올인원
스프링 프레임워크

Chapter 03
스프링 맛보기

목차

학습목표

- 메이븐이 무엇인지 살펴봅니다.
- 메이븐을 이용하여 스프링 프로젝트를 생성합니다.
- pom.xml을 이해합니다.
- 스프링 DI와 IoC를 이용해서 프로그램을 만들어봅니다.

Section 01

메이븐 프로젝트

1. 컴파일과 빌드의 차이점

■ 컴파일(compile)

- 코딩한 코드 파일을 컴파일러(compiler)가 바이트코드(bytecode) 파일로 변환하는 과정을 뜻함
- 바이트코드 파일은 JVM에 의해 기계어로 바뀌어 컴퓨터에서 실행됨

■ 빌드(build)

- 컴파일보다 넓은 의미로 라이브러리 다운로드 및 연결, 컴파일, 링크, 패키징 등 애플리케이션 제작에 필요한 전반적인 과정을 뜻함
  - ✓ 링크: 서로 다른 파일을 연결 해서 메서드 호출 등의 업무가 가능하게 만드는 것
  - ✓ 패키징: 구현된 각 각의 기능을 하나로 합쳐서 실행 파일을 만드는 것
  - ✓ 빌드는 많은 과정을 담당하는데 이러한 과정은 개발자가 직접 하는 것이 아니고, 빌드 툴에 의해 모두 자동화되어 있음

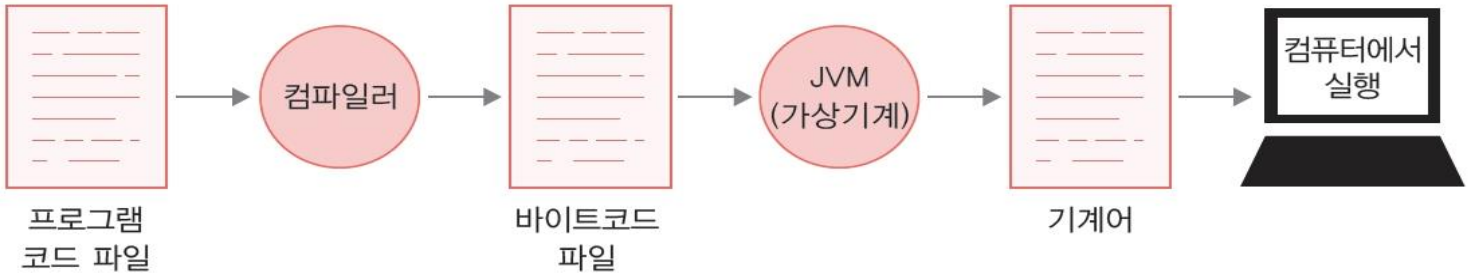


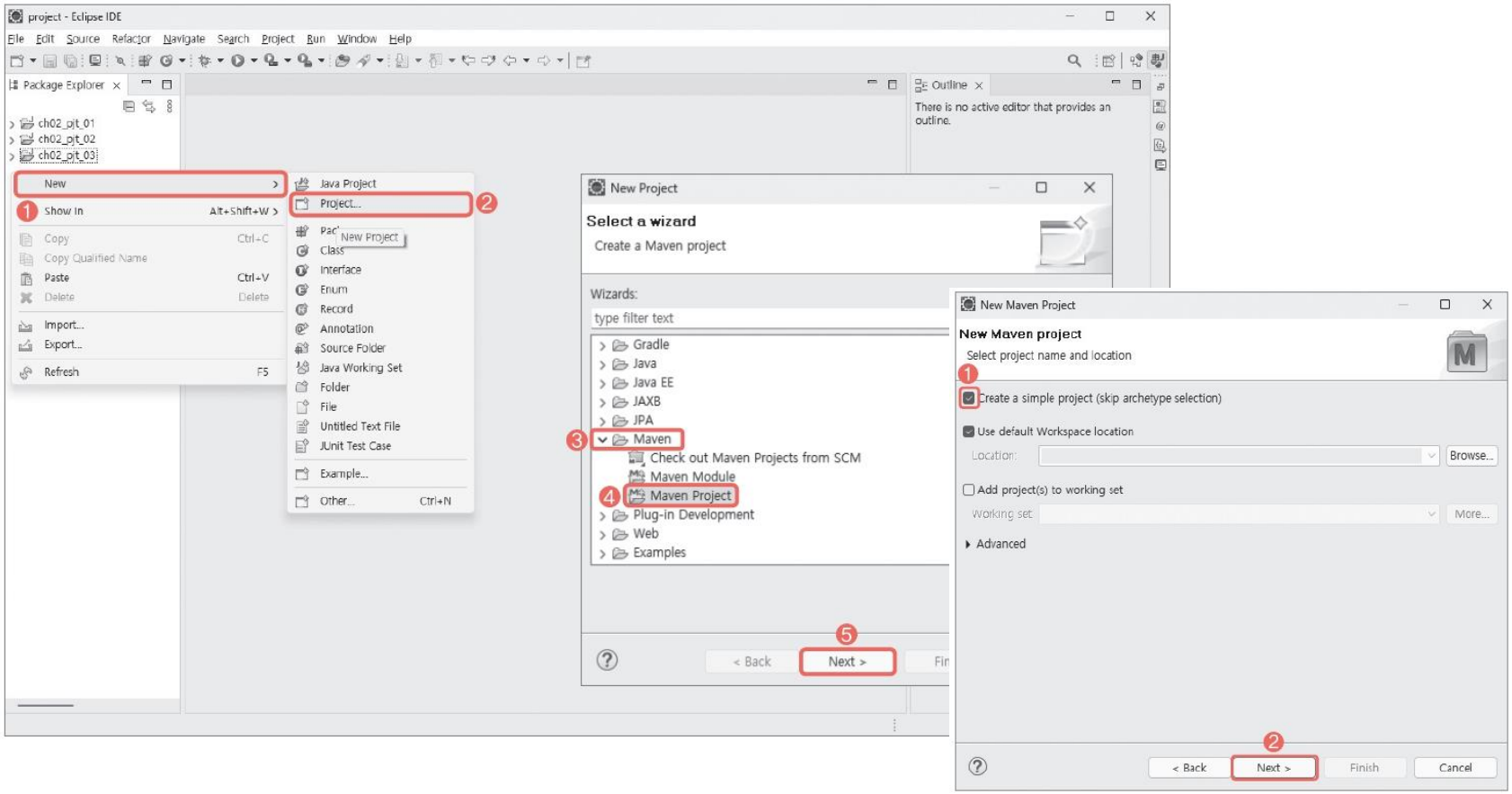
그림 3-1 프로그램 코드 파일은 컴파일러를 통해 바이트코드 파일로 변환된다.

