



Chapter 9. 빈 생명주기 콜백

빈 생명주기 콜백 시작

- 데이터베이스 커넥션 풀이나, 네트워크 소켓처럼 애플리케이션 시작 시점에 필요한 연결을 미리 해두고, 애플리케이션 종료 시점에 연결을 모두 종료하는 작업을 진행하려면, 객체의 초기화와 종료 작업이 필요하다.
- 이 작업을 어떻게 진행하는지에 대한 예제
 - 외부 네트워크에 미리 연결하는 객체를 하나 생성한다고 가정
 - 이 `NetworkClient`는 애플리케이션 시작 시점에 `connect()` 를 호출해서 연결을 맺어두어야 하고, 애플리케이션이 종료되면 `disconnect()`를 호출해서 연결을 끊어야 한다.

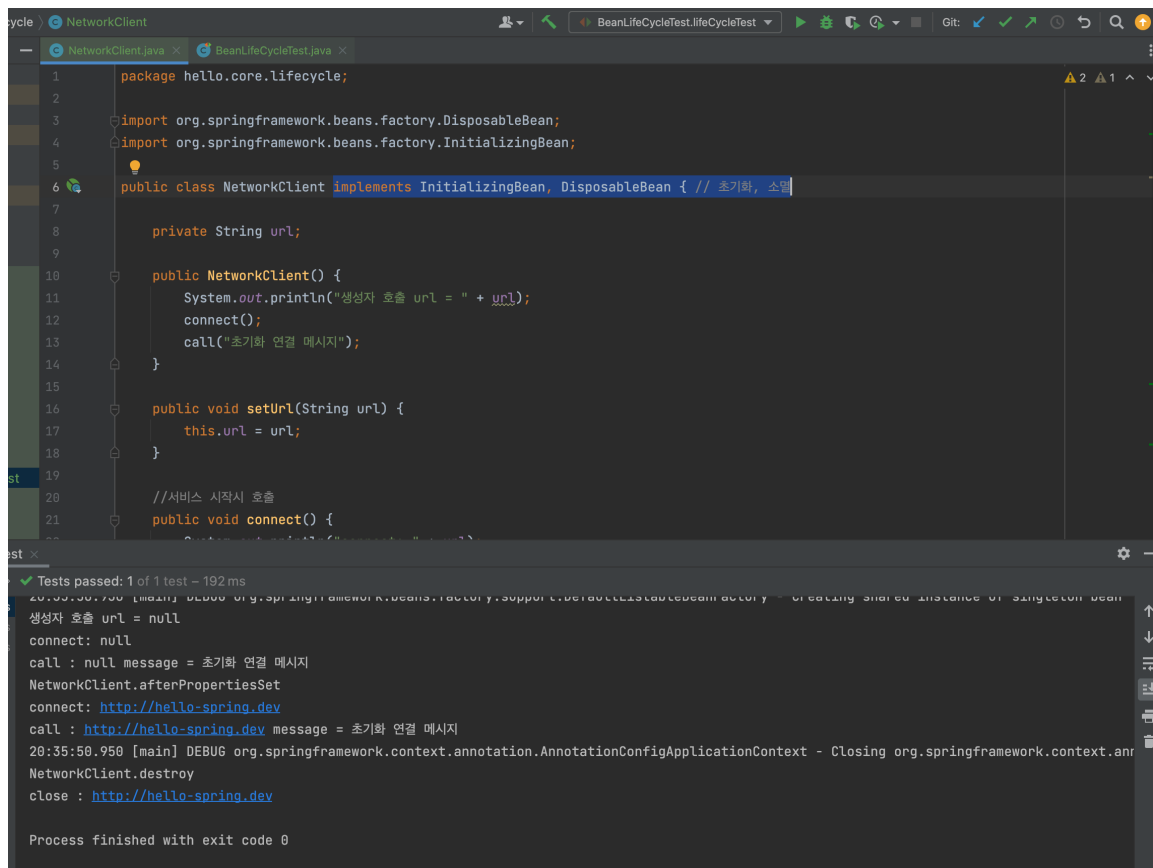
- 스프링 빈은 다음과 같은 라이프 사이클을 가진다.

