4. 생명주기와 범위

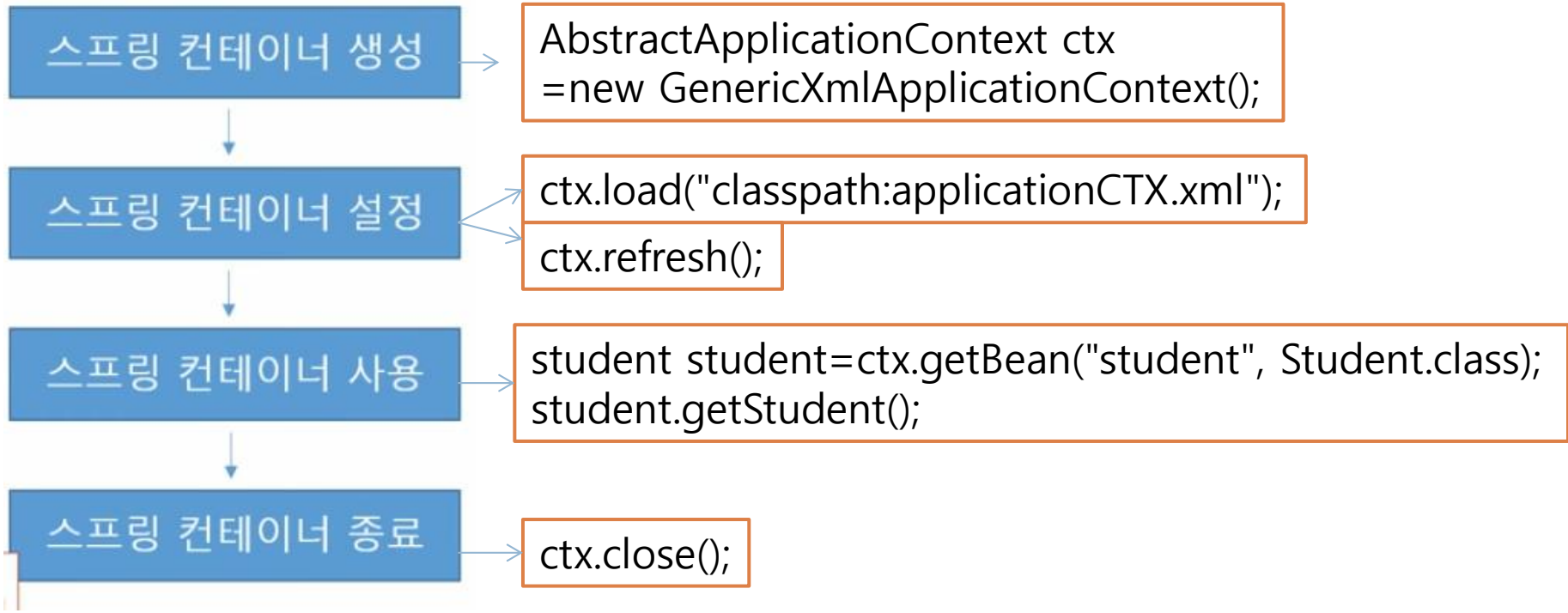
스프링 컨테이너의 생명주기

빈의 생명 주기

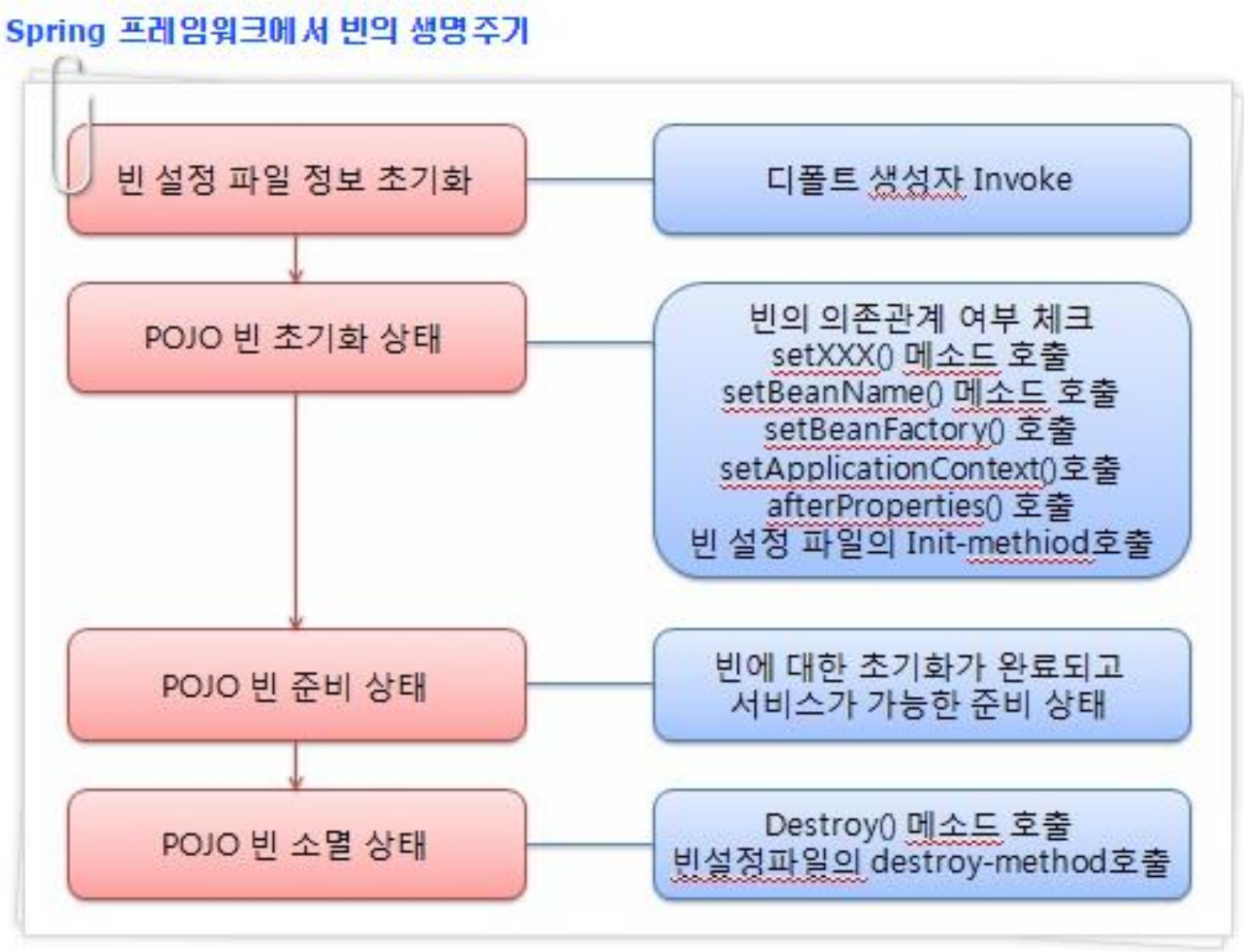
스프링 빈의 범위

1. 스프링 컨테이너의 생명 주기

- 컨테이너 생명주기



2. 스프링 빈의 생명주기


Spring 프레임워크에서 빈의 생명주기
빈 설정 파일 정보 초기화
디폴트 생성자 Invoke
POJO 빈 초기화 상태
빈의 의존관계 여부 체크
setXXX() 메소드 호출
setBeanName() 메소드 호출
setBeanFactory() 호출
setApplicationContext()호출
afterProperties() 호출
빈 설정 파일의 Init-methiod호출
POJO 빈 준비 상태
빈에 대한 초기화가 완료되고
서비스가 가능한 준비 상태
POJO 빈 소멸 상태
Destroy() 메소드 호출
빈설정파일의 destroy-method호출

2. 스프링 빈 생명주기

1) implements InitializingBean, DisposableBean

```
@Override
public void afterPropertiesSet() throws Exception {
    // TODO Auto-generated method stub
    System.out.println("afterPropertiesSet()");
}

@Override
public void destroy() throws Exception {
    // TODO Auto-generated method stub
    System.out.println("destroy()");
}
```

빈 초기화 과정에서 호출 됩니다.

빈 소멸 과정에서 생성 됩니다.

```
GenericXmlApplicationContext ctx = new GenericXmlApplicationContext();

ctx.load("classpath:applicationCTX.xml");

ctx.refresh();

ctx.close();
```

[참고 하세요]
ctx.close()의 경우 컨테이너가 소멸 하는 단계입니다.
컨테이너가 소멸 하면, 빈은 자동 소멸 됩니다.
빈만 소멸하게 한다면, student.destroy() API를 이용하면 됩니다. 한번 해보세요.^^

2) @PostConstruct, @PreDestroy

속성명	설명
init-method	인스턴스 최기화 시 호출되는 메서드
destory-method	인스턴스 소멸 시 호출되는 메서드

```
}

@PreDestroy
public void destroyMethod() {
    System.out.println("destroyMethod()");
}
```

빈 소멸 과정에서 생성 됩니다.

```
GenericXmlApplicationContext ctx = new GenericXmlApplicationContext();

ctx.load("classpath:applicationCTX.xml");

ctx.refresh();

ctx.close();
```

스프링 빈의 범위(scope)

```xml
<bean id="student" class="com.javalec.ex.Student" scope="singleton">