빌드툴의 종류

- Ant: 과거에 많이 사용했지만, 내부 스크립트가 다소 복잡해서 점차 사용 빈도가 줄고 있음
- Maven: 오래 전부터 스프링 애플리케이션 개발에 사용된 툴로 많은 산업 현장에서 여전히 사용되고 있음
  - ✓ 앞으로 진행할 스프링 프로젝트는 메이븐(Maven)을 사용함
  - ✓ 메이븐은 이클립스에 기본적으로 설치가 되어 있어 별도의 설치 과정이 필요 없음
- Gradle: 최근에 인기를 얻고 있음
  - ✓ 메이븐보다 약 두 배가량 빠름

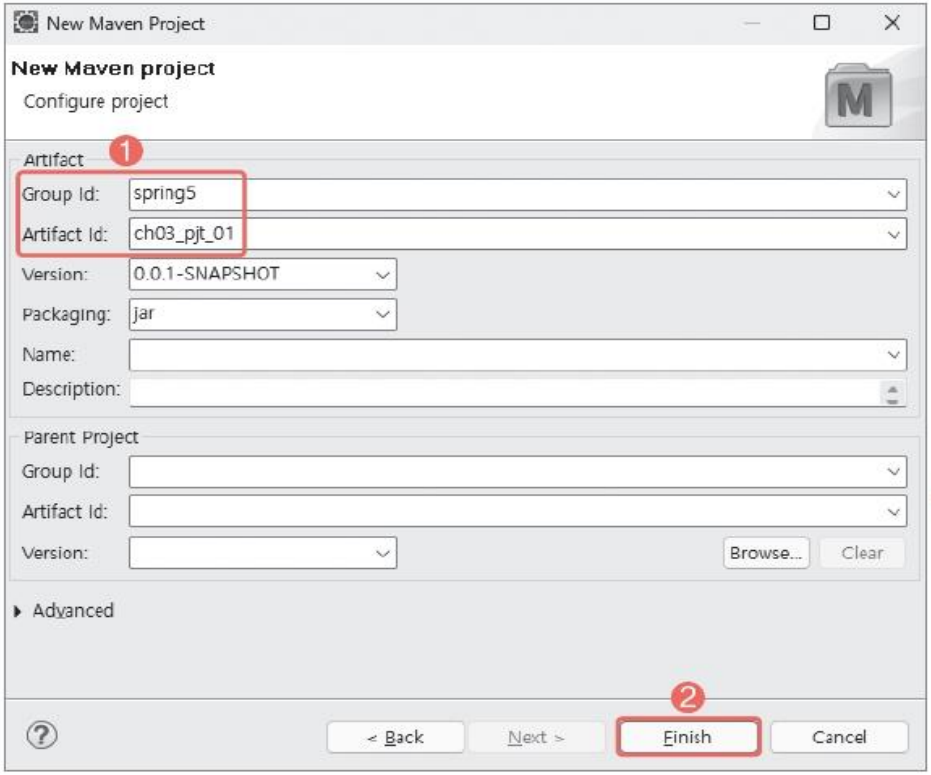
2. 메이븐 프로젝트 생성하기

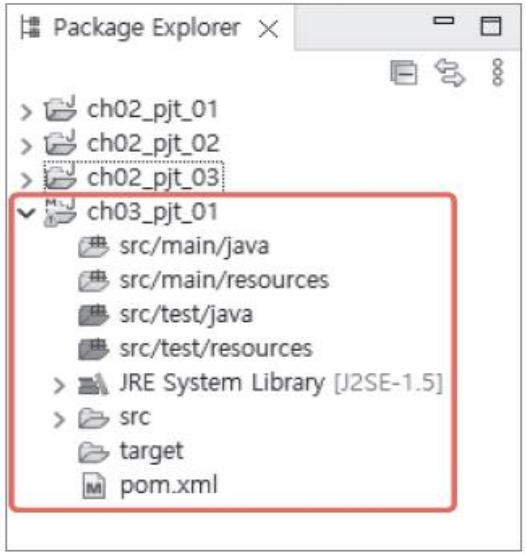
- 1. Package Explorer에서 마우스 오른쪽 버튼을 클릭하여 [New]-[Project...]를 선택하기
  - [New Project] 창에서 Maven 폴더의 Maven Project를 선택한 후 <Next> 클릭
- 2. [New Maven Project] 창이 나오면 'Create a simple project(skip archetype selection)'에 체크하고 <Next> 클릭



3. Artifact 정보를 입력하는 창에 Group Id, Artifact Id를 입력하고 <Finish> 버튼을 클릭

- Group Id: spring5
- Artifact Id: ch03_pjt_01





■ 프로젝트의 시나리오

① MainClass에서 이동수단(TransportationWalk) 객체를 생성한다.

② 생성된 이동수단(TransportationWalk) 객체의 move() 메서드를 호출한다.

③ move() 메서드 호출 시 콘솔 창에 해당하는 이동수단(도보)을 출력한다.

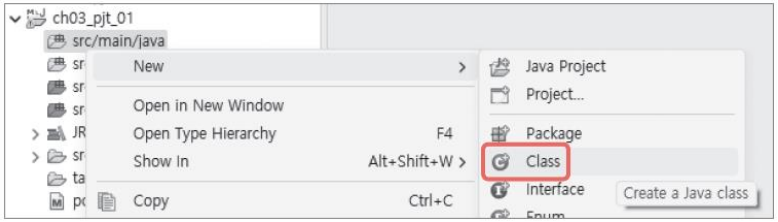
■ 프로젝트의 구조

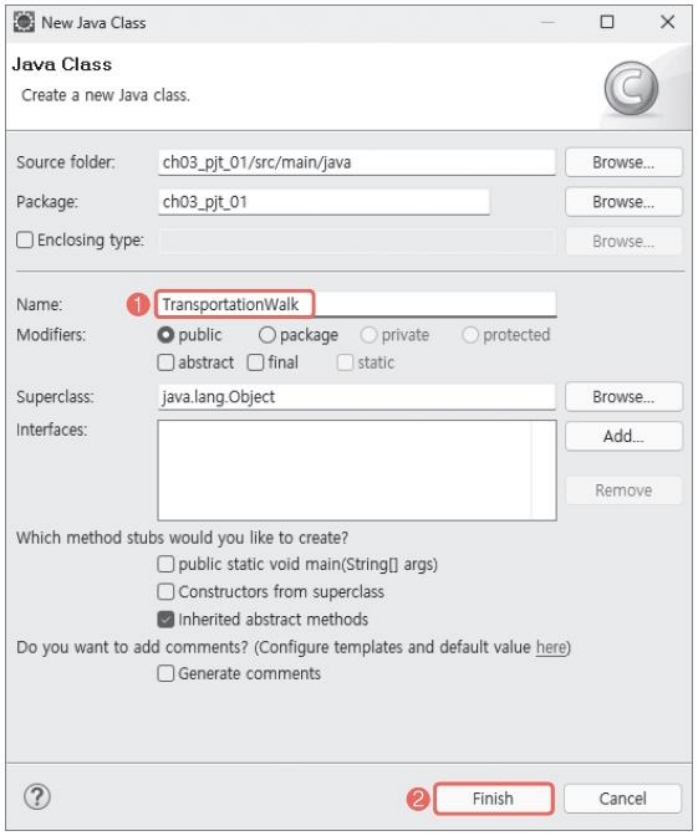
- MainClass 클래스는 TransportationWalk 클래스를 사용함

그림 3-2 ch03_pjt_01 프로젝트의 구조

■ 클래스 생성과 코딩

- 1. ch03_pjt_01의 [src/main/java]에서 마우스 오른쪽 버튼을 클릭하여 새로운 Class 파일을 생성하기
- 2. 클래스 이름을 TransportationWalk로 하고 <Finish> 클릭





■ 클래스 생성과 코딩

- 3. TransportationWalk.java 파일을 다음과 같이 작성함

코드 3-1 ch03_pjt_01\src\main\java\ch03_pjt_01\TransportationWalk.java

```
01 package ch03_pjt_01;
02 public class TransportationWalk {
03     public void move() {
04         System.out.println("도보로 이동합니다!");
05     }
06 }
```