스프링 컨테이너 생성 → 스프링 빈 생성 → 의존관계 주입 → 초기화 콜백 → 사용 → 소멸전 콜백 → 스프링 종료

- 스프링 빈 생성 : 생성자
 - 의존관계 주입 : setter injection
 - 초기화 콜백 : 빈이 생성되고, 빈의 의존관계 주입이 완료된 후 호출
 - 소멸전 콜백 : 빈이 소멸되기 직전에 호출
- 객체의 생성과 초기화를 분리하자!
 - 스프링은 크게 3가지 방법으로 빈 생명주기 콜백을 지원한다.
 - 인터페이스
 - 설정 정보에 초기화 메서드, 종료 메서드 지정
 - `@PostConstruct`, `@PreDestroy` 애노테이션 지원

인터페이스 InitializingBean, DisposableBean

- 인터페이스 사용



```
package hello.core.lifecycle;

import org.springframework.beans.factory.DisposableBean;
import org.springframework.beans.factory.InitializingBean;

public class NetworkClient implements InitializingBean, DisposableBean { // 초기화, 소멸

    private String url;

    public NetworkClient() {
        System.out.println("생성자 호출 url = " + url);
        connect();
        call("초기화 연결 메시지");
    }

    public void setUrl(String url) {
        this.url = url;
    }

    //서비스 시작시 호출
    public void connect() {
        System.out.println("connect: " + url);
    }

    public void call(String message) {
        System.out.println("call: " + message);
    }

    public void afterPropertiesSet() {
        System.out.println("afterPropertiesSet: " + url);
    }

    public void destroy() {
        System.out.println("destroy: " + url);
    }
}
```

Tests passed: 1 of 1 test - 192 ms

```
20:35:50.700 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton bean 'networkClient'
생성자 호출 url = null
connect: null
call : null message = 초기화 연결 메시지
NetworkClient.afterPropertiesSet
connect: http://hello-spring.dev
call : http://hello-spring.dev message = 초기화 연결 메시지
20:35:50.950 [main] DEBUG org.springframework.context.annotation.AnnotationConfigApplicationContext - Closing org.springframework.context.annotation.AnnotationConfigApplicationContext@...
NetworkClient.destroy
close : http://hello-spring.dev

Process finished with exit code 0
```

```
32 }
33
34 @Override
35 public void afterPropertiesSet() throws Exception {
36     System.out.println("NetworkClient.afterPropertiesSet");
37     connect();
38     call("초기화 연결 메시지");
39 }
40
41 @Override
42 public void destroy() throws Exception {
43     System.out.println("NetworkClient.destroy");
44     disconnect();
45 }
46 }
47
```

Test results:

```
Tests passed: 1 of 1 test - 192 ms
20:35:50.700 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton bean 'networkClient'
생성자 호출 url = null
connect: null
call : null message = 초기화 연결 메시지
NetworkClient.afterPropertiesSet
connect: http://hello-spring.dev
call : http://hello-spring.dev message = 초기화 연결 메시지
20:35:50.950 [main] DEBUG org.springframework.context.annotation.AnnotationConfigApplicationContext - Closing org.springframework.context.annotation.AnnotationConfigApplicationContext: org.springframework.context.annotation.AnnotationConfigApplicationContext@...
NetworkClient.destroy
close : http://hello-spring.dev
Process finished with exit code 0
```

- 초기화, 소멸 인터페이스 단점
 - 스프링 전용 인터페이스라서, 해당 코드가 스프링 전용 인터페이스에 의존한다.
 - 초기화, 소멸 메소드의 이름을 변경할 수 없다.
 - 외부라이브러리에 의존하기에, 코드를 변경할 수 없다.
- 참고
 - 인터페이스는 스프링 초창기! 거의 사용 안한다.

빈 등록 초기화, 소멸 메서드

- NetworkClient class에 init(), close() 메소드 생성 후,
- Bean에 (initMethod, destroyMethod) 등록

The screenshot shows the `NetworkClient.java` file in an IDE. The code defines three methods: `disconnect()`, `init()`, and `close()`. The `init()` method calls `connect()` and `call("초기화 연결 메시지")`. The `close()` method calls `disconnect()`. The test results pane shows that the test passed, with the following output:

```

Tests passed: 1 of 1 test - 190 ms
20:41:38.155 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton bean 'networkClient'
생성자 호출 url = null
connect: null
call : null message = 초기화 연결 메시지
NetworkClient.init
connect: http://hello-spring.dev
call : http://hello-spring.dev message = 초기화 연결 메시지
20:41:38.155 [main] DEBUG org.springframework.context.annotation.AnnotationConfigApplicationContext - Closing org.springframework.context.annotation.AnnotationConfigApplicationContext@...
NetworkClient.close
close : http://hello-spring.dev

Process finished with exit code 0

