.

IoC를 사용할 때



애플리케이션은 프레임워크(의존성)로 제어되어 실행된다.

@Autowired 애너테이션을 사용한 빈 주입

@Autowired로 클래스 필드를 이용한 값 주입

```
@Component
public class Person {

  private String name = "Ella";

  @Autowired
  private Parrot parrot;

  // ...
}
```

스테레오타입 애너테이션인 @Component는 스프링이 이 클래스(Person) 타입의 빈을 생성하고 추가하도록 지시한다.

스프링 컨텍스트

스프링이 스프링 컨텍스트에서 빈을 가져와 @Autowired 애너테이션된 필드 값에 직접 설정하도록 지시한다. 이렇게 하면 두 빈 사이에 관계가 설정된다.

@Autowired 에너테이션을 사용한 빈 주입

beans.Parrot2.java

```
package org.scoula.beans;

import org.springframework.stereotype.Component;

@Component
public class Parrot2 {
    private String name = "Koko";

    public String getName() {
        return name;
    }

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

    @Override
    public String toString() {
        return "Parrot : " + name;
    }

}
```

@Autowired 애너테이션을 사용한 빈 주입

beans.Person2.java

```
@Component
public class Person2 {

  private String name = "Ella";

  @Autowired
  private Parrot2 parrot;

  public String getName() {
    return name;
  }

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

  public Parrot2 getParrot() {
    return parrot;
  }

  public void setParrot(Parrot2 parrot) {
    this.parrot = parrot;
  }
}
```

@Autowired 애너테이션을 사용한 빈 주입

config.ProjectConfig2.java

```
package org.scoula.config;

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

@Configuration
@ComponentScan(basePackages = "org.scoula.beans")
public class ProjectConfig2 {

}
```

@Autowired 애너테이션을 사용한 빈 주입

main.Main2.java

```
package org.scoula.main;

import org.scoula.beans.Person2;
import org.scoula.config.ProjectConfig2;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class Main2 {

    public static void main(String[] args) {
        var context = new AnnotationConfigApplicationContext(ProjectConfig2.class);

        Person2 p = context.getBean(Person2.class);

        System.out.println("Person's name: " + p.getName());
        System.out.println("Person's parrot: " + p.getParrot());
    }

}
```

```
Person's name: Ella
Person's parrot: Parrot : Koko
```

@Autowired 애너테이션을 사용한 빈 주입

@Autowired를 사용하여 생성자로 값 주입

```
@Component
public class Person {

  private String name = "Ella";

  private final Parrot parrot;

  @Autowired
  public Person(Parrot parrot) {
    this.parrot = parrot;
  }

  // ...

}
```

@Component 스테레오타입 애너테이션은
스프링이 이 클래스(Person) 타입의 컨텍스트에
빈을 생성하고 추가하도록 지시한다.

스프링 컨텍스트

스프링이 Person 타입의 빈을 생성할 때
@Autowired 애너테이션이 달린 생성자를 호출한다.
스프링은 매개변수 값으로 컨텍스트에서
Parrot 타입의 빈을 전달한다.

@Autowired 애너테이션을 사용한 빈 주입

beans.Person3.java

```
@Component
public class Person3 {

  private String name = "Ella";

  private final Parrot2 parrot;

  @Autowired
  public Person3(Parrot2 parrot) {
    this.parrot = parrot;
  }

  public String getName() {
    return name;
  }

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

  public Parrot2 getParrot() {
    return parrot;
  }

}
```

