POLYTECH

Spring Framework

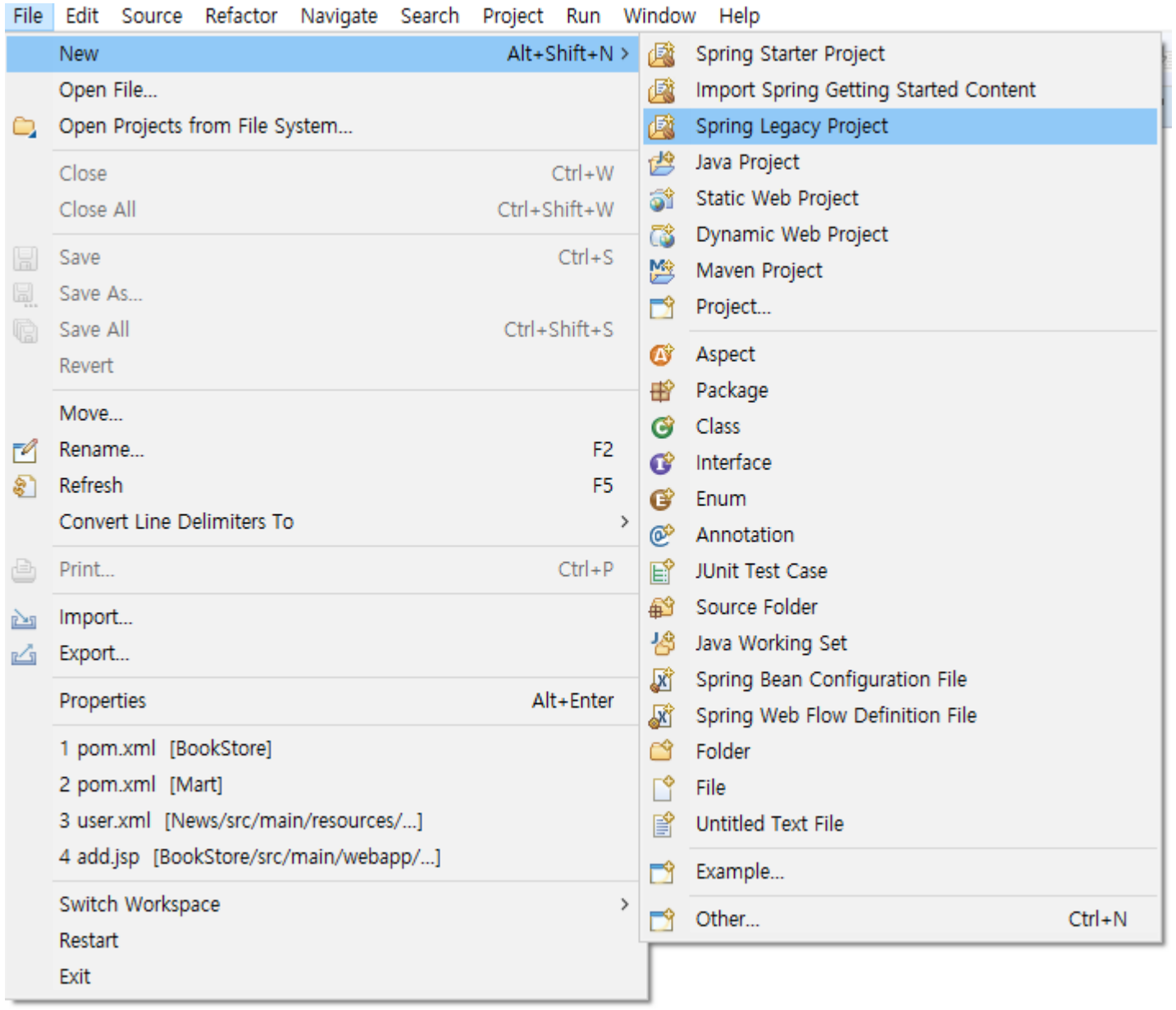
프로젝트 생성 및 CRUD 기본

Oracle

한국폴리텍대학 대전캠퍼스
스마트소프트웨어과
안광민

프로젝트 만들기 (1)

File -> New -> Spring Legacy Project

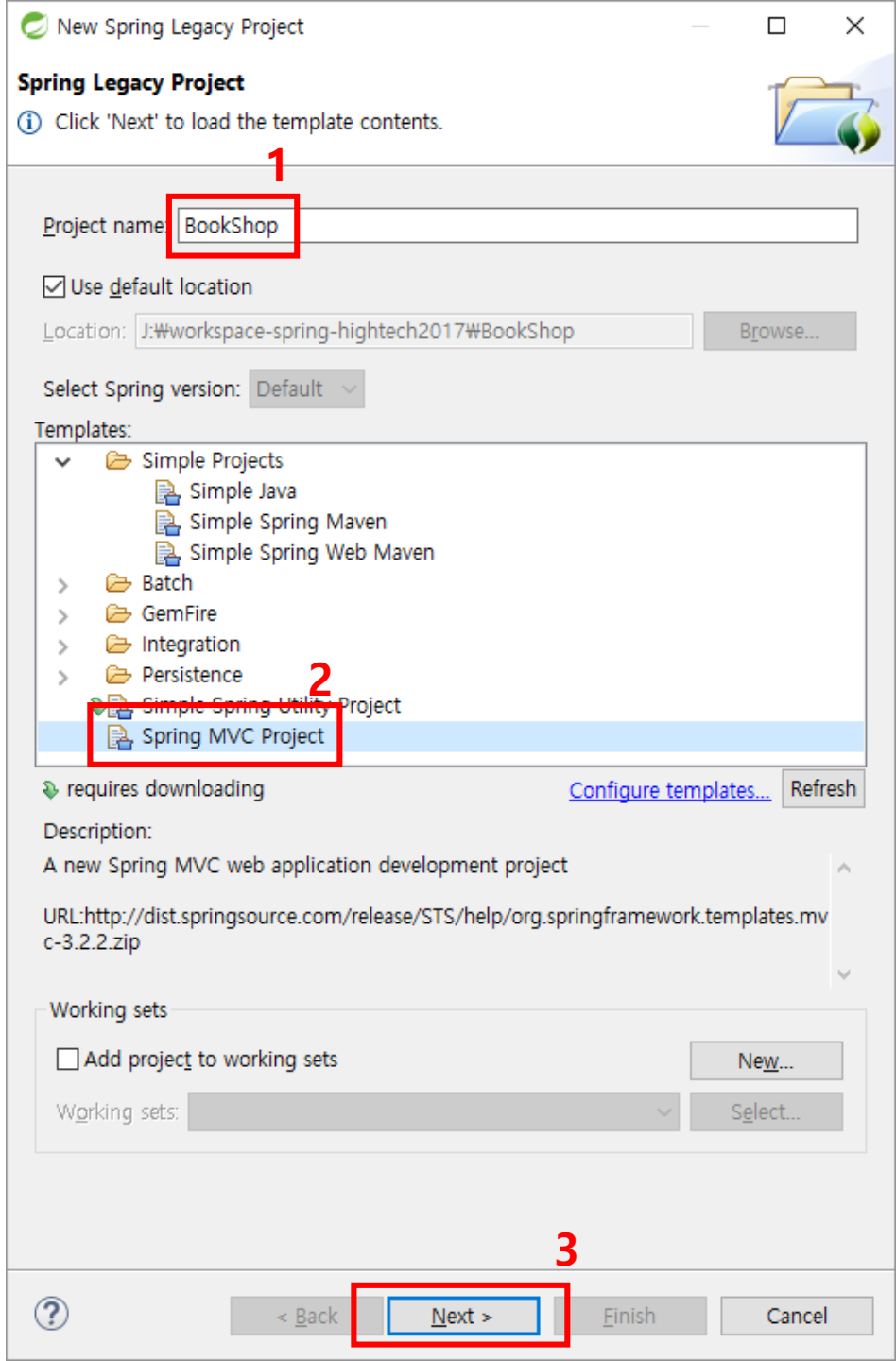


프로젝트 만들기 (2)