```

The screenshot shows the `BeanLifeCycleTest.java` file in an IDE. The code defines a `BeanLifeCycleTest` class with a `lifeCycleTest()` method. The `lifeCycleTest()` method creates an `AnnotationConfigApplicationContext` with `LifeCycleConfig.class`, gets a `NetworkClient` bean, and calls `close()`. The `LifeCycleConfig` class is a `@Configuration` class with a `@Bean` method that creates a `NetworkClient` bean. The test results pane shows that the test passed, with the following output:

```

Tests passed: 1 of 1 test - 190 ms
20:41:38.155 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton bean 'networkClient'
생성자 호출 url = null
connect: null
call : null message = 초기화 연결 메시지
NetworkClient.init
connect: http://hello-spring.dev
call : http://hello-spring.dev message = 초기화 연결 메시지
20:41:38.155 [main] DEBUG org.springframework.context.annotation.AnnotationConfigApplicationContext - Closing org.springframework.context.annotation.AnnotationConfigApplicationContext@...
NetworkClient.close
close : http://hello-spring.dev

Process finished with exit code 0

```

- 설정 정보 사용 특징
 - 메서드 이름을 자유롭게 정의 가능
 - 스프링 빈이 스프링 코드에 의존하지 않는다
 - 코드가 아니라 설정정보를 사용하기 때문에, 코드를 고칠 수 없는 외부 라이브러리에도 초기화, 종료 메서드를 적용할 수 있다.

- @Bean의 destroyMethod 속성의 특별한 기능
 - 디폴트 값은 (inferred) - 추론
 - 이 추론 기능은 close, shutdown 이라는 이름의 메서드를 자동으로 호출해줌
 - 추론 기능을 사용하지 싶다면 destroyMethod="" 라고 지정하면 됨

애노테이션 @PostConstruct, @PreDestroy

- 이걸 사용하면 됨 !!

```
cycle ) NetworkClient > close
NetworkClient.java x BeanLifeCycleTest.java x
32     System.out.println("close : " + url);
33 }
34
35 @PostConstruct
36 public void init() {
37     System.out.println("NetworkClient.init");
38     connect();
39     call("초기화 연결 메시지");
40 }
41
42 @PreDestroy
43 public void close() {
44     System.out.println("NetworkClient.close");
45     disconnect();
46 }
47 }
48
st

✓ Tests passed: 1 of 1 test - 191 ms
성명서 포트 URL = null
connect: null
call : null message = 초기화 연결 메시지
NetworkClient.init
connect: http://hello-spring.dev
call : http://hello-spring.dev message = 초기화 연결 메시지
20:49:09.661 [main] DEBUG org.springframework.context.annotation.AnnotationConfigApplicationContext - Closing org.springframework.context.an
NetworkClient.close
close : http://hello-spring.dev

Process finished with exit code 0
```

```
12 public void lifeCycleTest() {
13     ConfigurableApplicationContext ac = new AnnotationConfigApplicationContext(LifeCycleConfig.class);
14     NetworkClient client = ac.getBean(NetworkClient.class);
15     ac.close();
16 }
17
18 @Configuration
19 static class LifeCycleConfig{
20     // @Bean(initMethod = "init", destroyMethod = "close")
21     @Bean
22     public NetworkClient networkClient() {
23         NetworkClient networkClient = new NetworkClient();
24         networkClient.setUrl("http://hello-spring.dev");
25         return networkClient;
26     }
27 }
28 }
```

Tests passed: 1 of 1 test - 191 ms

connect: null
call : null message = 초기화 연결 메시지
NetworkClient.init
connect: http://hello-spring.dev
call : http://hello-spring.dev message = 초기화 연결 메시지
20:49:09.661 [main] DEBUG org.springframework.context.annotation.AnnotationConfigApplicationContext - Closing org.springframework.context.annotation.AnnotationConfigApplicationContext: beans = {networkClient}
NetworkClient.close
close : http://hello-spring.dev
Process finished with exit code 0

- 단점
 - 외부 라이브러리에 적용하지 못한다.
 - 외부 라이브러리를 사용해야한다면 @Bean에 (initMethod, destroyMethod) 사용