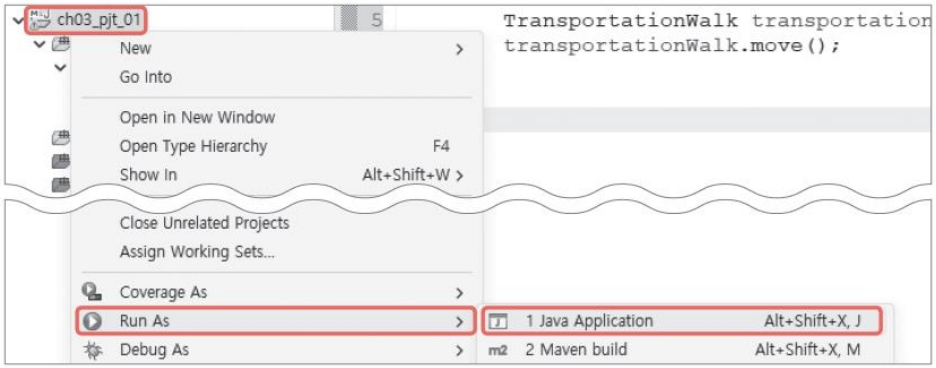
- 4. MainClass 클래스를 생성하고 다음과 같이 작성함

코드 3-2 ch03_pjt_01\src\main\java\ch03_pjt_01\MainClass.java

```
01 package ch03_pjt_01;
02 public class MainClass {
03     public static void main(String[] args) {
04         TransportationWalk transportationWalk = new TransportationWalk();
05         transportationWalk.move();
06     }
07 }
```

■ 클래스 생성과 코딩

- 5. 작성을 모두 완료했으면 ch03_pjt_01에서 마우스 오른쪽 버튼을 클릭하고 [Run As]-[Java Application]을 선택하여 프로그램을 실행하기



- 6. main()에서 TransportationWalk의 move()를 호출한 결과 확인하기

실행 결과

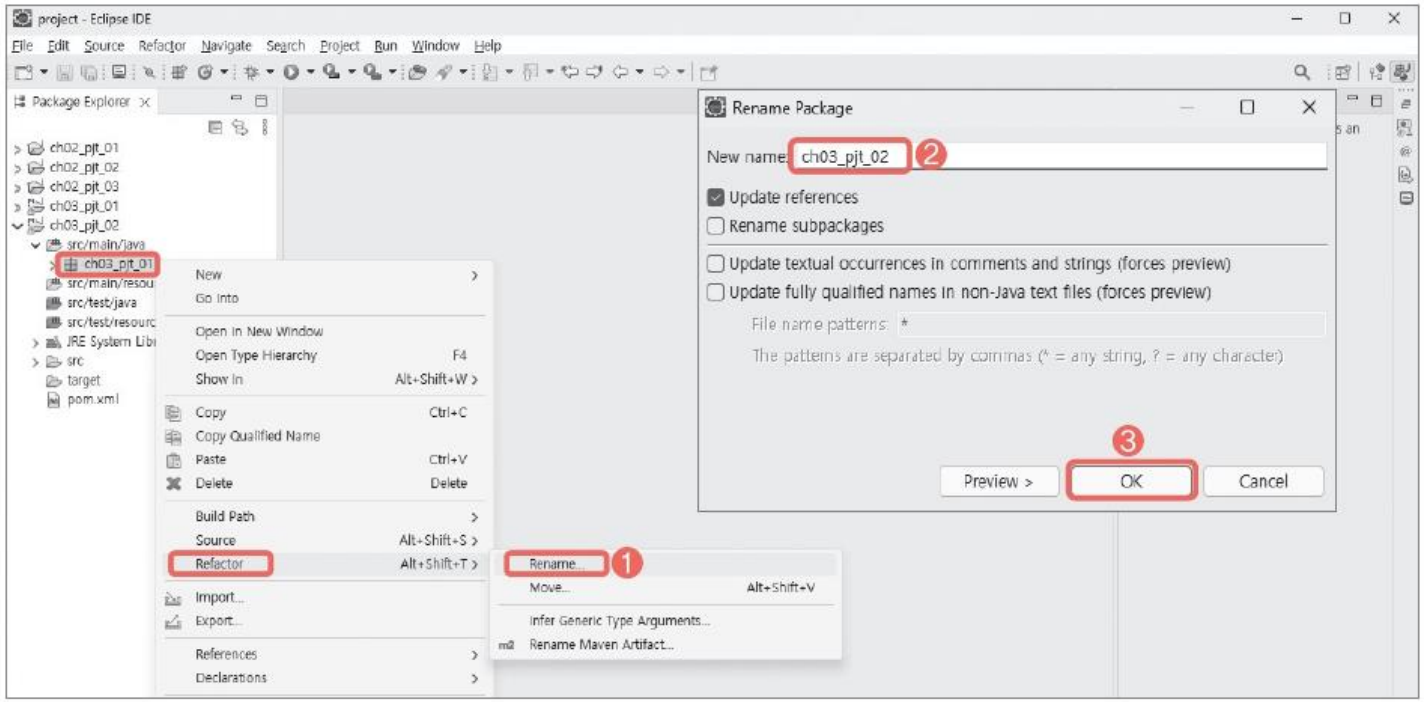
```
도보로 이동합니다!
```

Section 02

처음 만들어보는 스프링 프로젝트

1. 프로젝트 만들기

- 1. ch03_pjt_01을 복사해서 붙인 후, [Copy Project] 창이 열리면 Project name에 ch03_pjt_02를 작성한 후 <Copy>를 클릭하기
- 2. ch03_pjt_02의 [src/main/java]에서 ch03_pjt_01 패키지명에 마우스 오른쪽 버튼을 클릭하여 [Refactor]-[Rename]을 선택하면 [Rename Package] 창이 열림.
  - New name에 ch03_pjt_02를 입력한 후 <OK>를 클릭



2. pom.xml 작업하기

Ch03_pjt_02가 ch03_pjt_01과 다른 점

- GenericXmlApplicationContext를 이용해서 main()에 TransportationWalk를 직접 생성하지 않고 스프링 IoC 컨테이너에서 Bean으로 생성하여 사용함
- 이를 위해 메이븐을 이용해서 spring-context-5.2.9.RELEASE.jar를 가져와야 함

pom.xml 코딩

코드 3-3 ch03_pjt_02\pom.xml

```
01 <project xmlns="http://maven.apache.org/POM/4.0.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
   https://maven.apache.org/xsd/maven-4.0.0.xsd">
02     <modelVersion>4.0.0</modelVersion>
03     <groupId>spring5</groupId>
04     <artifactId>ch03_pjt_02</artifactId>
05     <version>0.0.1-SNAPSHOT</version>
06
07     <!-- spring-context 모듈 -->
08     <dependencies>
09         <dependency>
10             <groupId>org.springframework</groupId>
11             <artifactId>spring-context</artifactId>
12             <version>5.2.9.RELEASE</version>
13         </dependency>
14     </dependencies>
15
```

2. pom.xml 작업하기

```
16      <!-- 빌드 설정 -->
17      <build>
18          <plugins>
19              <plugin>
20                  <artifactId>maven-compiler-plugin</artifactId>
21                  <version>3.1</version>
22                  <configuration>
23                      <source>11</source>
24                      <target>11</target>
25                      <encoding>utf-8</encoding>
26                  </configuration>
27              </plugin>
28          </plugins>
29      </build>
30
31  </project>
```

■ JRE 라이브러리 버전을 11로 변경하기

- ch03_pjt_02 프로젝트에서 사용되는 JRE 라이브러리 버전이 1.5로 설정되어 있음([J2SE-1.5])
- 프로젝트 이름 위에서 마우스 오른쪽 버튼을 클릭하여 [Maven]-[Update Project...(<Alt>+<F5>)]를 선택