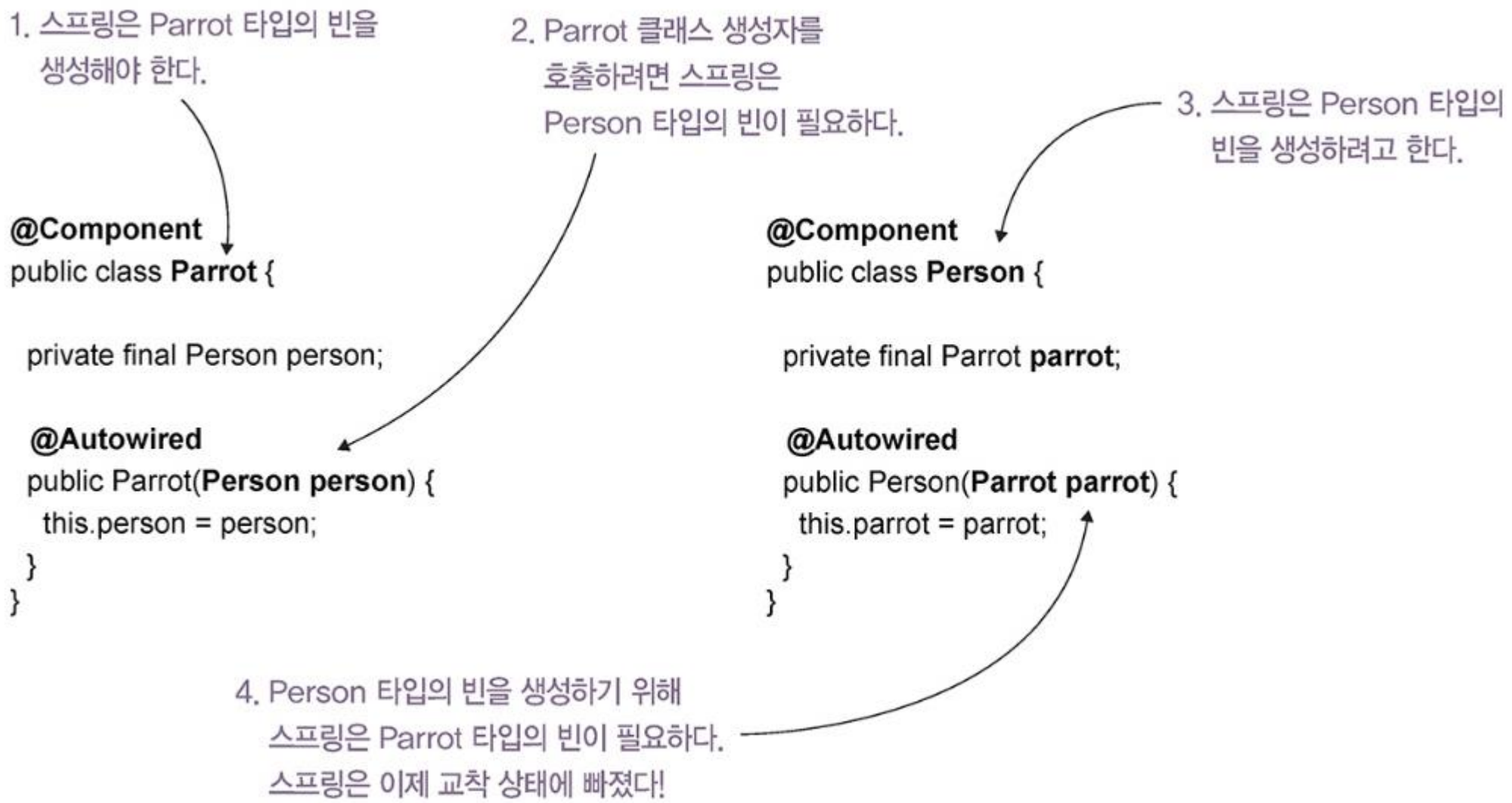
@Autowired 애너테이션을 사용한 빈 주입

setter를 이용한 의존성 주입 사용

```
@Component
public class Person {

  private String name = "Ella";

  private Parrot parrot;

  public String getName() {
    return name;
  }

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

  public Parrot getParrot() {
    return parrot;
  }

  @Autowired
  public void setParrot(Parrot parrot) {
    this.parrot = parrot;
  }
}
```

