

JUnit

Maven Project

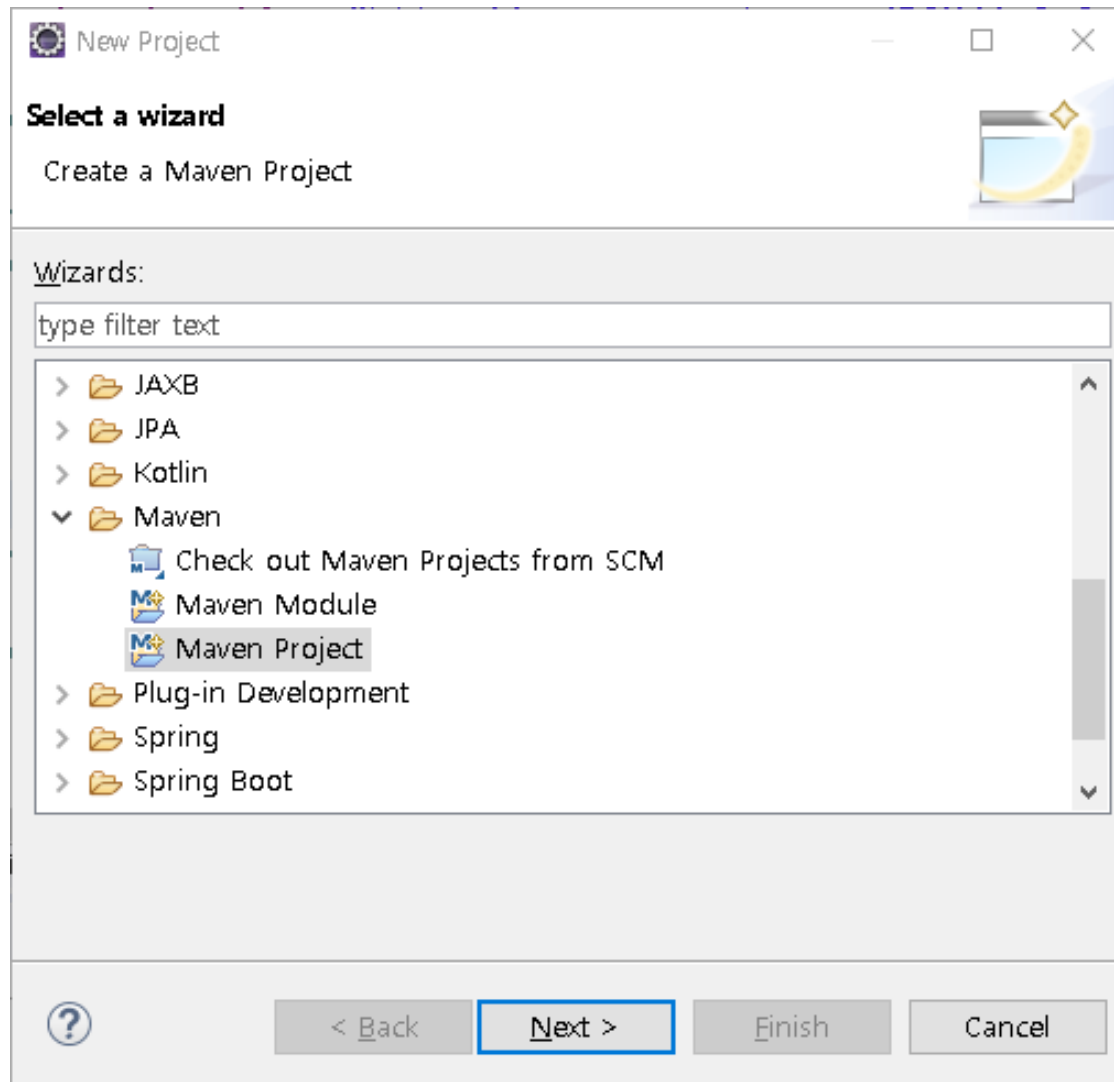
❖ Maven

- Java 자동 빌드 도구
- Maven Project 만들기

Maven Project

❖ Maven Project

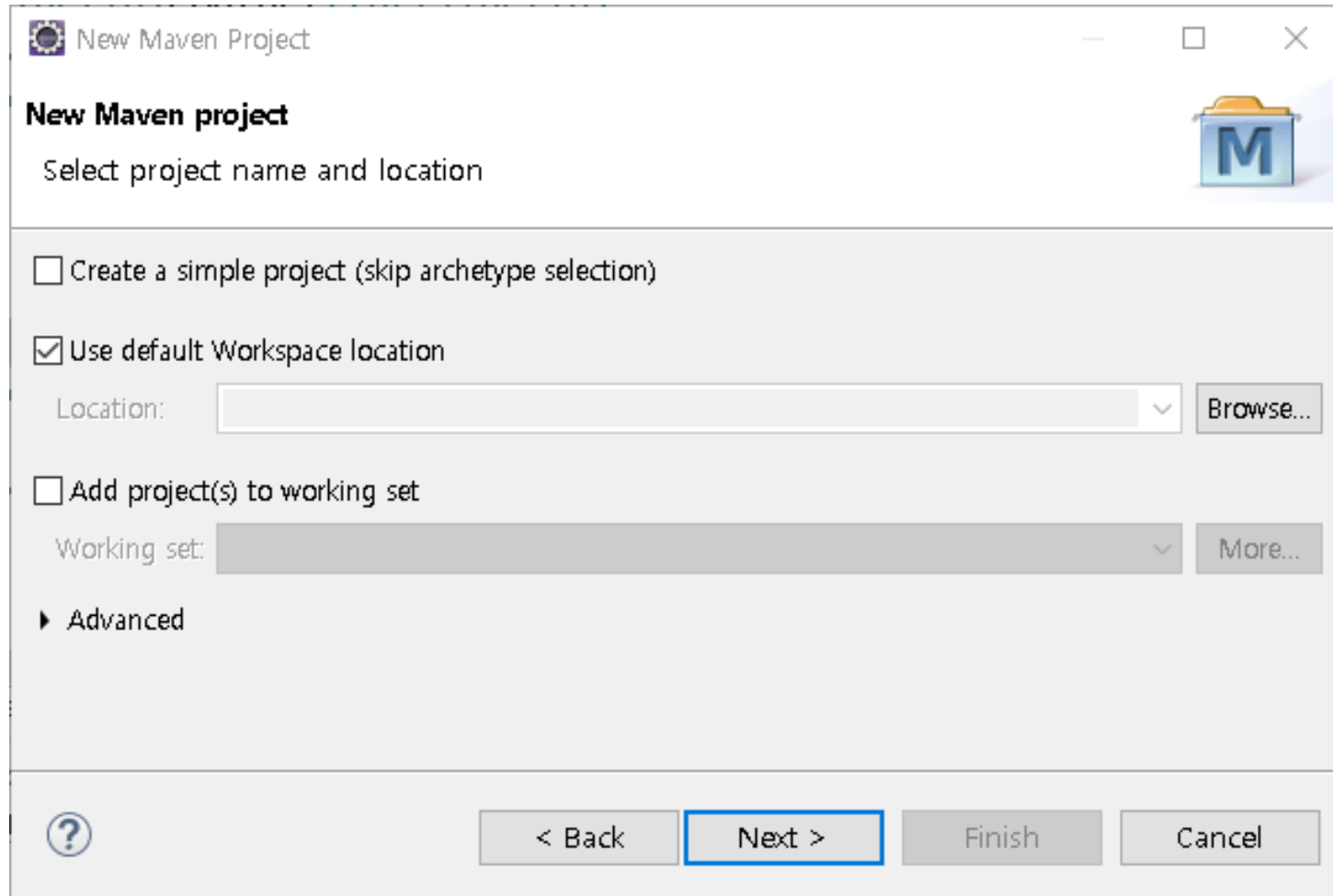
- New > Project ...