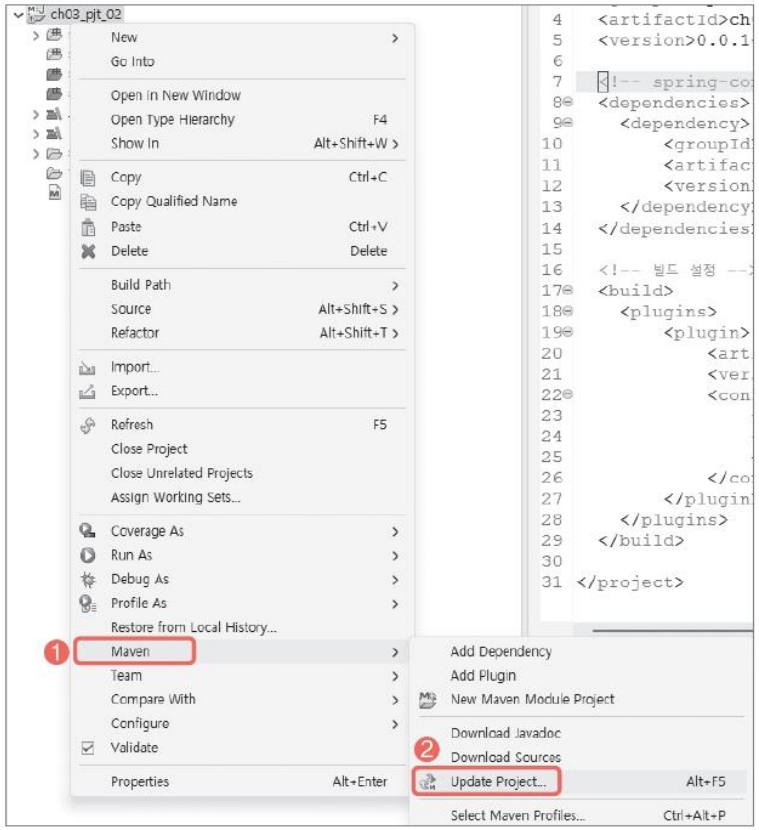
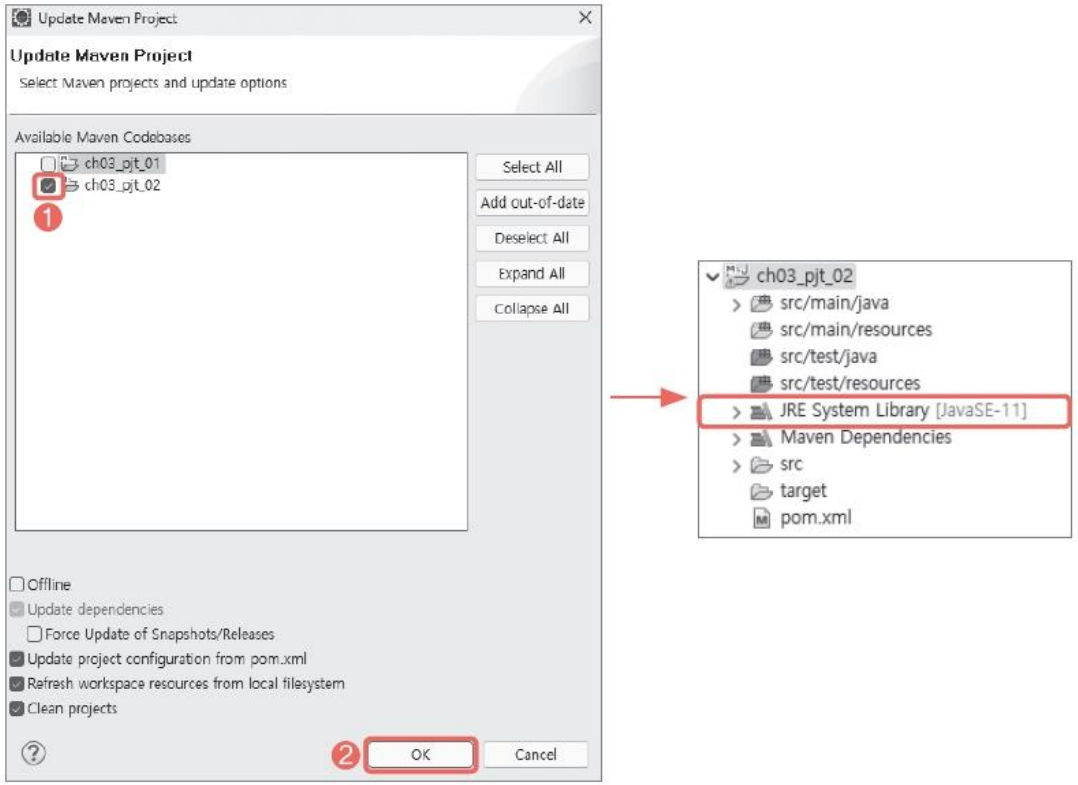