1) 프로젝트명 입력

2) Templates -> Spring MVC Project

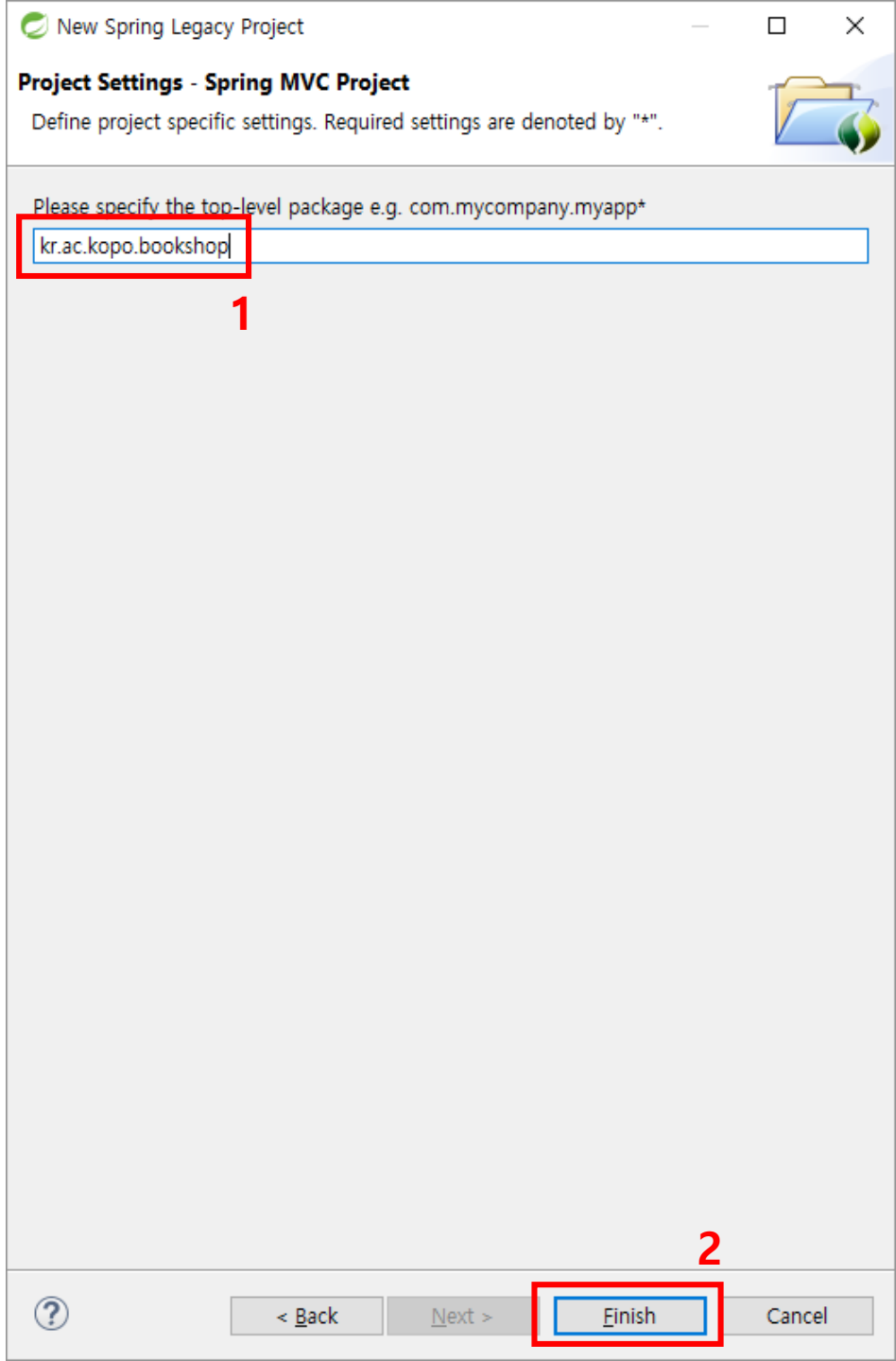
3) Next



프로젝트 만들기 (3)

1) 패키지명 설정
   - 패키지명은 도메인을 역순으로 배치하고 프로젝트명을 추가한다

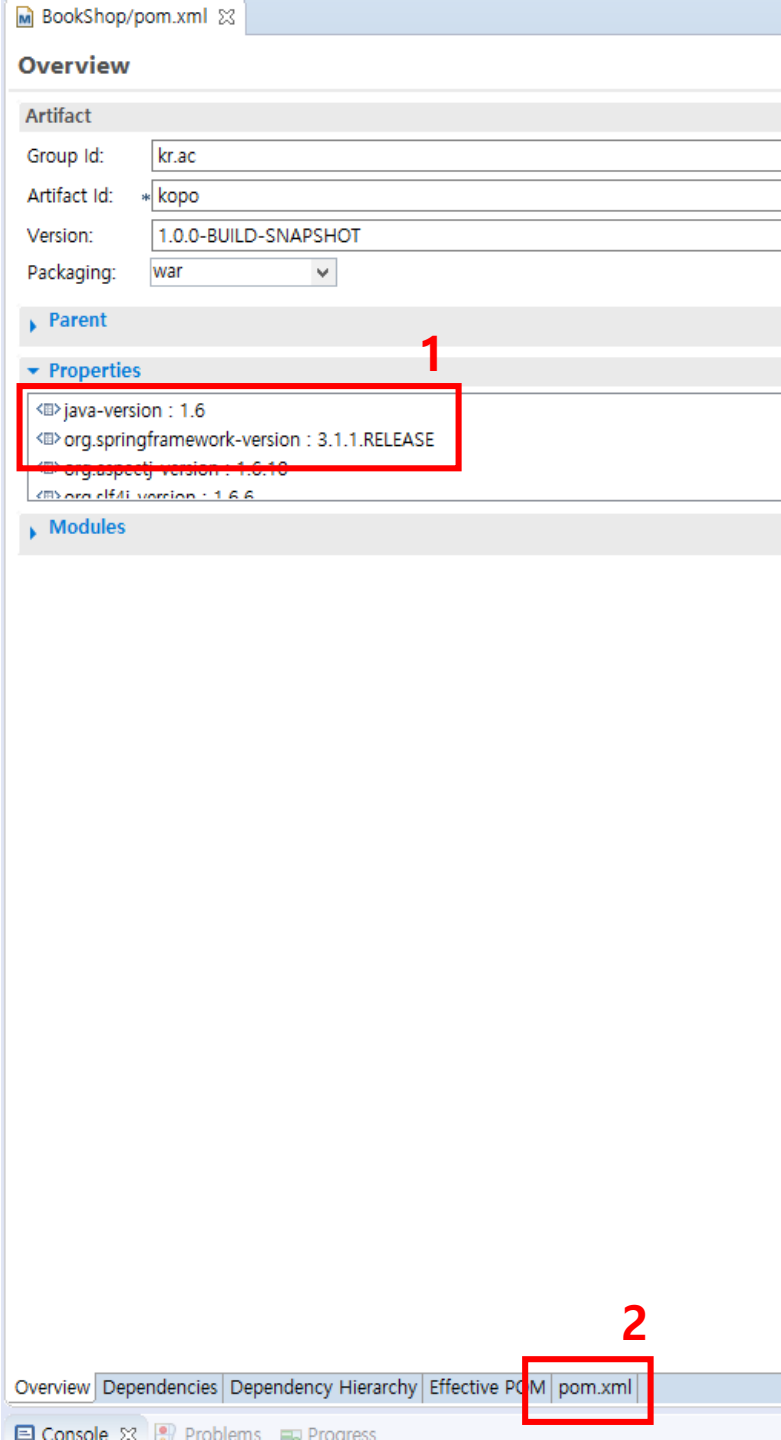
2) Finish
   - 메이븐(Maven) 빌드가 이루어지는 동안 기다린다 (우측하단에 진행상황 표시)



