Spring Framework

7. DI 애플리케이션 작성(2)

Tacademy

CONTENTS

학습 목표

- jUnit의 개요와 특징에 대하여 이해할 수 있습니다.
- jUnit을 사용한 DI 테스트 클래스를 작성할 수 있습니다.
- Spring-Test를 사용한 DI 테스트 클래스를 작성할 수 있습니다.

1. jUnit의 개요와 특징

jUnit의 개요

Java에서 독립된 단위테스트(Unit Test)를 지원해주는 **프레임워크**이다.

❖ 단위테스트(Unit Test)란?

- 소스 코드의 특정 모듈이 의도된 대로 정확히 작동하는지 검증하는 절차, 즉 **모든 함수와 메소드에 대한 테스트 케이스(Test case)를 작성하는 절차**를 말한다.
- jUnit은 보이지 않고 숨겨진 단위 테스트를 끌어내어 정형화시켜 단위테스트를 쉽게 해주는 테스트 지원 프레임워크다.

jUnit의 특징

- TDD의 창시자인 Kent Beck과 디자인 패턴 책의 저자인 Erich Gamma가 작성했다.
- 단정(assert) 메서드로 테스트 케이스의 수행 결과를 판별한다.
  예) assertEquals(예상 값, 실제 값)
- jUnit4부터는 테스트를 지원하는 어노테이션을 제공한다.
  @Test @Before @After
- 각 @Test 메서드가 호출할 때 마다 새로운 인스턴스를 생성하여 독립적인 테스트가 이루어지도록 한다.

2. jUnit을 사용한 DI 테스트 클래스

jUnit 라이브러리 설치

http://mvnrepository.com에 접근한다.

↓

junit으로 검색한다.

↓

junit 4.12버전을 pom.xml에 추가한다.

```
<!-- https://mvnrepository.com/artifact/junit/junit -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
</dependency>
```

jUnit에서 테스트를 지원하는 어노테이션(Annotation) (1)

@Test

- @Test가 선언된 메서드는 테스트를 수행하는 메소드가 된다.
- Junit은 각각의 테스트가 서로 영향을 주지 않고 독립적으로 실행됨을 원칙으로 하므로 @Test 마다 객체를 생성한다.

@Ignore

- @Ignore가 선언된 메서드는 테스트를 실행하지 않게 한다.

@Before

- @Before가 선언된 메서드는 @Test 메소드가 실행되기 전에 반드시 실행되어 진다.
- @Test 메소드에서 공통으로 사용하는 코드를 @Before 메소드에 선언하여 사용하면 된다.

jUnit에서 테스트를 지원하는 어노테이션(Annotation) (2)

@After

- @After가 선언된 메서드는 @Test 메소드가 실행된 후 실행된다.