그림 3-3 메이븐 프로젝트 업데이트

3. 프로젝트 업데이트하기

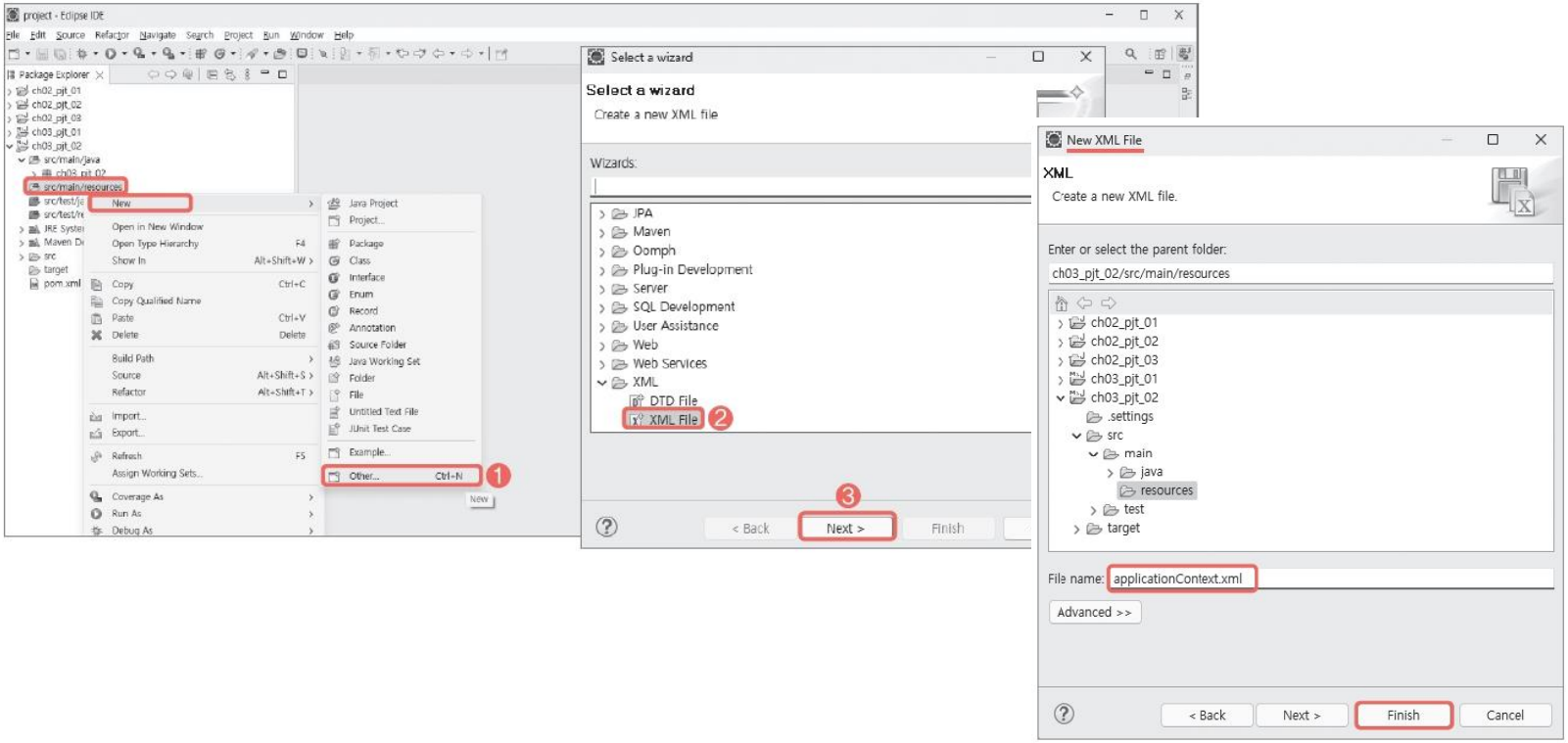
■ JRE 라이브러리 버전을 11로 변경하기

- [Update Maven Project] 창에서 프로젝트를 선택하고 <OK>를 클릭해서 JRE11 버전 ([JavaSE-11])으로 변경



IoC 컨테이너 역할을 하는 스프링 설정 파일: applicationContext.xml

- 1. [src/main/resources]에서 마우스 오른쪽 버튼을 클릭하여 [New]-[Other...]를 선택(<Ctrl>+<N>)하기. 이어서 [Select a wizard] 창에서 스크롤을 아래쪽으로 내려 [XML]-[XML File]을 선택한 후 <Next> 클릭
- 2. [New XML File] 창에서 File name을 applicationContext.xml로 하고 <Finish> 클릭



4. applicationContext.xml 생성하기

■ IoC 컨테이너 역할을 하는 스프링 설정 파일: applicationContext.xml

- 3. applicationContext.xml을 코딩하기

코드 3-4 ch03_pjt_02\src\main\resources\applicationContext.xml

```
01 <?xml version="1.0" encoding="UTF-8"?>
02
03 <beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd">
04
05   <bean id="tWalk" class="ch03_pjt_02.TransportationWalk" />
06
07 </beans>
```

4. applicationContext.xml 생성하기

■ IoC 컨테이너 역할을 하는 스프링 설정 파일: applicationContext.xml

- 4. [src/main/java]의 MainClass.java를 수정하기

코드 3-5 ch03_pjt_02\src\main\java\MainClass.java

```
01 package ch03_pjt_02;
02
03 import org.springframework.context.support.GenericXmlApplicationContext;
04
05 public class MainClass {
06     public static void main(String[] args) {
07
08 //  TransportationWalk transportationWalk = new TransportationWalk();
09 //  transportationWalk.move();
10
11          GenericXmlApplicationContext ctx =
                new GenericXmlApplicationContext("classpath:applicationContext.xml");
12
13          TransportationWalk transportationWalk =
                ctx.getBean("tWalk", TransportationWalk.class);
14          transportationWalk.move();
15
16          ctx.close();
17      }
18 }
```

실행 결과

```
도보로 이동합니다!
```

Section 03

디렉터리와
pom.xml 파일의 이해

■ 디렉터리 구성

- java(ch03_pjt_02/src/main/java) 디렉터리
  - ✓ 앞으로 만들어지는 자바 파일들이 관리되는 곳임
  - ✓ 즉 프로젝트에 필요한 자바 파일을 이곳에 만들면 됨
- resources(ch03_pjt_02/src/main/resources) 디렉터리
  - ✓ 자원을 관리하는 폴더로 스프링 설정 파일(XML) 또는 프로퍼티 파일 등이 관리됨

표 3-1 디렉터리 구성과 역할

디렉터리	역할
ch03_pjt_02	프로젝트 root
ch03_pjt_02/src/main/java	.java 파일 관리
ch03_pjt_02/src/main/resources	자원 관리

메이븐 설정 파일: pom.xml

- 필요한 라이브러리를 연결해주고 빌드 설정을 담당함
  - ✓ 과거에는 개발자가 스프링에 필요한 라이브러리를 프로젝트에 직접 연결해서 사용하기도 했음
- 빌드에 필요한 정보도 가지고 있음
- **다시 말해 pom.xml은 스프링 프로젝트에 필요한 라이브러리와 빌드에 필요한 정보가 적혀 있는 명세서**