메이븐 설정 (pom.xml)

1) pom.xml 열기
   - 자바 버전 1.6 -> 1.8 로 변경
   - 스프링 버전 3.1.1.RELEASE -> 4.2.5.RELEASE 로 변경

2) pom.xml 탭 열기
   - <dependencies></dependencies> 에 <dependency></dependency> 가 다수 배치 됨
   - mvnrepository.com 에서 필요한 모듈을 찾아 추가



```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>4.2.5.RELEASE</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.4.4</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis-spring</artifactId>
    <version>1.3.1</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.commons/commons-dbcp2 -->
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-dbcp2</artifactId>
    <version>2.1.1</version>
</dependency>
```

Oracle JDBC Driver 설정

1) ojdbc6.jar 복사 (오라클 11g 이하 버전의 경우)
   - src > main > webapp > WEB-INF > lib 폴더

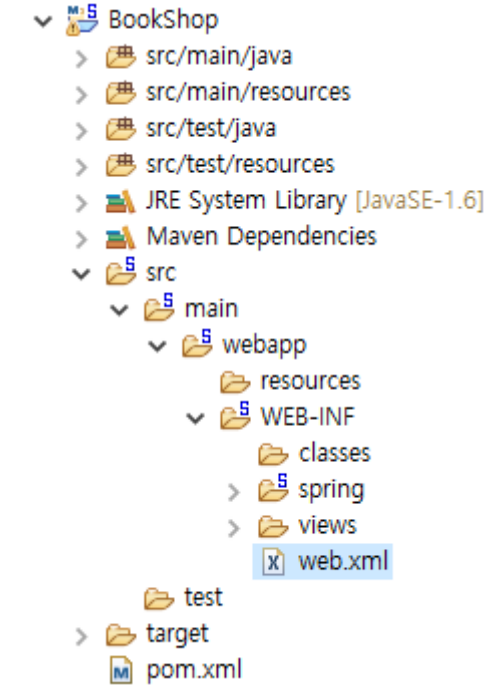
2) ojdbc8.jar 복사 (오라클 12c 이상 버전의 경우)
   - src > main > webapp > WEB-INF > lib 폴더

* jar 파일은 오라클 설치 폴더에 jdbc 라는 폴더에 존재 함

한글처리 설정

1) web.xml 설정
   - src > main > webapp > WEB-INF 폴더
   - </servlet-mapping> 다음에 추가

```
<filter>
        <filter-name>encodingFilter</filter-name>
        <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
    <init-param>
            <param-name>encoding</param-name>
            <param-value>UTF-8</param-value>
        </init-param>
</filter>
<filter-mapping>
        <filter-name>encodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>
```

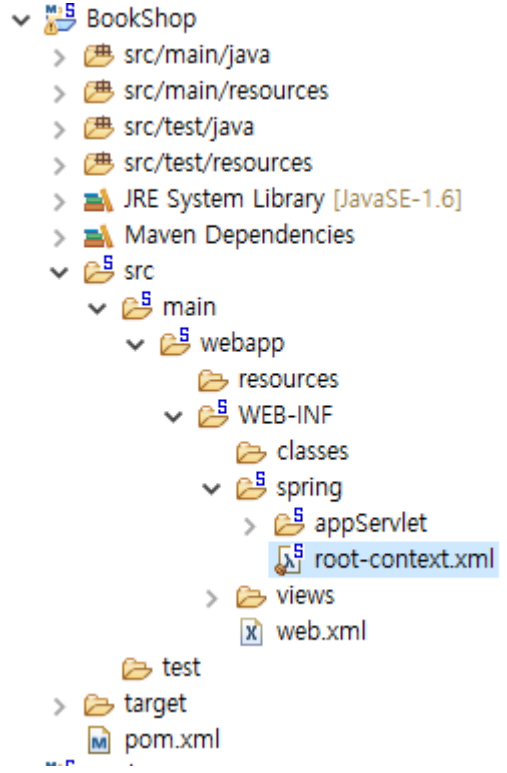


스프링 설정

1) root-context.xml 설정

- src > main > webapp > WEB-INF > spring 폴더

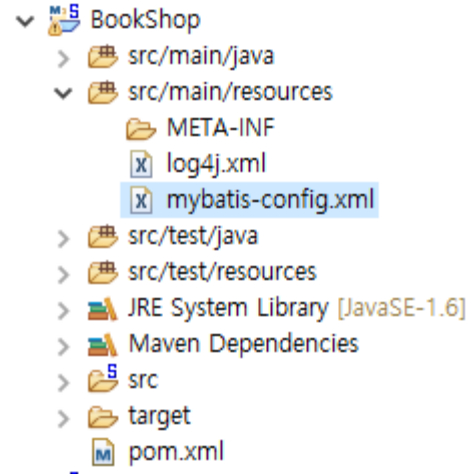
```
<bean id="dataSource" class="org.apache.commons.dbcp2.BasicDataSource" destroy-method="close">
    <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver"/>
    <property name="url" value="jdbc:oracle:thin:@127.0.0.1:1521:xe"/>
    <property name="username" value="madang"/>
    <property name="password" value="madang"/>
</bean>

<bean id="sqlSession" class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dataSource" />
    <property name="configLocation" value="classpath:mybatis-config.xml"></property>
</bean>

<bean id="sqlSessionTemplate" class="org.mybatis.spring.SqlSessionTemplate">
    <constructor-arg  index="0"  ref="sqlSession"/>
</bean>
```



마이바티스 설정

1) mybatis-config.xml 생성
   - src/main/resources 폴더
   - 필요에 따라 `<typeAliases>` 와 `<mapper>` 를 추가

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN" "http://mybatis.org/dtd/mybatis-3-config.dtd">

<configuration>
	<typeAliases>
		<typeAlias type="kr.ac.kopo.bookshop.model.Book" alias="Book" />
	</typeAliases>