Maven Project

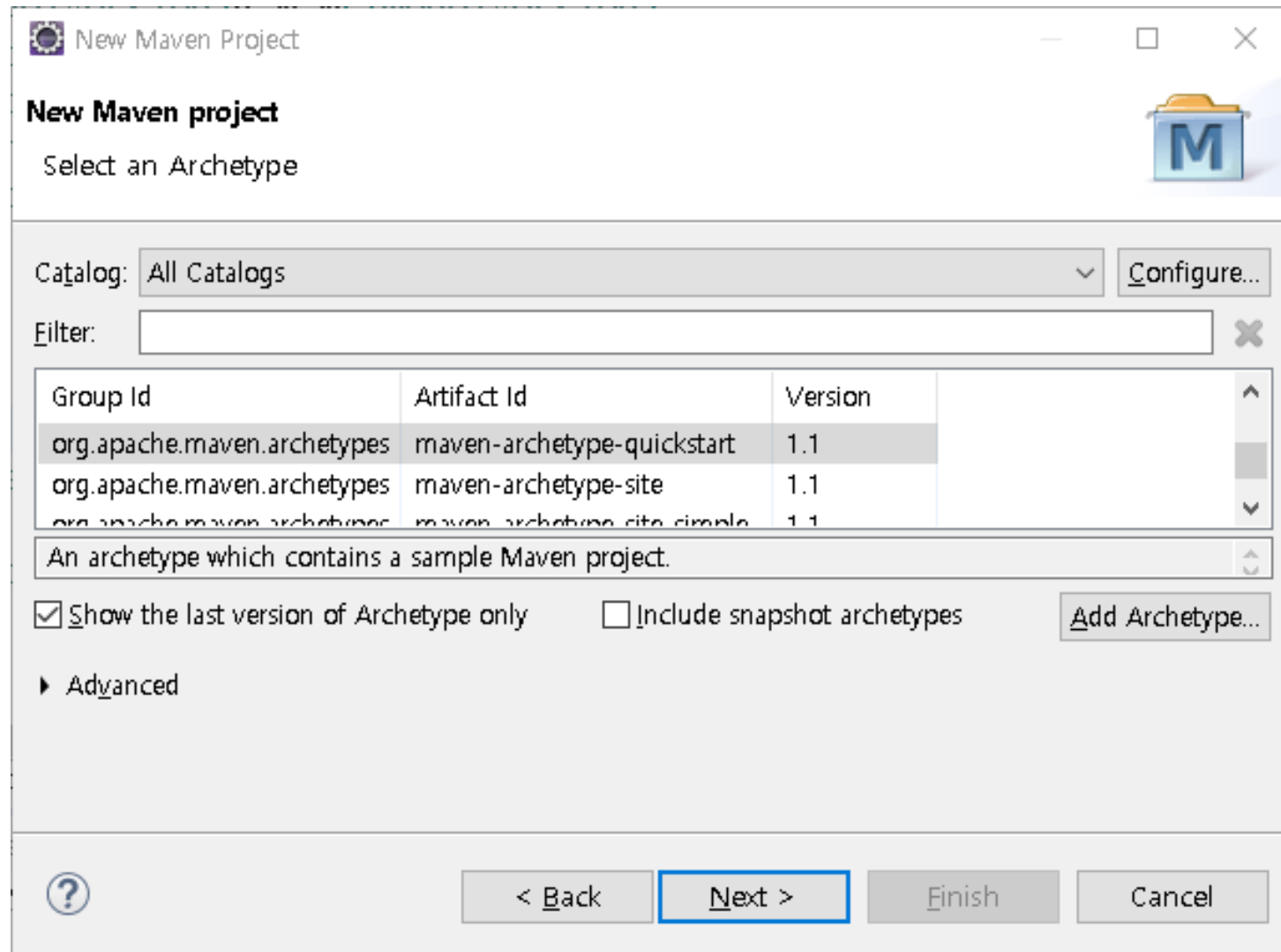
❖ Maven Architecture

- Maven에서 정의한 대표적인 프로젝트 템플릿(구조)