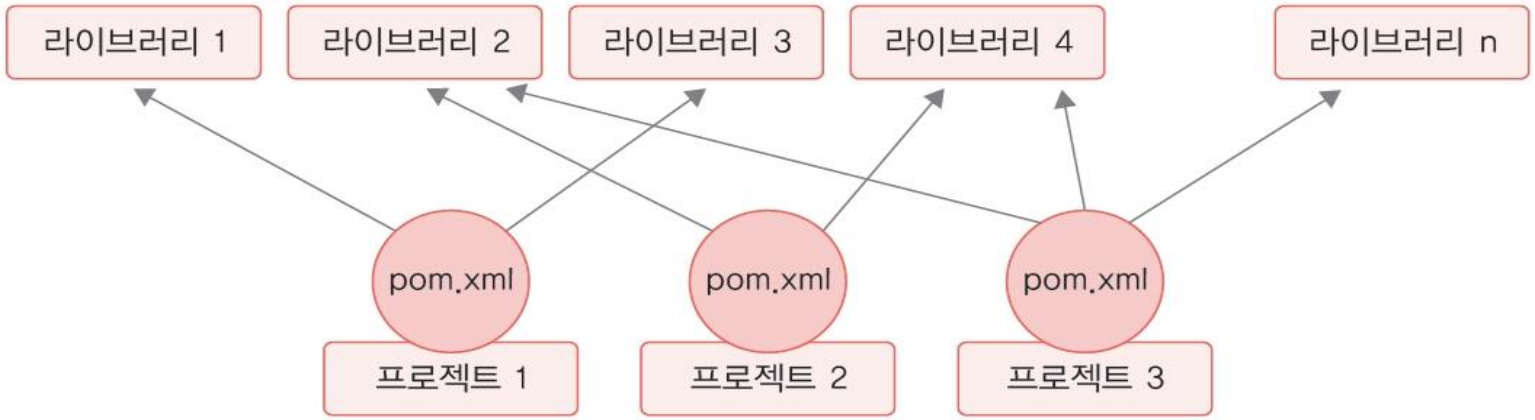


그림 3-5 pom.xml에 다운로드가 필요한 모듈을 설정

■ 메인 리포지터리

- 아파치 재단 에서 관리하는 메일 리포지터리에서 모듈을 다운로드함

✓ 메일 리포지터리 주소: https://mvnrepository.com/

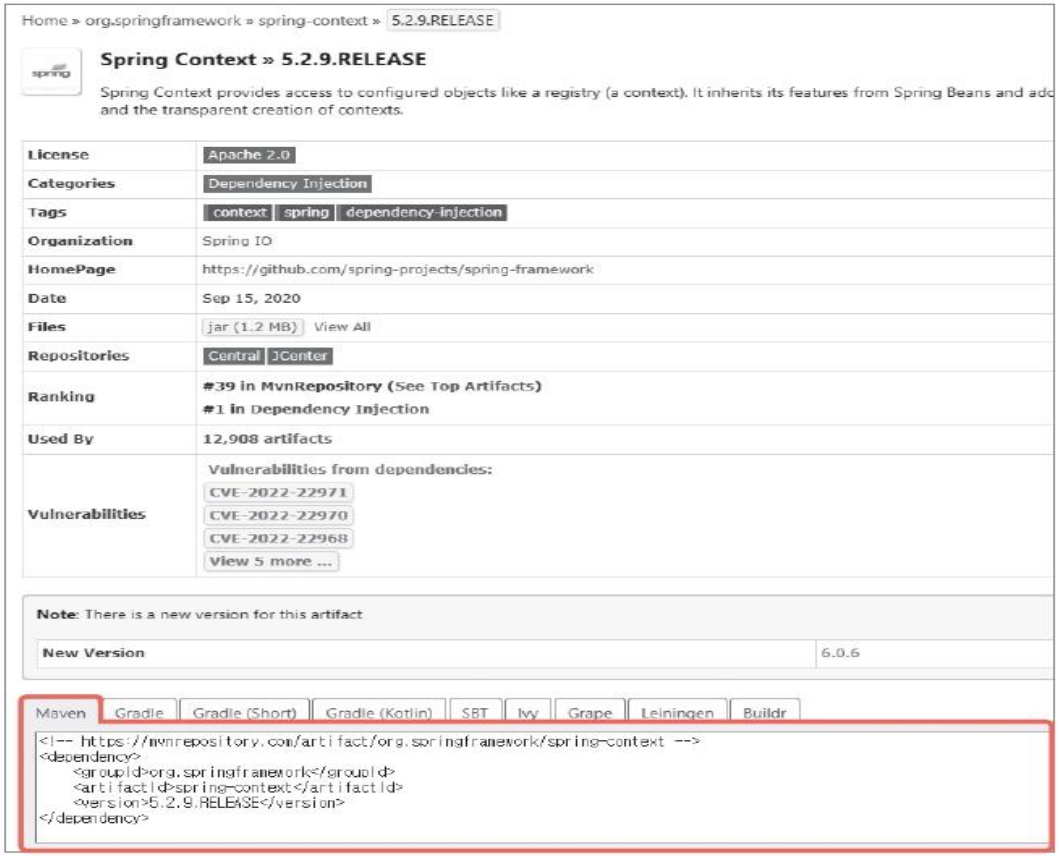


그림 3-6 메인 리포지터리의 spring-context 모듈

로컬 리포지터리

- 다운로드한 라이브러리(.jar)가 저장된 개발자 컴퓨터
- spring-context 모듈 설정
  - ✓ 모듈 하나를 artifact(아티팩트)라는 단위로 관리함
  - ✓ org.spirngframework 그룹에 있는 spring-context라는 모듈의 5.2.9.RELEASE라는 버전을 프로젝트에서 사용(의존)하는 코드

```
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.2.9.RELEASE</version>
    </dependency>
</dependencies>
```

로컬 리포지터리

- 로컬 리포지터리 경로

```
C:\Users\사용자\.m2\repository\
```

- spring-context-5.2.9.RELEASE.jar 경로

```
C:\Users\homsi\.m2\repository\org\springframework\spring-context\5.2.9.RELEASE\spring-context-5.2.9.RELEASE.jar
```

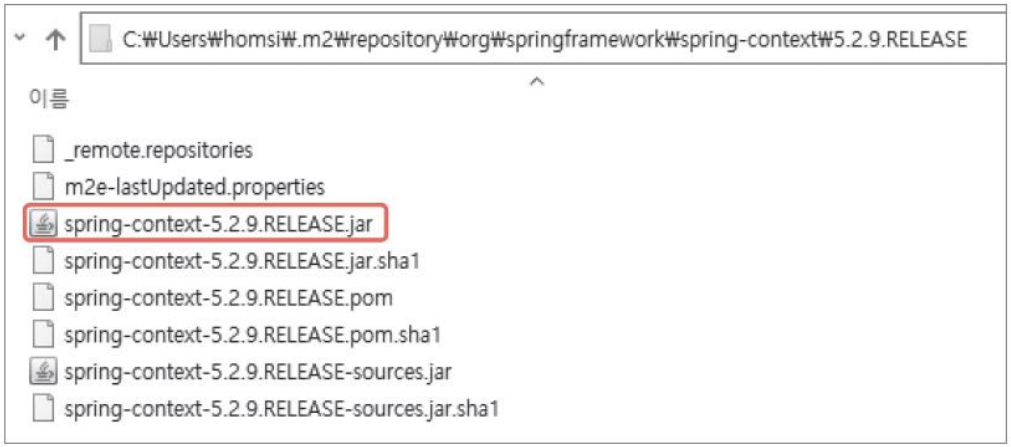


그림 3-9 로컬 리포지터리에서 확인할 수 있는 spring-context 모듈

■ ₩org₩springframework 그룹의 또 다른 모듈

- 개발자는 spring-context-5.2.9.RELEASE.jar가 필요(의존해야 함)해서 다운로드함
- 나머지 .jar 파일들은 spring-context-5.2.9.RELEASE.jar가 의존하고 있기 때문에 같이 다운로드됨
  - ✓ `\spring-aop\5.2.9.RELEASE\spring-aop-5.2.9.RELEASE.jar`
  - ✓ `\spring-beans\5.2.9.RELEASE\spring-beans-5.2.9.RELEASE.jar`
  - ✓ `\spring-core\5.2.9.RELEASE\spring-core-5.2.9.RELEASE.jar`
  - ✓ `\spring-expression\5.2.9.RELEASE/spring-expression-5.2.9.RELEASE.jar`

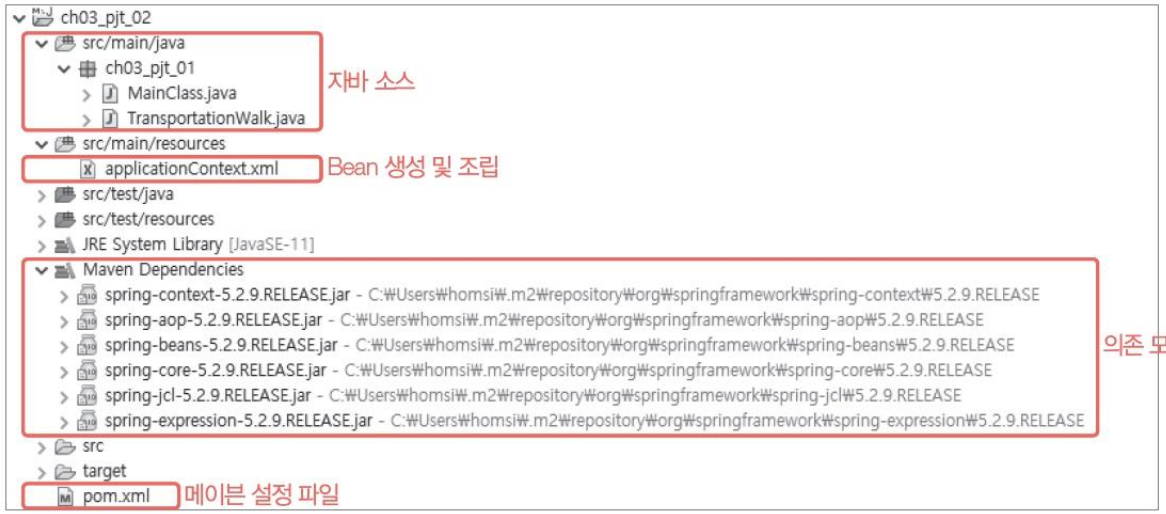


그림 3-10 다운로드한 모듈과 프로젝트 구조

pom.xml 파일에서 프로젝트 빌드를 설정하는 부분의 코드

✓ ([코드 3-3] 17~29행)