	<mappers>
		<mapper resource="mybatis/book.xml" />
	</mappers>
</configuration>
```

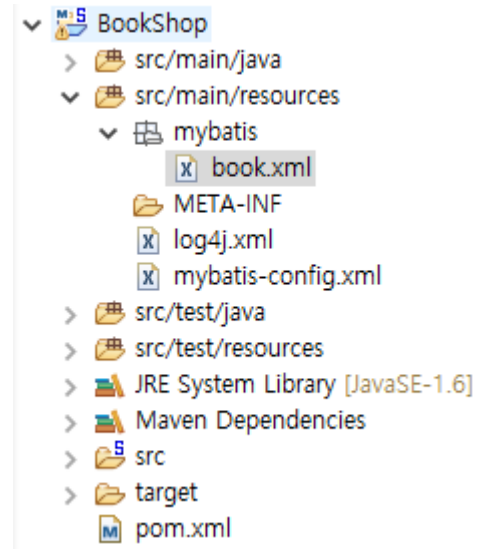


마이바티스 매퍼 추가

1) book.xml 생성

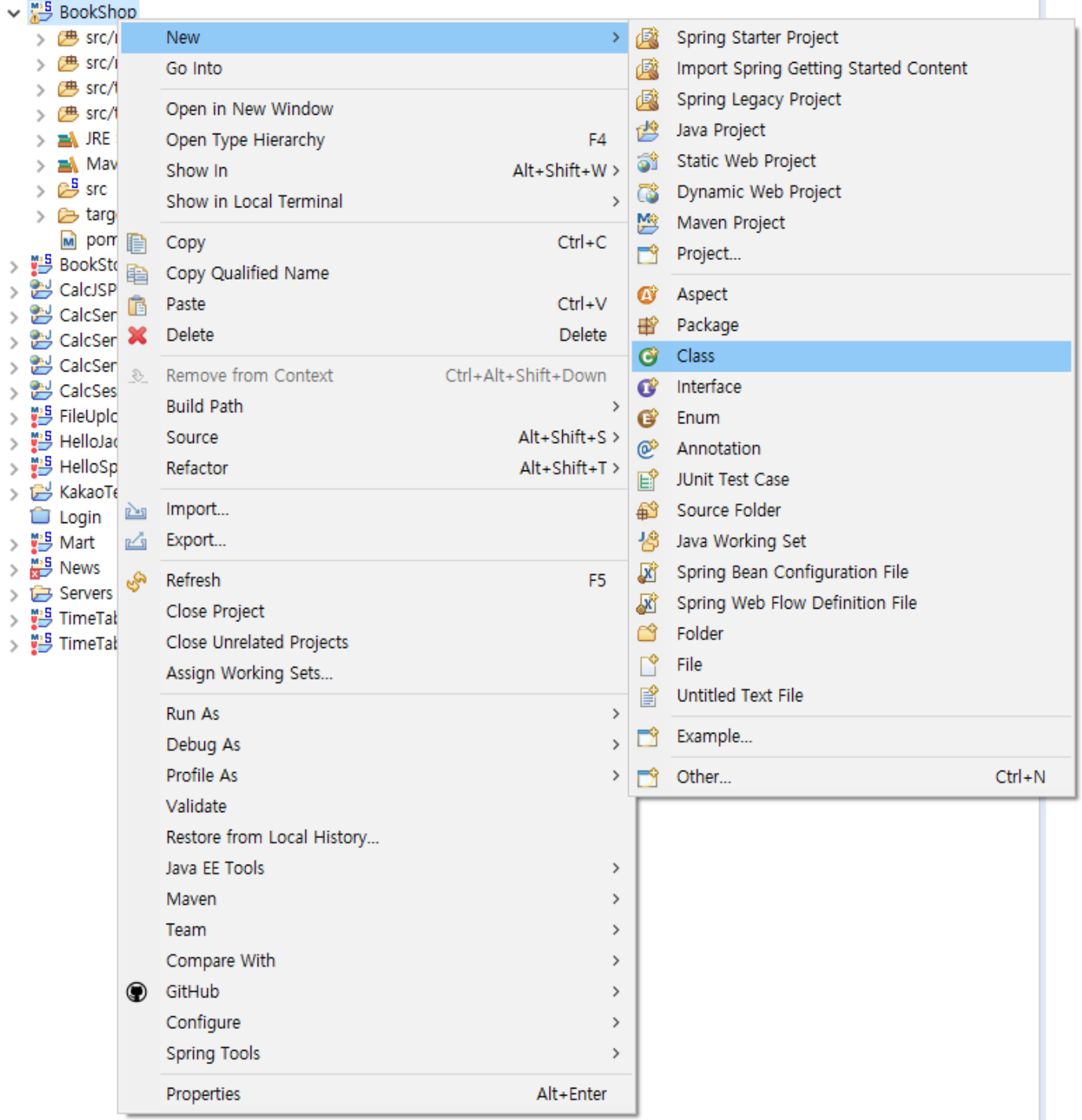
- src/main/resources 에 mybatis 폴더 생성

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="book">
	<select id= "list" resultType="Book">
		SELECT * FROM book
	</select>

	<delete id="delete">
		DELETE FROM book WHERE bookid=#{bookid}
	</delete>

	<insert id="add">
		INSERT INTO book (bookid, bookname, publisher, price)
		VALUES ((#{bookid}, #{bookname}, #{publisher}, #{price})
	</insert>

	<update id="update">
		UPDATE book SET bookname=#{bookname}, publisher=#{publisher}, price=#{price}
		WHERE bookid=#{bookid}
	</update>

	<select id="item" resultType="Book">
		SELECT * FROM book WHERE bookid=#{bookid}
	</select>
</mapper>
```



모델 클래스 추가 (1)