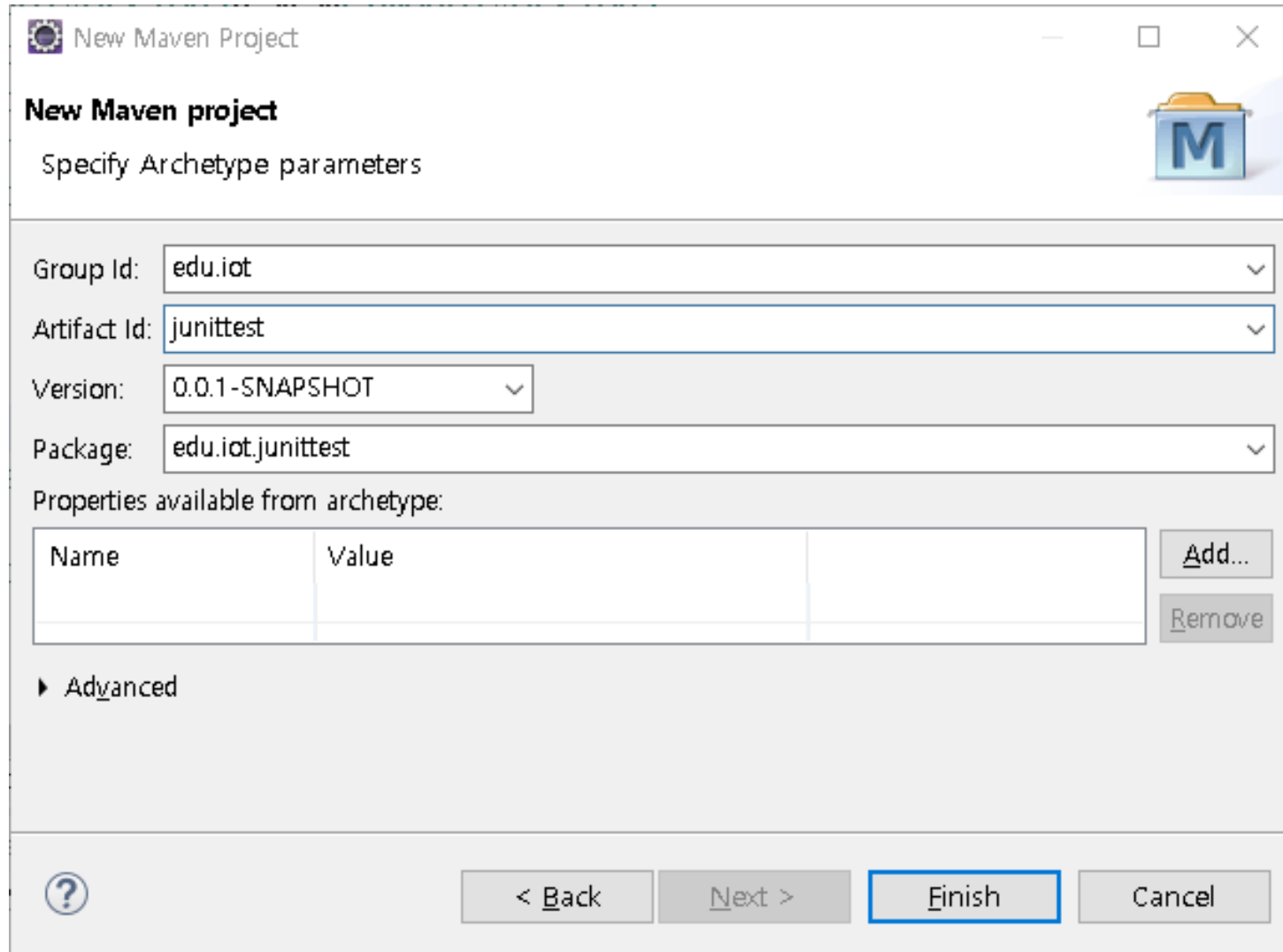
Maven Project

❖ Maven Architecture 선택



Maven Project

❖ base 패키지 지정(Group Id, Artifact Id)



New Maven Project

New Maven project
Specify Archetype parameters

Group Id:

Artifact Id:

Version:











Package:

Properties available from archetype:

Name	Value

▶ Advanced

❖ 프로젝트 기본 구조

- ▼  junittest
 - ▼  src/main/java
 - >  edu.iot.junittest
 - ▼  src/test/java
 - >  edu.iot.junittest
 - >  JRE System Library [J2SE-1.5]
 - >  Maven Dependencies
 - >  src
 - >  target
 -  pom.xml