- java 파일을 컴파일할 때 JDK11 버전으로 컴파일한다는 것과 자바 소스를 UTF-8로 인코딩한다는 것을 명시한 내용
- 프로젝트 빌드 설정

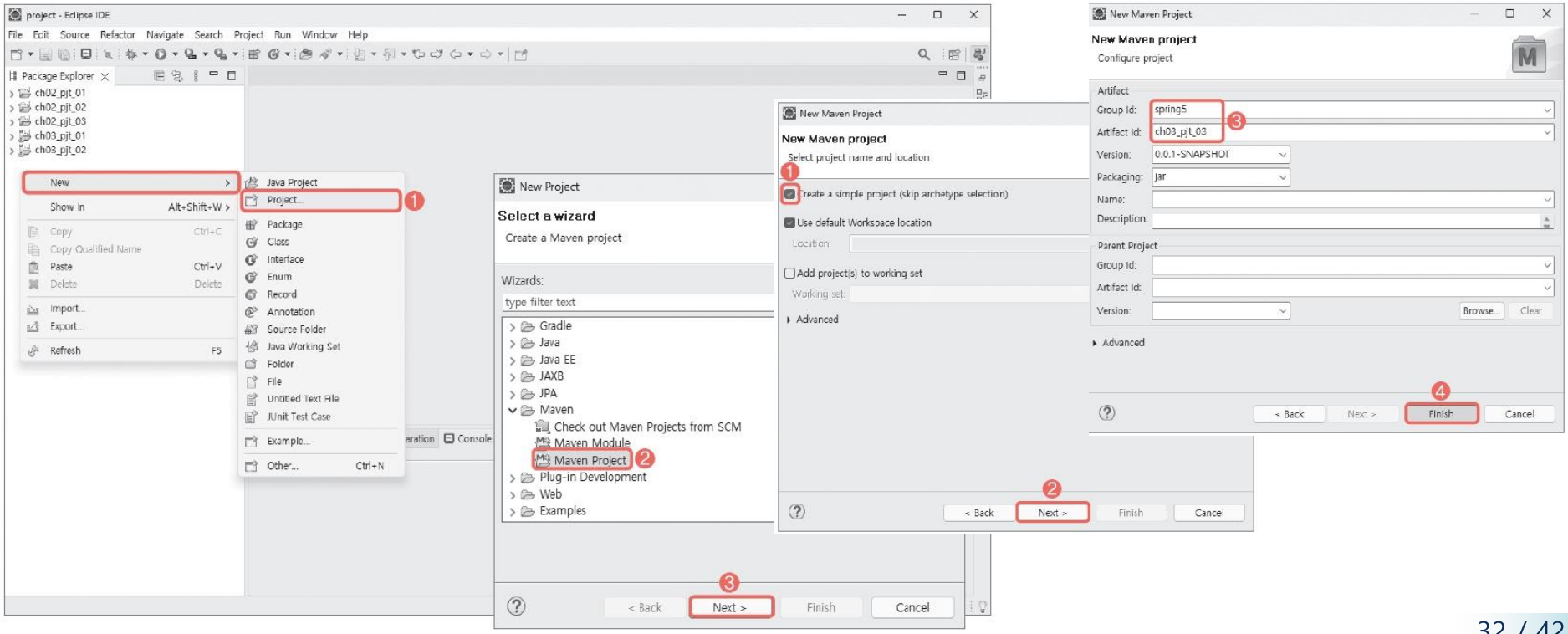
```
<build>
    <plugins>
        <plugin>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.1</version>
            <configuration>
                <source>11</source>
                <target>11</target>
                <encoding>utf-8</encoding>
            </configuration>
        </plugin>
    </plugins>
</build>
```

Section 04

스프링을 이용한 계산기 프로그램

1. 메이븐을 이용한 스프링 계산기 프로그램

- 1. Package Explorer 탭의 흰 바탕에서 마우스 오른쪽 버튼을 클릭하여 [New]-[Project...]를 선택하기. [New Project] 창에서 'Maven'의 'Maven Project'를 선택한 후 <Next> 클릭
- 2. [New Maven Project] 창에서 'Create a simple project(skip archetype selection)'에 체크한 후 <Next> 클릭하기. Artifact 입력하고 <Finish> 클릭
  - Group Id: spring5
  - Artifact Id: ch03_ pjt_03



■ 3. 생성된 ch03_pjt_03 프로젝트의 pom.xml에 스프링과 빌드를 설정하기

코드 3-6 ch03_pjt_03\pom.xml

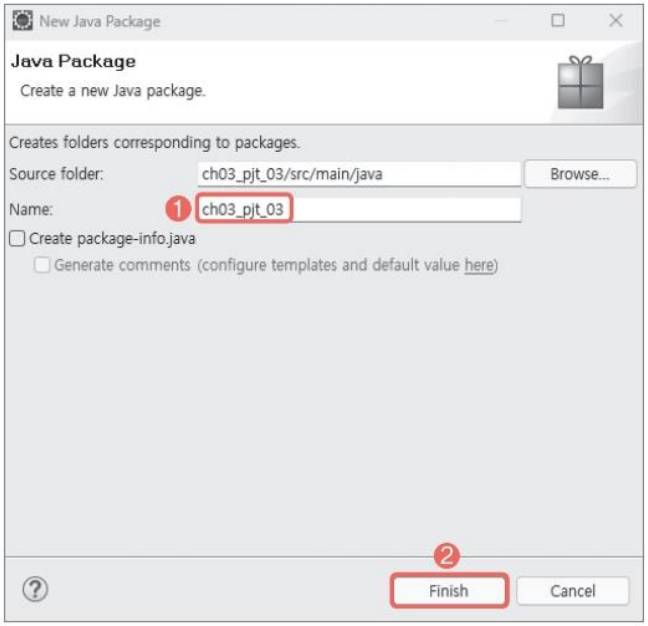
```
01 <project xmlns="http://maven.apache.org/POM/4.0.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
   https://maven.apache.org/xsd/maven-4.0.0.xsd">
02     <modelVersion>4.0.0</modelVersion>
03     <groupId>spring5</groupId>
04     <artifactId>ch03_pjt_02</artifactId>
05     <version>0.0.1-SNAPSHOT</version>
06
07     <!-- spring-context 모듈 -->
08     <dependencies>
09         <dependency>
10             <groupId>org.springframework</groupId>
11             <artifactId>spring-context</artifactId>
12             <version>5.2.9.RELEASE</version>
13         </dependency>
14     </dependencies>
15
16     <!-- 빌드 설정 -->
17     <build>
18         <plugins>
19             <plugin>
20                 <artifactId>maven-compiler-plugin</artifactId>
21                 <version>3.1</version>
```

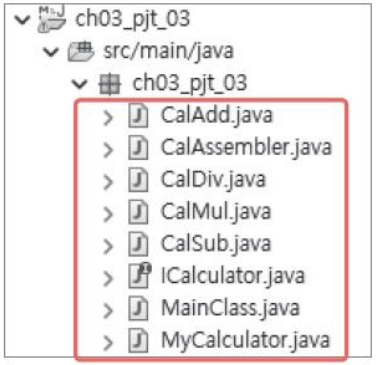
■ 3. 생성된 ch03_pjt_03 프로젝트의 pom.xml에 스프링과 빌드를 설정하기

```
22                  <configuration>
23                      <source>11</source>
24                      <target>11</target>
25                      <encoding>utf-8</encoding>
26                  </configuration>
27              </plugin>
28          </plugins>
29      </build>
30
31  </project>
```