1) Book 클래스 생성

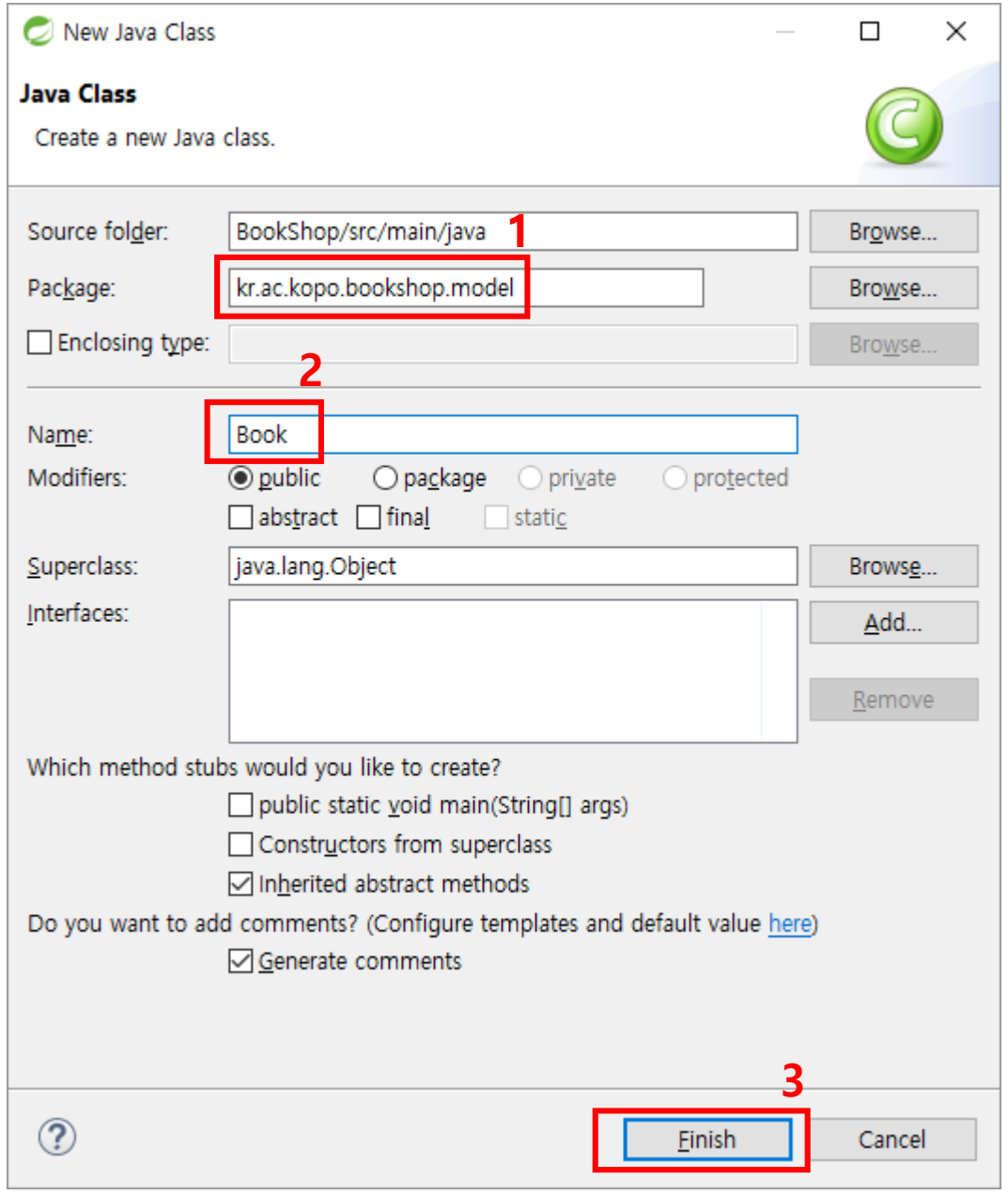


모델 클래스 추가 (2)

1) 패키지명은 기본값에 model 을 추가

2) 클래스명은 Book
   - BookVO 혹은 BookDTO 와 같이 작성하는 경우도 있음

3) Finish

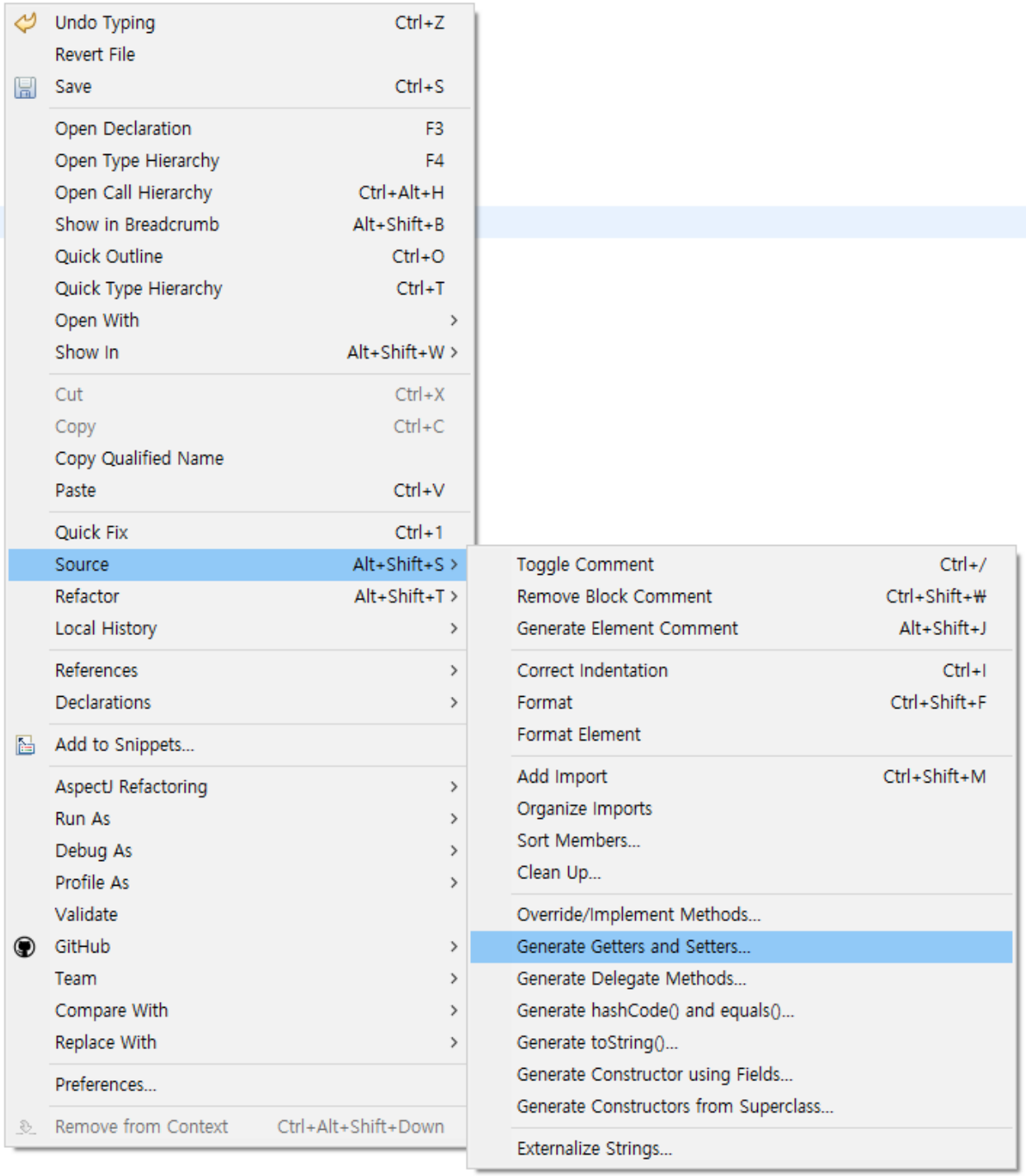


모델 클래스 추가 (3)