소스를 위한 패키지

테스트를 위한 패키지

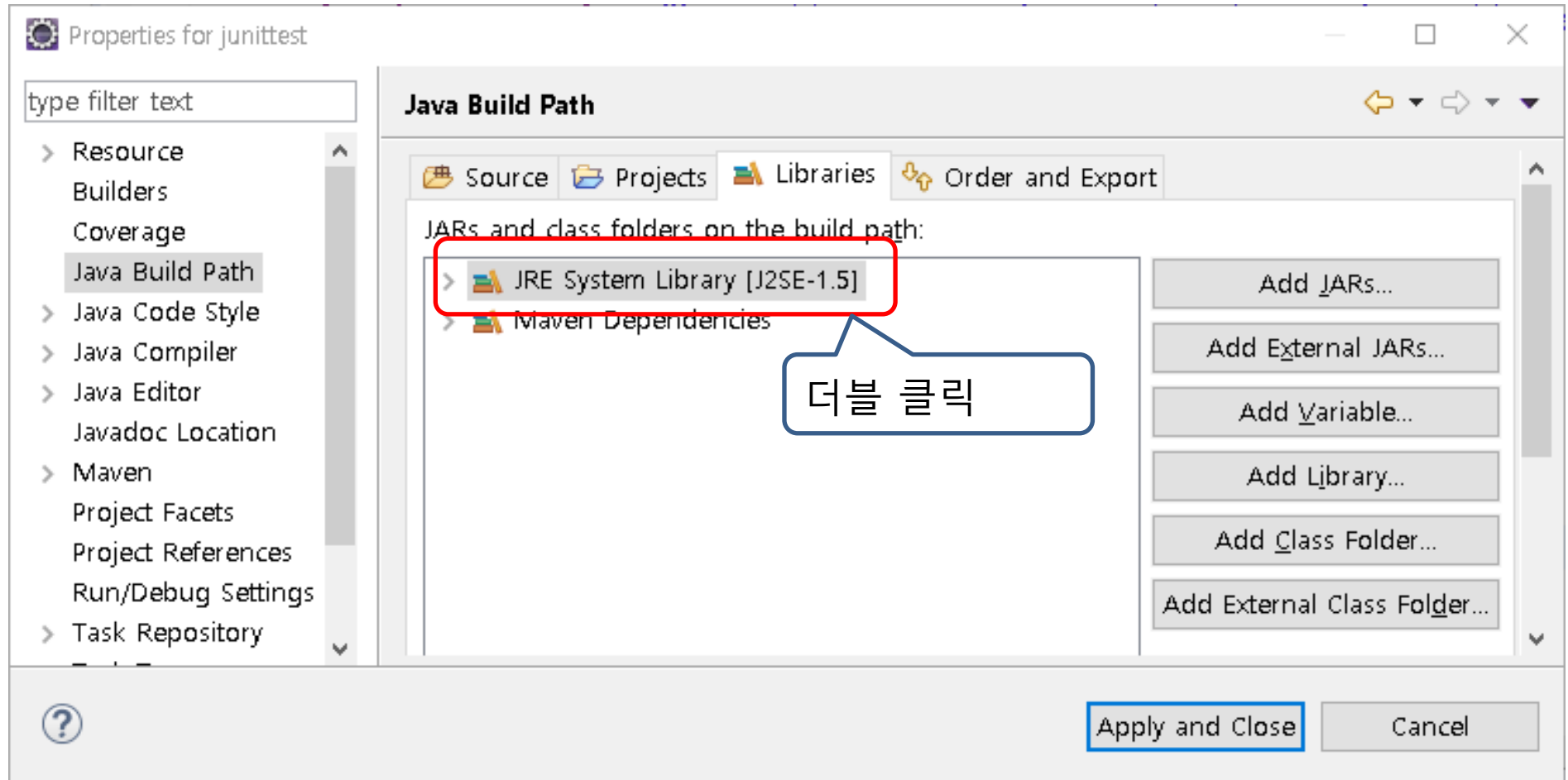
JRE 버전 : 1.8로 수정 필요

메이븐 설정 파일

Maven Project

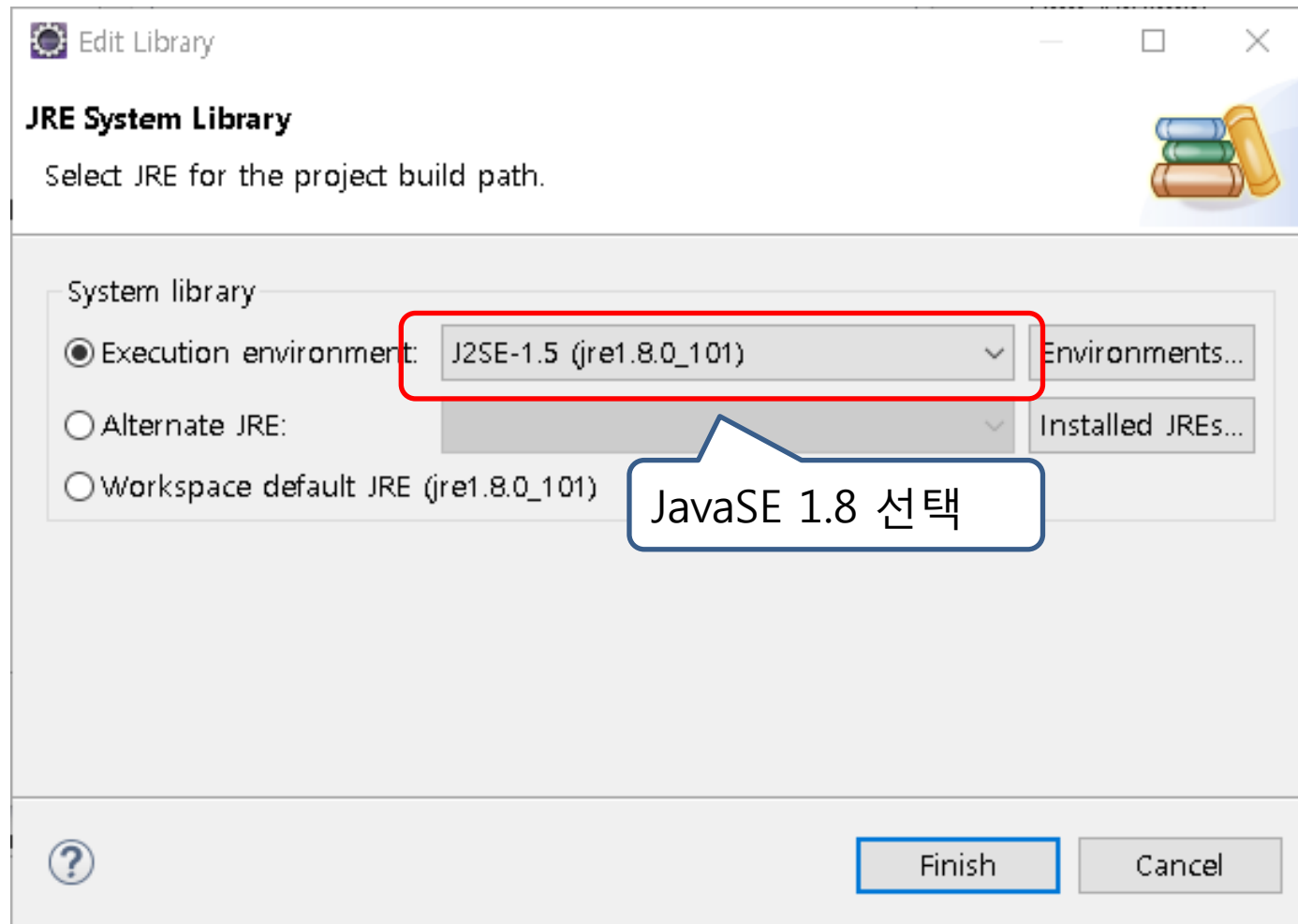
❖ JRE 버전 수정

- 프로젝트 >> Property > Libraries



Maven Project

❖ JRE 버전 수정



Maven Project

❖ JUnit 설정

- 디폴트 버전 : 3.8.1
- pom.xml

```
:  
<dependencies>  
  <dependency>  
    <groupId>junit</groupId>  
    <artifactId>junit</artifactId>  
    <version>3.8.1</version>  
    <scope>test</scope>  
  </dependency>  
</dependencies>  
:
```