1. 메이븐을 이용한 스프링 계산기 프로그램

- 4. ch03_pjt_03 프로젝트에서 사용되는 JRE 라이브러리 버전이 1.5로 설정([J2SE-1.5])되어 있다면 메이븐 프로젝트를 업데이트하기
- 5. [src/main/java]에 ch03_pjt_03 패키지를 생성하기
- 6. ch03_pjt_03 패키지에 다음과 같이 클래스와 인터페이스 만들고 코딩하기





클래스	인터페이스
MainClass.java	ICalculator.java
MyCalculator.java	
CalAssembler.java	
CalAdd.java	
CalSub.java	
CalMul.java	
CalDiv.java	

(a) 생성 완료된 전체 프로젝트 구조 (b) 생성할 클래스와 인터페이스

그림 3-11 스프링을 활용한 계산기 프로젝트에 필요한 클래스와 인터페이스

■ 클래스와 인터페이스 코딩하기

코드 3-7 ch03_pjt_03\src\main\java\ch03_pjt_03\MainCalss.java

```
01 package ch03_pjt_03;
02 import org.springframework.context.support.GenericXmlApplicationContext;
03
04 public class MainClass {
05     public static void main(String[] args) {
06
07         GenericXmlApplicationContext ctx =
             new GenericXmlApplicationContext("classpath:applicationContext.xml");
08
09         CalAssembler calAssembler =
             ctx.getBean("calAssembler", CalAssembler.class);
10         calAssembler.assemble();
11
12         ctx.close();
13     }
14 }
```

1. 메이븐을 이용한 스프링 계산기 프로그램

코드 3-8 ch03_pjt_03\src\main\java\ch03_pjt_03\ICalculator.java

```
01 package ch03_pjt_03;
02 public interface ICalculator {
03     public int doOperation(int firstNum, int secondNum);
04 }
```

코드 3-9 ch03_pjt_03\src\main\java\ch03_pjt_03\MyCalculator.java

```
01 package ch03_pjt_03;
02 public class MyCalculator {
03     public void calculate(int fNum, int sNum, ICalculator calculator) {
04         // ICalculator 객체 주입
05         int value = calculator.doOperation(fNum, sNum);  // 연산 실행
06         System.out.println("result : " + value);
07     }
08 }
```

1. 메이븐을 이용한 스프링 계산기 프로그램

코드 3-10 ch03_pjt_03\src\main\java\ch03_pjt_03\CalAssembler.java

```
01 package ch03_pjt_03;
02 public class CalAssembler {
03
04     MyCalculator calculator;
05     CalAdd calAdd;
06     CalSub calSub;
07     CalMul calMul;
08     CalDiv calDiv;
09
10     public CalAssembler(MyCalculator calculator, CalAdd calAdd, CalSub calSub,
   CalMul calMul, CalDiv calDiv) {
11
12         this.calculator = calculator;
13         this.calAdd = calAdd;
14         this.calSub = calSub;
15         this.calMul = calMul;
16         this.calDiv = calDiv;
17     }
18
19     public void assemble() {
20         calculator.calculate(10, 5, calAdd);
21         calculator.calculate(10, 5, calSub);
22         calculator.calculate(10, 5, calMul);
23         calculator.calculate(10, 5, calDiv);
24     }
25 }
```

1. 메이븐을 이용한 스프링 계산기 프로그램

코드 3-11 ch03_pjt_03\src\main\java\ch03_pjt_03\CalAdd.java

```
01 package ch03_pjt_03;
02 public class CalAdd implements ICalculator {
03     public int doOperation(int firstNum, int secondNum) {
04         return firstNum + secondNum;
05     }
06 }
```

코드 3-12 ch03_pjt_03\src\main\java\ch03_pjt_03\CalSub.java

```
01 package ch03_pjt_03;
02 public class CalSub implements ICalculator {
03     public int doOperation(int firstNum, int secondNum) {
04         return firstNum - secondNum;
05     }
06 }
```

코드 3-13 ch03_pjt_03\src\main\java\ch03_pjt_03\CalMul.java

```
01 package ch03_pjt_03;
02 public class CalMul implements ICalculator {
03     public int doOperation(int firstNum, int secondNum) {
04         return firstNum * secondNum;
05     }
06 }
```

코드 3-14 ch03_pjt_03\src\main\java\ch03_pjt_03\CalDiv.java

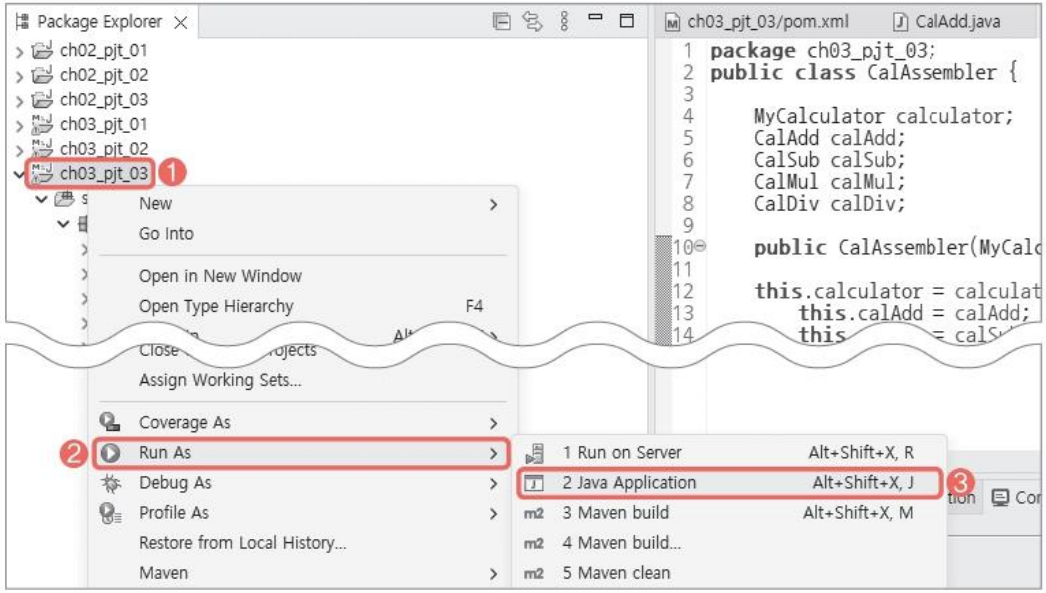
```
01 package ch03_pjt_03;
02 public class CalDiv implements ICalculator {
03     public int doOperation(int firstNum, int secondNum) {
04         return secondNum != 0 ? (firstNum / secondNum) : 0;
05     }
06 }
```

7. [src/main/resources]에 applicationContext.xml 생성하기

코드 3-15 ch03_pjt_03\src\main\resources\applicationContext.xml

```
01 <?xml version="1.0" encoding="UTF-8"?>
02
03 <beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd">
04
05     <bean id="cAdd" class="ch03_pjt_03.CalAdd" />
06     <bean id="cSub" class="ch03_pjt_03.CalSub" />
07     <bean id="cMulw" class="ch03_pjt_03.CalMul" />
08     <bean id="cDiv" class="ch03_pjt_03.CalDiv" />
09
10     <bean id="myCalculator" class="ch03_pjt_03.MyCalculator" />
11
12     <bean id="calAssembler" class="ch03_pjt_03.CalAssembler" >
13         <constructor-arg ref="myCalculator" />
14         <constructor-arg ref="cAdd" />
15         <constructor-arg ref="cSub" />
16         <constructor-arg ref="cMul" />
17         <constructor-arg ref="cDiv" />
18     </bean>
19
20 </beans>
```

8. ch03_pjt_03 프로그램을 실행하기([Run As]-[Java Application])



실행 결과

```
result : 15
result : 5
result : 50
result : 2
```