1) 데이터베이스 테이블에 대응하는 필드 작성

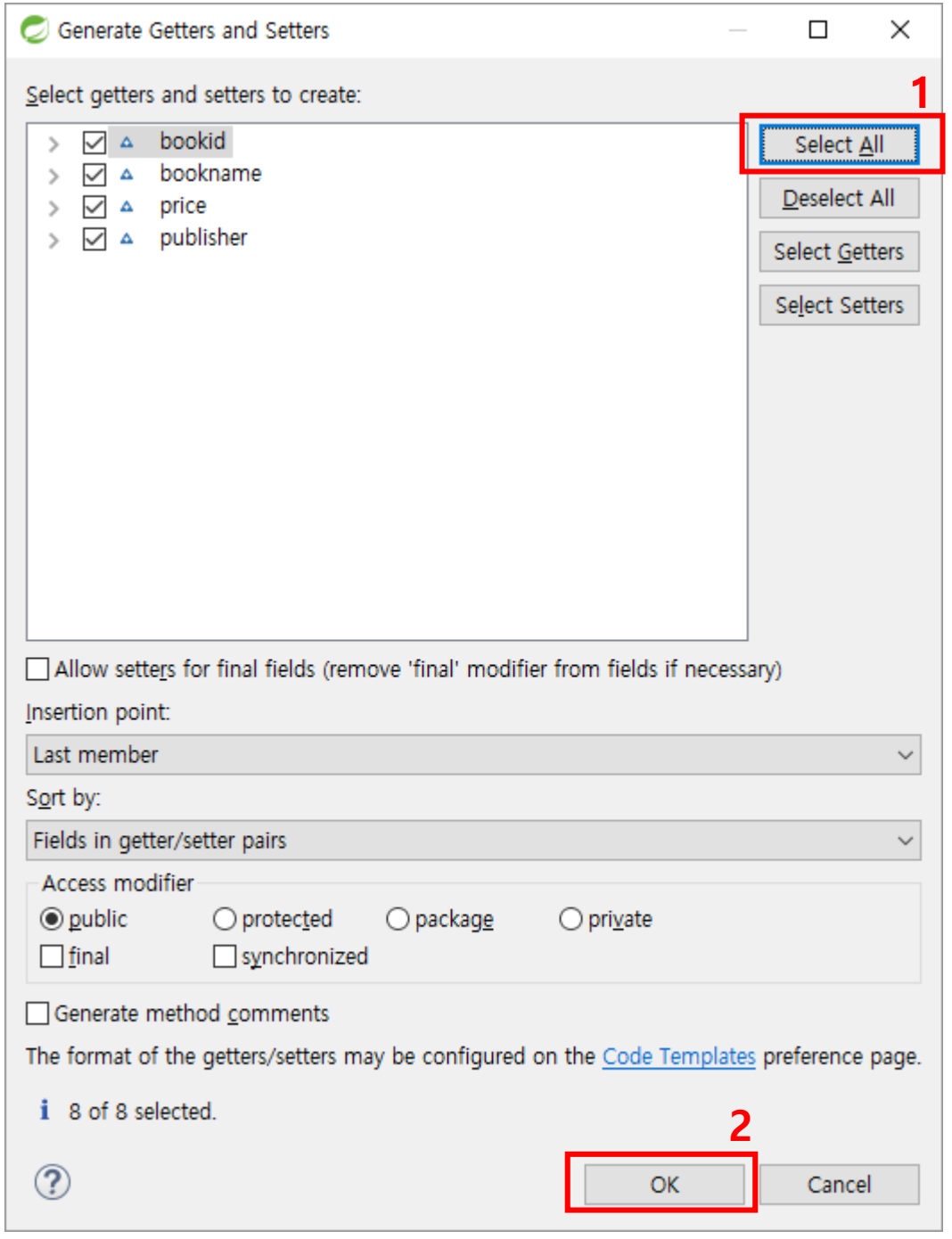
```
int bookid;
String bookname;
String publisher;
int price;
```

2) Getter Setter 를 생성



모델 클래스 추가 (4)

1) Getter Setter 를 생성

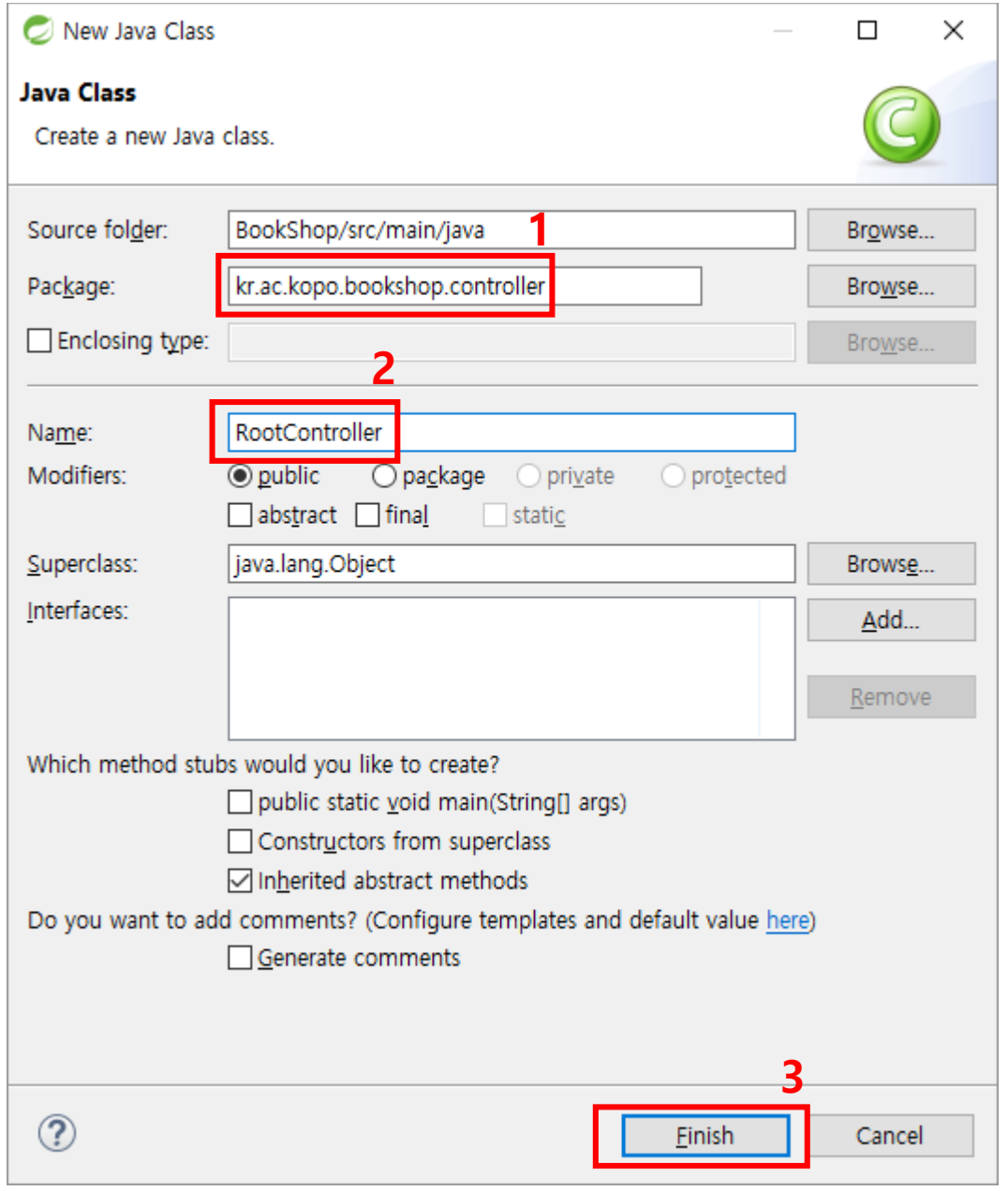


RootController 클래스 생성 (1)

1) 패키지명은 기본값에 controller 를 추가해서 작성

2) 클래스명은 RootController
   - 자바에서 두개 이상의 단어가 합성되는 경우는 각 단어의 첫 글자는 대문자 (낙타 표기법)

3) Finish



RootController 클래스 생성 (2)

1) @Controller

2) @RequestMapping("/")