    <constructor-arg value="홍길순"></constructor-arg>
    <constructor-arg value="30"></constructor-arg>
</bean>
```

```
Student student1 = ctx.getBean("student", Student.class);
System.out.println("이름 : " + student1.getName());
System.out.println("나이 : " + student1.getAge());

System.out.println("==========");

Student student2 = ctx.getBean("student", Student.class);
student2.setName("홍길자");
student2.setAge(100);

System.out.println("이름 : " + student1.getName());
System.out.println("나이 : " + student1.getAge());

System.out.println("==========");

if(student1.equals(student2)) {
    System.out.println("student1 == student2");
} else {
    System.out.println("student1 != student2");
}
```

```
이름 : 홍길순
나이 : 30
==========
이름 : 홍길자
나이 : 100
==========
student1 == student2
```

빈의 영역

- IoC 컨터이너에서 Spring 빈 인스턴스가 생성될때 생성되는 영역을 설정
- scope attribute

scope	설명
signleton	ApplicationContext 컨터이너 당 하나의 인스턴스 생성 (디폴트)
prototype	getBean() 메소드가 생성될때 마다 인스턴스 생성
request	HTTP request 영역안에서 인스턴스 생성함
session	HTTP session 영역안에서 인스턴스 생성함
global sesson	전역 HTTP session 영역안에서 인스턴스가 생성됨

빈의 영역 사용 예

```
<bean id="customerService"
class="com.miya.order.CustomerServiceImpl" scope="prototype">
```

```
@Bean
@Scope("prototype")
CustomerService customerService(){
        return new CustomerServiceImpl();
}
```

```
CustomerService bean1=(CustomerService)ctx.getBean("customerService");
CustomerService bean2=(CustomerService)ctx.getBean("customerService");
CustomerService bean3=(CustomerService)ctx.getBean("customerService");
System.out.println(bean1);
System.out.println(bean2);
System.out.println(bean3);
```