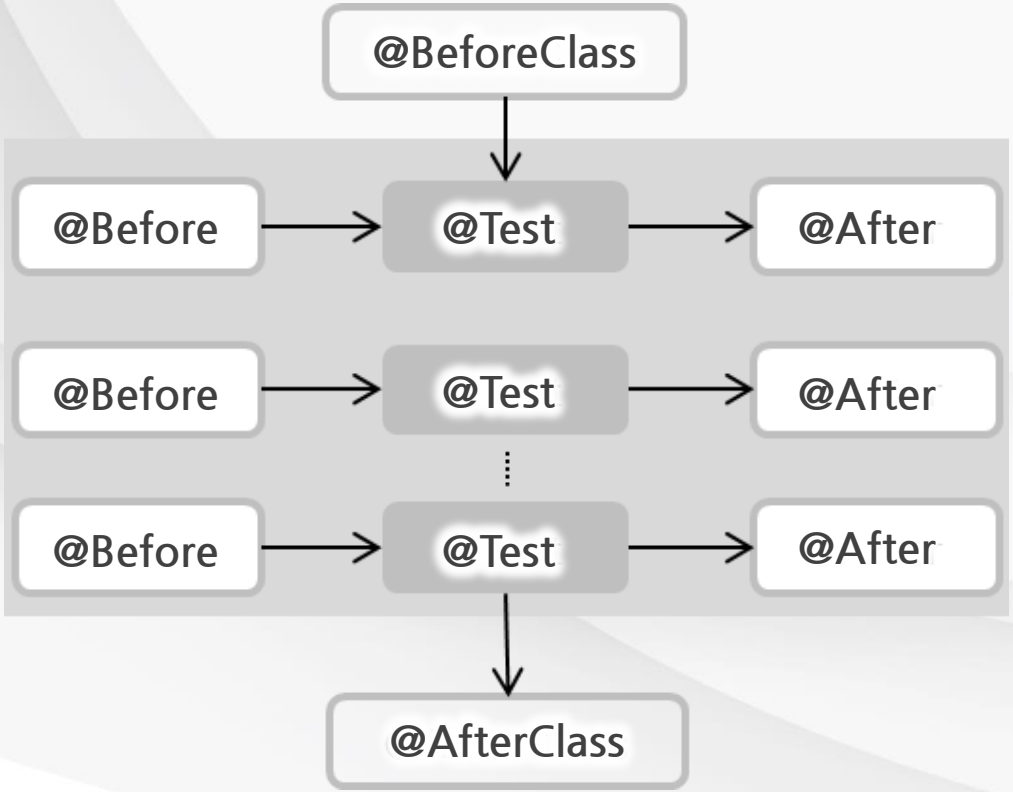
@BeforeClass

- @BeforeClass 어노테이션은 @Test 메소드 보다 먼저 한번만 수행되어야 할 경우에 사용하면 된다.

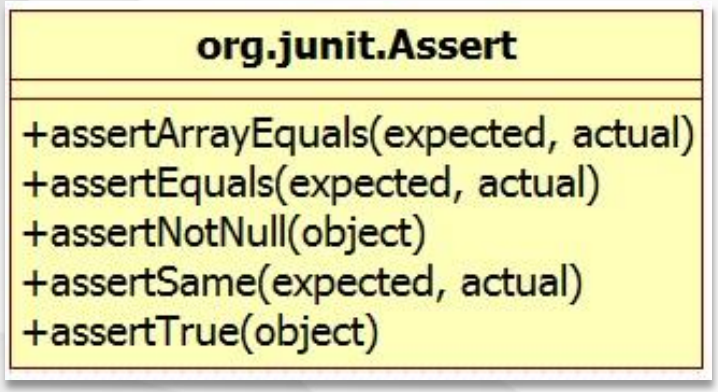
@AfterClass

- @AfterClass 어노테이션은 @Test 메소드 보다 나중에 한번만 수행되어야 할 경우에 사용하면 된다.

jUnit에서 테스트를 지원하는 어노테이션(Annotation) (3)



테스트 결과를 확인하는 단정(assert) 메서드



테스트 결과를 확인하는 단정(assert) 메서드

assertEquals(a, b);	▪ 객체 A와 B가 일치함을 확인한다.
assertArrayEquals(a, b);	▪ 배열 A와 B가 일치함을 확인한다.
assertSame(a, b);	▪ 객체 A와 B가 같은 객체임을 확인한다. ▪ assertEquals 메서드는 두 객체의 값이 같은지 확인하고, assertSame메서드는 두 객체의 레퍼런스가 동일한가를 확인한다. (== 연산자)
assertTrue(a);	▪ 조건 A가 참인가를 확인한다.
assertNotNull(a);	▪ 객체 A가 null이 아님을 확인한다.

◉ 이외에도 다양한 assert 메서드가 존재함.
- http://junit.sourceforge.net/javadoc/org/junit/Assert.html

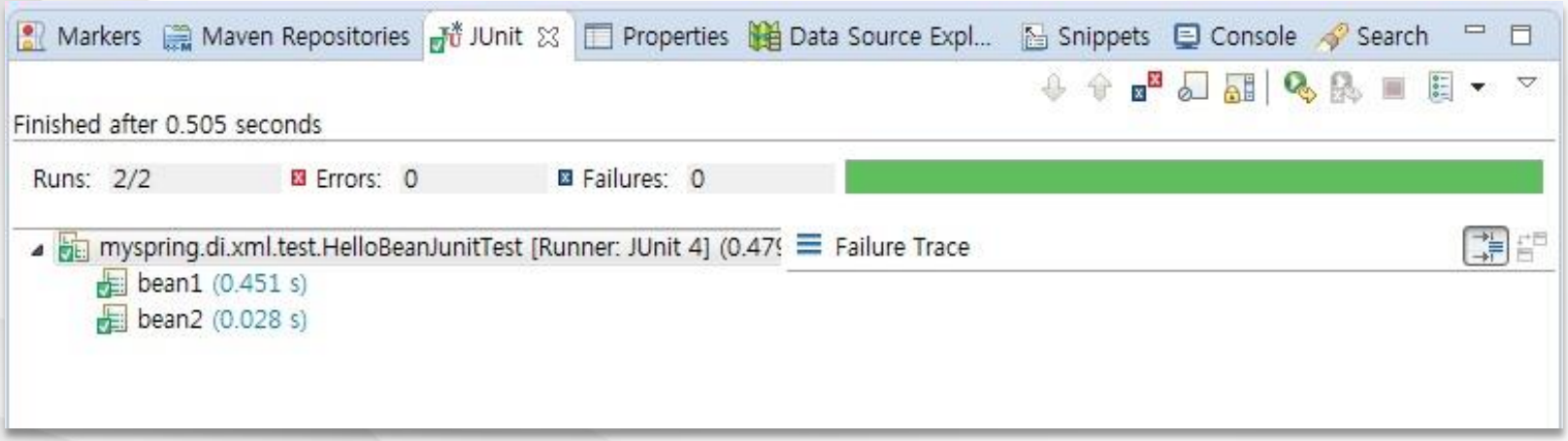
jUnit을 사용한 DI 테스트 클래스(HelloBeanJunitTest.java) 작성