Maven Project

❖ JUnit

- Maven 레파지토리 검색(검색어 : junit)



Indexed Artifacts (8.65M)

Popular Categories

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers

Found 1199 results

Sort: **relevance** | popular | newest



1. JUnit

junit » junit

64,977 usages **EPL**

JUnit is a unit testing framework for Java, created by Erich Gamma and Kent Beck.

Last Release on Dec 4, 2014



2. JUnit

junit » junit-dep

1,345 usages **CPL** **CPAL**

JUnit is a regression testing framework written by Erich Gamma and Kent Beck. It is used by the developer who implements unit tests in Java.

Maven Project

❖ JUnit



JUnit

JUnit is a unit testing framework for Java, created by Erich Gamma and Kent Beck.

License	EPL 1.0
Categories	Testing Frameworks
Tags	testing junit
Used By	64,977 artifacts

Central (24) Redhat GA (3) JBoss 3rd-party (1) Alfresco Public (1)				
Version		Repository	Usages	Date
4.12.x	4.12	Central	26,052	(Dec, 2014)
	4.12-beta-3	Central	28	(Nov, 2014)
	4.12-beta-2	Central	31	(Sep, 2014)

Maven Project

License	EPL 1.0
Categories	Testing Frameworks
Organization	JUnit
HomePage	http://junit.org
Date	(Dec 04, 2014)
Files	pom (23 KB) jar (307 KB) View All
Repositories	Central Aspose Redhat GA Sonatype Releases
Used By	64,977 artifacts

Maven Gradle SBT Ivy Grape Leiningen Buildr

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.12</version>
  <scope>test</scope>
</dependency>
```

클릭하면 자동 복사됨

Maven Project

❖ JUnit 설정

- 디폴트 버전 : 3.8.1 → 4.12로 변경
- pom.xml

```
:  
<dependencies>  
  <dependency>  
    <groupId>junit</groupId>  
    <artifactId>junit</artifactId>  
    <version>4.12</version>  
    <scope>test</scope>  
  </dependency>  
</dependencies>  
:
```

❖ 용어

- 테스트 타겟 클래스(Test Target Class)
 - 테스트 대상 클래스
- 테스트 케이스(Test Case)
 - 테스트를 정의한 클래스
- 테스트 러너(Test Runner)
 - 테스트 실행기
- 테스트 스위트(Test Suite, Suite)
 - 테스트 집합
 - 여러 개의 테스트 클래스를 동시에 실행하는 특수한 테스트 러너
 - 테스트 클래스와 동일한 방식으로 동작

JUnit

❖ 테스트 타겟 클래스

- Calculator (src/main/java에 생성)

```
package edu.iot.junittesttest;

public class Calculator {
    public double add(double number1, double number2) {
        return number1 + number2;
    }
}
```


JUnit

❖ JUnit 핵심 객체

JUnit 개념	역할
Assert	테스트하려는 조건을 명시한다. assert 메서드는 조건이 만족되면 아무 일도 없었다는 듯이 조용히 지나가며, 만족되지 못하면 예외를 던진다.
Test	@Test 애노테이션이 부여된 메서드로, 하나의 테스트를 뜻한다. JUnit은 먼저 메서드를 포함하는 클래스의 인스턴스를 만들고, 애노테이션된 메서드를 찾아 호출한다.
Test 클래스	@Test 메서드를 포함한 클래스이다.
Suite	스위트는 여러 테스트 클래스를 하나로 묶는 수단을 제공한다.
Runner	러너는 테스트를 실행시킨다. JUnit 4는 하위 호환성을 유지하여, JUnit 3의 테스트도 역시 실행할 수 있다

JUnit

❖ JUnit Assertions

- `import static org.junit.Assert.*`
 - 메서드를 바로 사용

코드	설명
<code>assertEquals([message], expected, actual)</code>	두 값이 같은 지 비교
<code>assertSame([message], expected, actual)</code> <code>assertNotSame([message], expected, actual)</code>	두 객체가 동일한 객체인지 비교
<code>assertTrue([message], expected)</code> <code>assertFalse([message], expected)</code>	참/거짓 판별
<code>assertNull([message], expected)</code> <code>assertNotNull([message], expected)</code>	null여부 판단
<code>fail([message])</code>	테스트 실패로 판단

❖ 테스트 클래스(테스트 케이스) 작성

- 테스트 클래스명
 - 대상 클래스 명 + Test
 - CalculatorTest
- 테스트 메서드
 - @Test 어노테이션 설정
 - 반드시 public이고, 매개변수가 없어야 함
 - 반환형은 void

❖ Junit4 애노테이션

- `@BeforeClass`
 - 테스트 클래스 내에서 수행 전 한 번만 실행, `static method` 여야 함
- `@AfterClass` :
 - 테스트 클래스 내에서 수행 후 한 번만 실행
 - `static method` 여야 함
- `@Before`
 - 테스트 케이스 수행 전 반복 실행
- `@After`
 - 테스트 케이스 수행 후 반복 실행
- `@Test`
 - 테스트 메소드 지정, 메서드명은 `test`로 시작
 - 예외 테스트
 - `@Test(expected=NumberFormatException.class)`
 - 시간 제한 테스트
 - `@Test(timeout=2000)`
- `@Ignore("")`
 - 테스트 무시

JUnit

❖ @RunWith(클래스 이름.class)

- JUnit Test 클래스를 실행하기 위한 러너(Runner)를 명시적으로 지정한다.
- @RunWith는 junit.runner.Runner를 구현한 외부 클래스를 인자로 갖는다.

❖ @SuiteClasses(Class[])

- 보통 여러 개의 테스트 클래스를 수행하기 위해 쓰인다.
- @RunWith를 이용해 Suite.class를 러너로 사용한다.

❖ 파라미터를 이용한 테스트

@RunWith(Parameterized.class)

@Parameters