순환 의존성 다루기

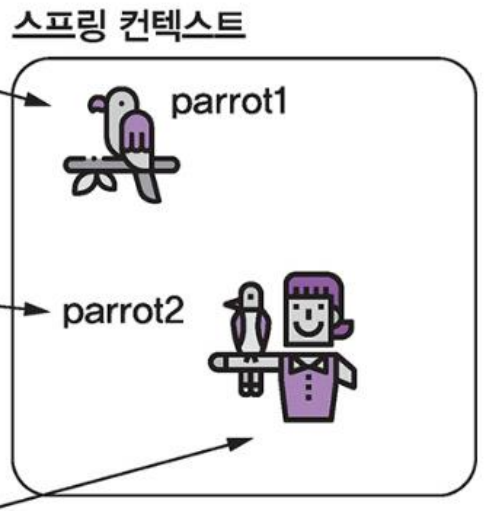
순환 의존성



스프링 컨텍스트에서 여러 빈 중 선택하기


```
@Configuration
public class ProjectConfig {

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

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

  @Bean
  public Person person(Parrot parrot2) {
    Person p = new Person();
    p.setName("Ella");
    p.setParrot(parrot2);
    return p;
  }
}
```

우리는 컨텍스트에 빈 두 개를 정의한다.
이 빈 두 개는 이를 생성하는 메서드 이름인
parrot1과 parrot2를 갖는다.
또 Person 타입의 빈도 하나 정의한다.



스프링은 우리가 정의한 매개변수 이름과
동일한 이름의 빈 값을 찾아 전달한다.

스프링 컨텍스트에서 여러 빈 중 선택하기

@Qualifier로 대상 지정하기

- @Qualifier(value="빈 이름")으로 연결할 빈 이름 지정

```
public Person person(@Qualifier("parrot2") Parrot parrot) {
   Person p = new Person();
   p.setName("Ella");
   p.setParrot(parrot);
   return p;
 }
```

스프링 컨텍스트에서 여러 빈 중 선택하기

생성자 주입에서 대상 지정하기

```
@Component
public class Person {

  private String name = "Ella";

  private final Parrot parrot;

  public Person(Parrot parrot2) {
    this.parrot = parrot2;
  }

  public String getName() {
    return name;
  }

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

  public Parrot getParrot() {
    return parrot;
  }

}
```

```
public class Parrot {

  private String name;

  public String getName() {
    return name;
  }

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

  @Override
  public String toString() {
    return "Parrot : " + name;
  }
}
```

스프링 컨텍스트에서 여러 빈 중 선택하기

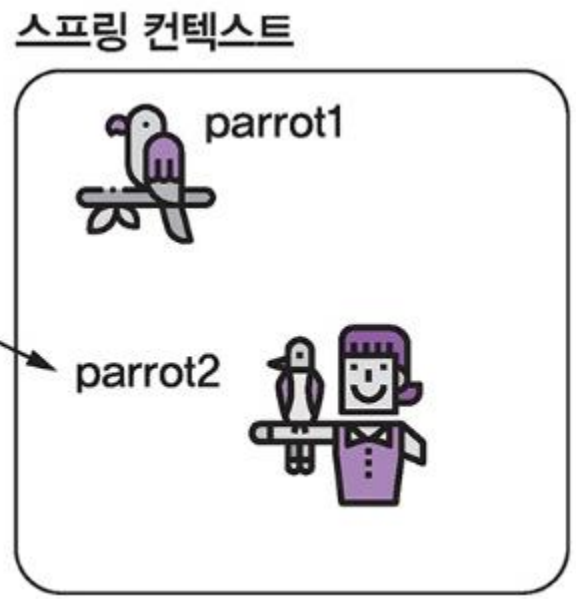
config.ProjectConfig.java

```
@Configuration
@ComponentScan(basePackages = "beans")
public class ProjectConfig {

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

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

스프링 컨텍스트에서 여러 빈 중 선택하기


```
@Component
public class Person {

  private String name = "Ella";

  private final Parrot parrot;

  public Person(Parrot parrot2) {
    this.parrot = parrot2;
  }

  // 코드 생략
}
```

스프링이 Person 타입의 빈을 생성할 때
생성자의 매개변수 값도 전달해야 한다.
타입이 동일한 빈을 여러 개 사용할 수 있다면
스프링은 매개변수와 이름이 동일한 빈을 선택한다.



스프링 컨텍스트에서 여러 빈 중 선택하기


```
@Component
public class Person {

  private String name = "Ella";

  private final Parrot parrot;

  public Person(@Qualifier("parrot2") Parrot parrot) {
    this.parrot = parrot;
  }

  public String getName() {
    return name;
  }

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

  public Parrot getParrot() {
    return parrot;
  }

}
```