HelloBeanJunitTest.java

```
3  import static org.junit.Assert.*;
9
10 public class HelloBeanJunitTest {
11     private ApplicationContext context;
12
13     @Before
14     public void init() {
15         context = new GenericXmlApplicationContext("config/beans.xml");
16     }
17     @Test
18     public void bean1() {
19         Hello hello = (Hello) context.getBean("hello");
20         assertEquals("Hello Spring", hello.sayHello());
21         hello.print();
22
23         Printer printer = (Printer) context.getBean("printer");
24         assertEquals("Hello Spring",printer.toString());
25     }
26     @Test
27     public void bean2() {
28         Printer printer = (Printer) context.getBean("printer");
29         Printer printer2 = context.getBean("printer",Printer.class);
30
31         assertSame(printer,printer2);
32     }
33 }
```

jUnit을 사용한 DI 테스트 클래스(HelloBeanJunitTest.java) 실행

jUnit을 사용한 DI 테스트 클래스(HelloBeanJunitTest.java) 실행



3. Spring-Test를 사용한 DI 테스트 클래스

Spring-Test 라이브러리 설치

http://mvnrepository.com에 접근한다.

↓

spring-test로 검색한다.

↓

Spring-test 3.2.17버전을 pom.xml에 추가한다.

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-test -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>3.2.17.RELEASE</version>
</dependency>
```

Spring-Test에서 테스트를 지원하는 어노테이션(Annotation) (1)

@RunWith(SpringJUnit4ClassRunner.class)

- @RunWith는 jUnit 프레임워크의 테스트 실행방법을 확장할 때 사용하는 어노테이션이다.
- SpringJUnit4ClassRunner라는 클래스를 지정해주면 jUnit이 테스트를 진행하는 중에 ApplicationContext를 만들고 관리하는 작업을 진행해 준다.
- @RunWith 어노테이션은 각각의 테스트 별로 객체가 생성되더라도 싱글톤(Singleton)의 ApplicationContext를 보장한다.

Spring-Test에서 테스트를 지원하는 어노테이션(Annotation) (2)

@ContextConfiguration

- 스프링 빈(Bean) 설정 파일의 위치를 지정할 때 사용되는 어노테이션이다.

@Autowired

- 스프링DI에서 사용되는 특별한 어노테이션이다.
- 해당 변수에 자동으로 빈(Bean)을 매핑 해준다.
- 스프링 빈(Bean) 설정 파일을 읽기 위해 굳이 GenericXmlApplicationContext를 사용할 필요가 없다.

Spring-Test를 사용한 DI 테스트 클래스(HelloBeanSpringTest.java)작성

*HelloBeanSpringTest.java

```
1  package myspring.di.xml.test;
2
3  import static org.junit.Assert.*;
13
14 @RunWith(SpringJUnit4ClassRunner.class)
15 @ContextConfiguration(locations = "classpath:config/beans.xml")
16 public class HelloBeanSpringTest {
17     @Autowired
18     private ApplicationContext context;
19
20     @Test
21     public void bean1() {
22         Hello hello = (Hello) context.getBean("hello");
23         assertEquals("Hello Spring", hello.sayHello());
24         hello.print();
25         assertEquals(context.getBean("printer").toString(), "Hello Spring");
26
27         Hello hello2 = context.getBean("hello", Hello.class);
28         hello2.print();
29         assertSame(hello,hello2);
30     }
31 }
```

학습정리

지금까지 [DI 애플리케이션 작성(2)]에 대해서 살펴보았습니다.

jUnit의 개요와 특징

단위테스트를 지원하는 프레임워크, assert 메서드를 사용하여 테스트 결과 확인

jUnit을 사용한 DI 테스트 클래스

- jUnit 설치, asset 메서드를 사용하여 테스트 결과 확인
- @Test, @Before 어노테이션 사용

Spring-Test를 사용한 DI 테스트 클래스

Spring-Test 설치, @RunWith(SpringJUnit4ClassRunner.class), @ContextConfiguration, @Autowired 어노테이션 사용