```
public static Collection data() {  
}
```

JUnit

❖ 테스트 케이스 정의

- CalculatorTest (src/test/java에 정의)

```
import static org.junit.Assert.assertEquals;
import org.junit.Test;

public class CalculatorTest {

    @Test
    public void testAdd() {
        Calculator calculator = new Calculator();
        double result = calculator.add(10, 50);

        assertEquals(60, result, 0); // 인자 : 예상값, 실제값, 허용 오차
    }
}
```

JUnit

❖ 테스트 실행

- 테스트 클래스 선택
 - Run As > JUnit Test

○ 테스트 성공의 경우

The screenshot shows the Eclipse IDE interface with the JUnit runner selected. The top toolbar includes icons for Problems, Console, Type Hierarchy, and JUnit. Below the toolbar, it states "Finished after 0.028 seconds". A progress bar at the bottom indicates "Runs: 1/1", "Errors: 0", and "Failures: 0", with a green bar representing 100% success. The test runner list shows "edu.iot.junit.JUnitTest.CalculatorTest [Runner: JUnit 4] (0.000 s)". To the right, a "Failure Trace" tab is visible.

○ 테스트 실패의 경우

The screenshot shows the Eclipse IDE interface with the JUnit runner selected. The top toolbar is the same as the previous screenshot. Below the toolbar, it states "Finished after 0.034 seconds". A progress bar at the bottom indicates "Runs: 1/1", "Errors: 0", and "Failures: 1", with a red bar representing the failure. The test runner list shows "edu.iot.junit.JUnitTest.CalculatorTest [Runner: JUnit 4] (0.000 s)" with a sub-entry "testAdd (0.000 s)". To the right, the "Failure Trace" tab displays the error message: "java.lang.AssertionError: expected:<60.0> but was:<61.0>" and the location "at edu.iot.junit.JUnitTest.CalculatorTest.testAdd(CalculatorTest.java:12)".

JUnit

❖ JUnit 테스트 동작 원리

- @Test 메서드를 실행할 때마다 테스트 클래스의 인스턴스를 새로 생성
 - 테스트 메서드를 독립된 메모리 공간에서 실행
 - 인스턴스 변수를 공유하지 않음
 - 혹시 모를 의도하지 않은 부작용 방지
- 테스트 검증
 - Assert 클래스에 정의된 단정 메서드 assert()를 사용
 - 정적 임포트를 통해 단순화

```
import static org.junit.Assert.assertEquals;
:
assertEquals(60, result, 0); // 인자 : 예상값, 실제값, 허용 오차
```


JUnit

❖ 테스트 러너(TestRunner)

러너	용도
org.junit.internal.runners. JUnit38ClassRunner	하위 호환 목적으로 제공되는 러너로, JUnit 3.8 테스트 케이스 전용이다.
org.junit.runners.JUnit4	JUnit 4 스타일의 테스트 케이스를 실행한다.
org.junit.runners.Parameterized	같은 테스트 케이스를 다른 입력 값을 사용해 반복 수행한다.
org.junit.runners.Suite	복수의 테스트를 묶을 수 있는 집합이다. 테스트 클래스 내의 모든 @Test 메서드를 찾아 실행하는 러너이기도 하다.

- @RunWith 어노테이션으로 테스트러너 지정
 - 디폴트는 JUnit4

❖ 파라미터화 테스트 실행

- 파라미터화 테스트 러너
 - 하나의 테스트를 여러 번 반복하여 실행하는 기능 제공
 - 테스트 데이터들을 파라미터화하여 매번 바뀌가며 테스트 실행
 - `@RunWith` 어노테이션으로 테스트 러너를 `Parameterized.class`로 지정
 - 파라미터 컬렉션을 리턴하는 `static` 메서드 준비
 - 파라미터를 인스턴스 변수로 설정하는 생성자 준비
 - 테스트 메서드 작성

JUnit

❖ 파라미터화 테스트 실행