3) @RequestMapping(value="/")

```
String index() {
        return "index";
}
```

index.jsp 생성

1) src > main > webapp > WEB-INF > views 에 생성

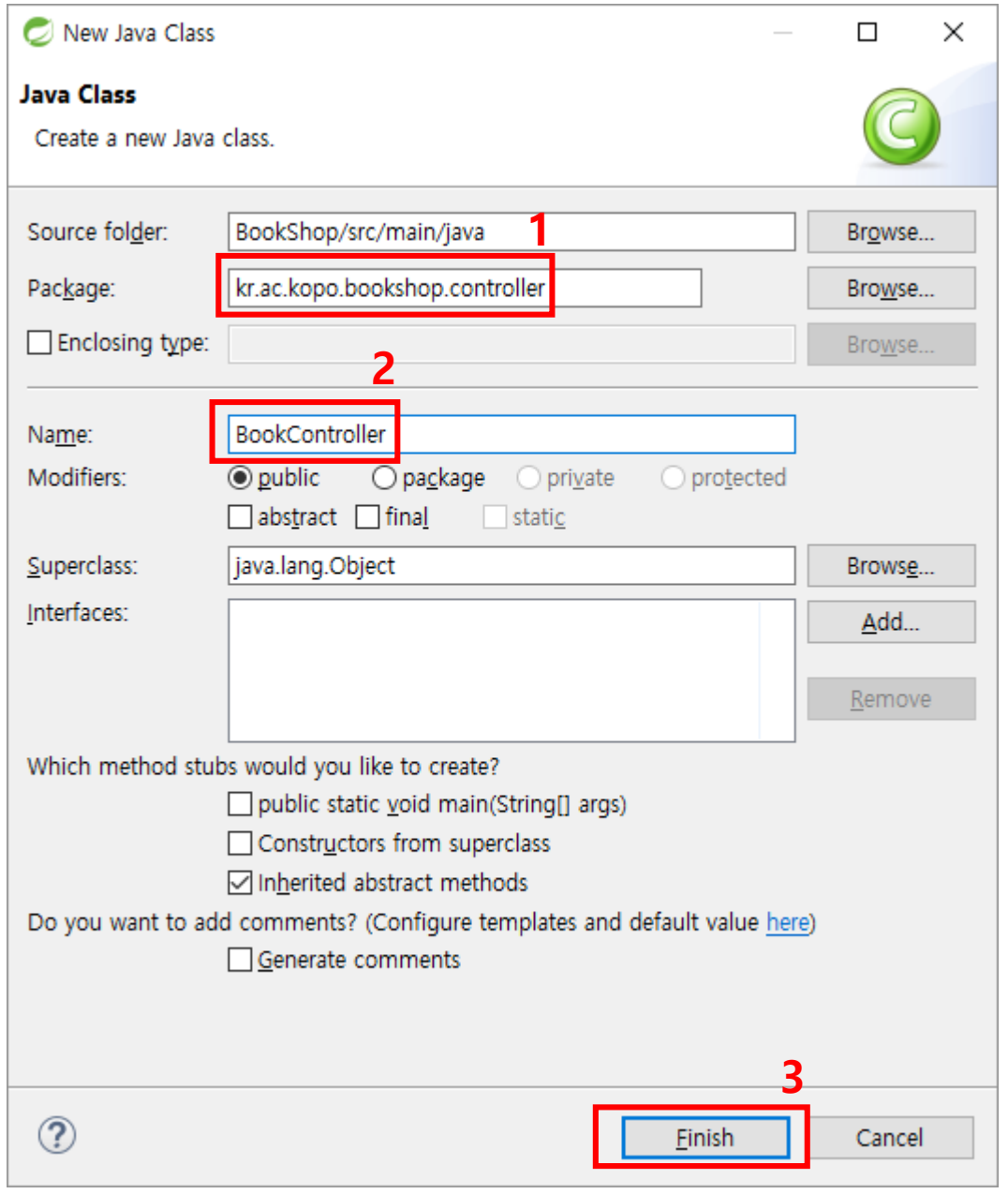
```
<a href="book/list">도서 관리</a>
```

Controller 클래스 생성 (1)

1) 패키지명은 기본값에 controller 를 추가해서 작성

2) 클래스명은 BookController
   - 자바에서 두개 이상의 단어가 합성되는 경우는 각 단어의 첫 글자는 대문자 (낙타 표기법)

3) Finish



Controller 클래스 생성 (2)

1) @Controller

2) @RequestMapping("/book")

3) final String path = "book/";

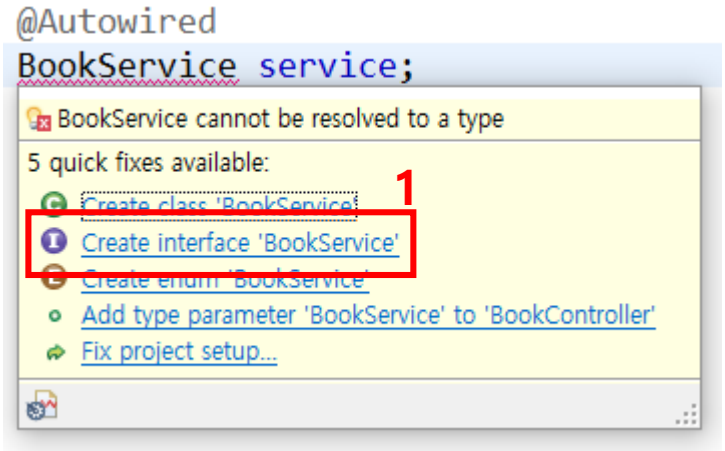
4) @Autowired
   BookService service;

```
@Controller
@RequestMapping(value="/book")
public class BookController {
	final String path = "book/";

	@Autowired
	BookService service;

	@RequestMapping(value="/list")
	String list(Model model) {
		List<Book> list = service.getList();

		model.addAttribute("list", list);

		return path + "list";
	}

	@RequestMapping(value="/delete")
	String delete(int bookid) {
		service.delete(bookid);

		return "redirect:list";
	}

	@RequestMapping(value="/add", method=RequestMethod.GET)
	String add() {
		return path + "add";
	}

	@RequestMapping(value="/add", method=RequestMethod.POST)
	String add(Book item) {
		service.add(item);

		return "redirect:list";
	}
```

```
	@RequestMapping(value="/update", method=RequestMethod.GET)
	String update(int bookid, Model model) {
		Book item = service.getItem(bookid);

		model.addAttribute("item", item);

		return path + "update";
	}

	@RequestMapping(value="/update", method=RequestMethod.POST)
	String update(Book item) {
		service.update(item);

		return "redirect:list";
	}
}
```