```
import static org.junit.Assert.assertEquals;
:
import org.junit.Test;
import org.junit.runner.RunWith;
import org.junit.runners.Parameterized;
import org.junit.runners.Parameterized.Parameters;

@RunWith(Parameterized.class)
public class ParameterizedTest {

    @Parameters
    public static Collection<Integer[]> getTestParameters() {
        return Arrays.asList(new Integer[][] {
            // 기대값, 값1, 값2
            {2, 1, 1},
            {3, 2, 1},
            {4, 3, 1},
        });
    }
}
```

❖ 파라미터화 테스트 실행

```
private double expected;  
private double number1;  
private double number2;
```

```
public ParameterizedTest(double expected,  
                           double number1, double number2) {  
    this.expected = expected;  
    this.number1 = number1;  
    this.number2 = number2;  
}
```

@Test

```
public void testAdd() {  
    Calculator calculator = new Calculator();  
    assertEquals(expected, calculator.add(number1, number2), 0);  
}  
}
```

JUnit

❖ 파라미터화 테스트 실행

- 실제 실행한 테스트

```
sum: assertEquals(2, calculator.add(1, 1), 0);  
sum: assertEquals(3, calculator.add(2, 1), 0);  
sum: assertEquals(4, calculator.add(3, 1), 0);
```

❖ Java assert

- assert 조건식 : '예외 문자열';
 - 조건식이 true 이면 통과
 - false 이면 예외 발생, 예외 문자열 설정됨

```
assert member != null : 'member는 null이 되면 안됨';
```

JUnit

❖ 스위트 테스트 러너

- 테스트 집합(여러 개의 테스트 클래스(테스트 케이스)로 구성)을 테스트
- @SuiteClasses로 테스트 케이스 클래스를 배열로 지정

```
@RunWith(Suite.class)
@SuiteClasses({
    TestCaseClass1.class,
    TestCaseClass2.class
})
public class AllTestSuite {

}
```

❖ 스위트의 스위트 만들기

- 여러 개의 테스트 스위트에 대한 마스터 스위트 운영

```
@RunWith(Suite.class)
@SuiteClasses({
    TestSuite1.class,
    TestSuite2.class
})
public class AllTestSuite {

}
```


JUnit

❖ JUnit Test Framework

- Junit4Test

```
import org.junit.*;

public class Junit4Test {

    @BeforeClass
    public static void setUpBeforeClass() throws Exception {
        System.out.println("@BeforeClass");
    }

    @AfterClass
    public static void tearDownAfterClass() throws Exception {
        System.out.println("@AfterClass");
    }
}
```

JUnit

@Before

```
public void setUp() throws Exception {  
    System.out.println("@Before");  
}
```

@After

```
public void tearDown() throws Exception {  
    System.out.println("@After");  
}
```

@Test

```
public void testCase1() throws Exception {  
    System.out.println("testCase1");  
}
```

@Test

```
public void testCase2() throws Exception {  
    System.out.println("testCase2");  
}
```

```
}
```

JUnit

❖ JUnit Test Framework

○ Junit4Exam

```
import org.junit.Ignore;
import org.junit.Test;
import static org.junit.Assert.*;

public class Junit4Exam {

    // 예외 테스트
    @Test(expected = NumberFormatException.class)
    public void testException() throws Exception {
        String str = "hello";
        System.out.println("문자열이 정수여야 함",
                           Integer.parseInt(str));
    }
}
```

JUnit

❖ JUnit Test Framework

- Junit4Exam

```
// 테스트 시간 제한
@Test(timeout = 1000)
public void testTimeout() throws Exception {
    long sum = 0;
    for (int i = 0; i < 10000; i++) {
        for (int j = 0; j < 10000; j++) {
            sum += j;
        }
    }
    System.out.println(sum);
}
```

JUnit

❖ JUnit Test Framework

○ Junit4Exam

```
// 테스트 무시
@Ignore
@Test
public void testIgnore() throws Exception {
    assertTrue("항상 실패", false);
}

// 배열 지원 - 값이랑 순서까지 동일해야 함
@Test
public void testAssertArrayEquals() throws Exception {
    Object[] a = {"Java", "Python", 1};
    Object[] b = {"Java", "Python", 1};
    assertEquals("두 배열이 같아야 함", a, b);
}
}
```