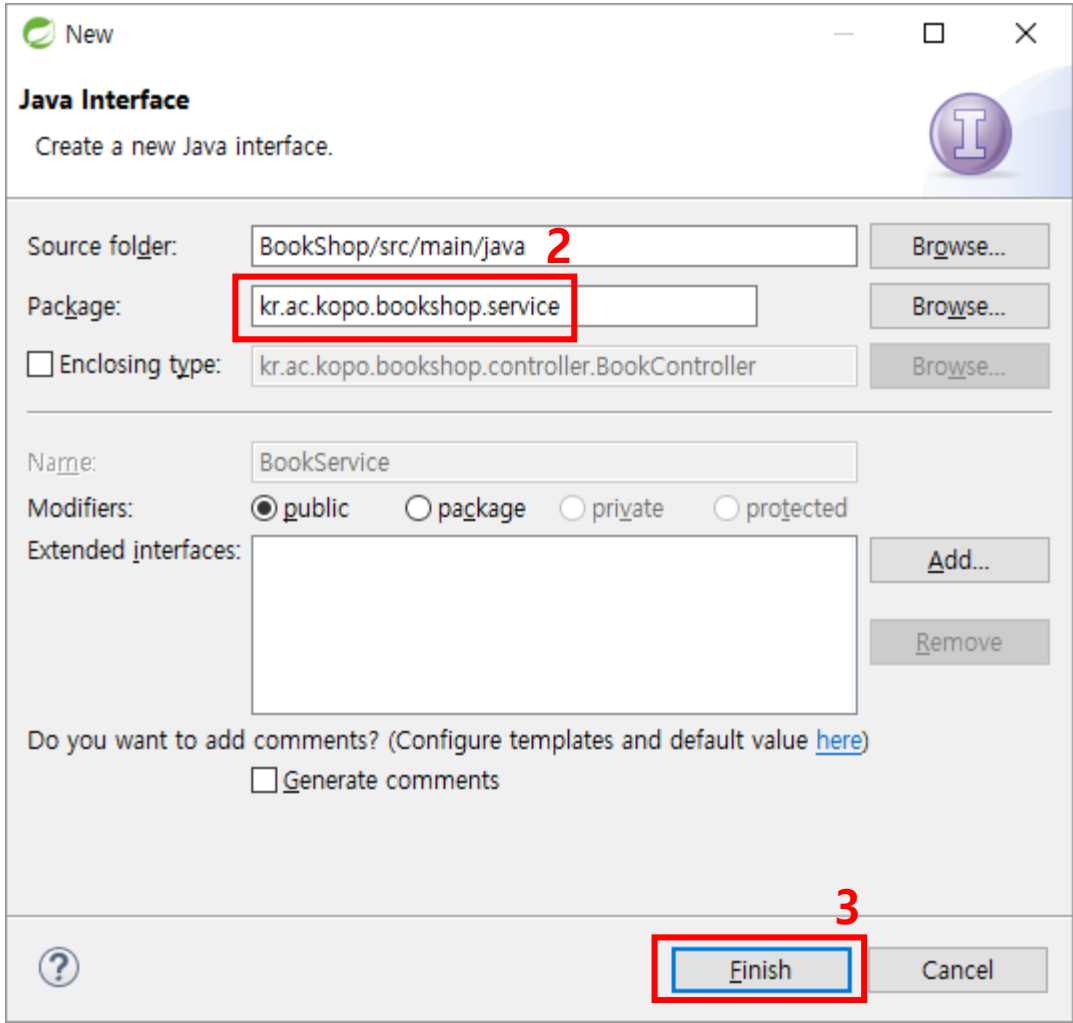
Service 인터페이스 생성

1) BookController 파일에서 BookService 에 마우스를 위치하여 "Create interface 'BookService'" 를 클릭하여 BookService 인터페이스 추가



2) 팝업창에서 Package 를 기본값에 service 를 추가하여 작성

3) Finish



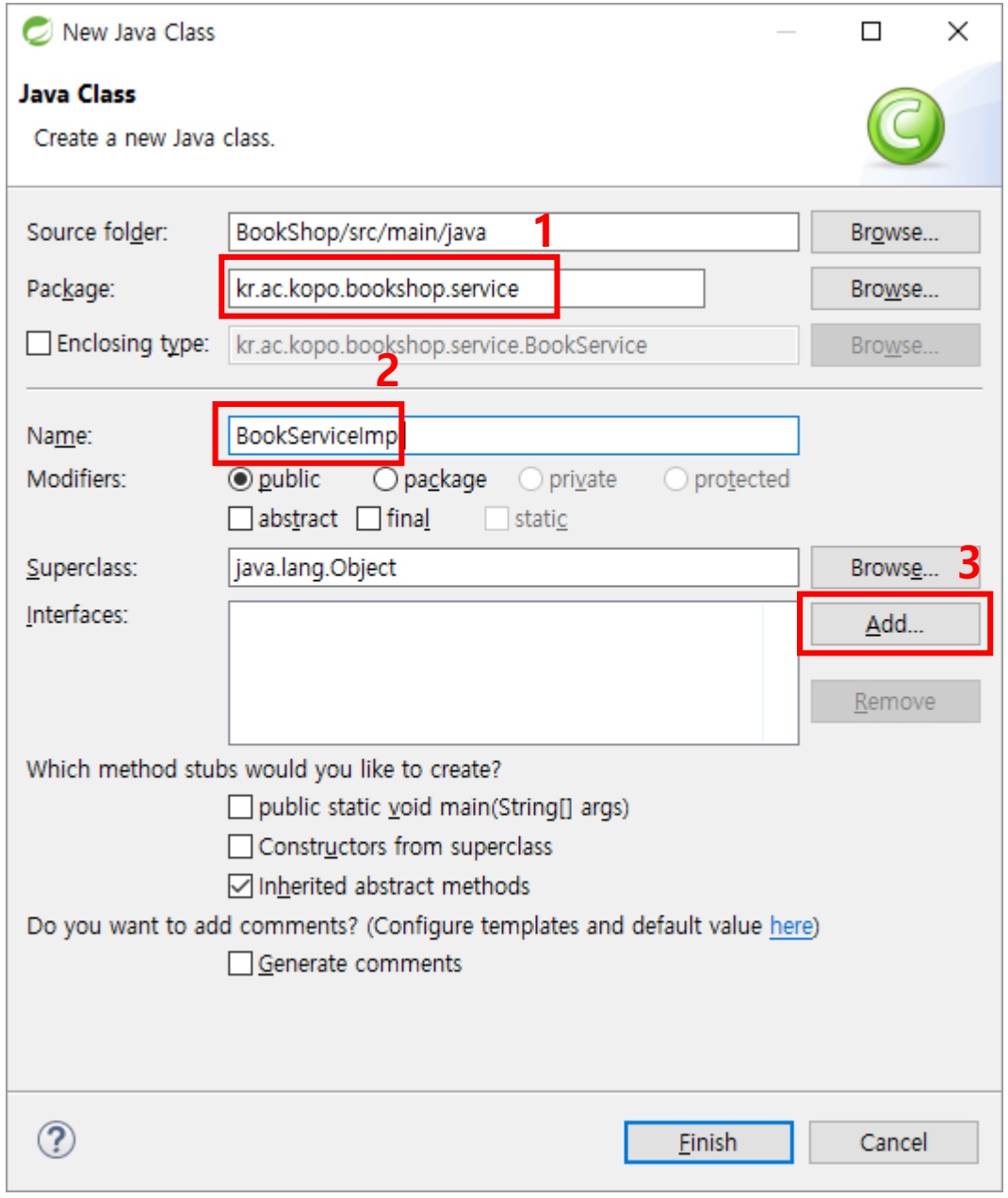
```
public interface BookService {

        List<Book> getList();

        void delete(int bookid);

        void add(Book item);

        Book getItem(int bookid);

        void update(Book item);

}
```

Service 클래스 생성 (1)

1) 패키지명은 기본값에 service 를 추가하여 작성

2) 클래스명은 BookServiceImpl
   - BookService 인터페이스를 구현했다는 의미

3) Add 버튼을 클릭하여 인터페이스 지정



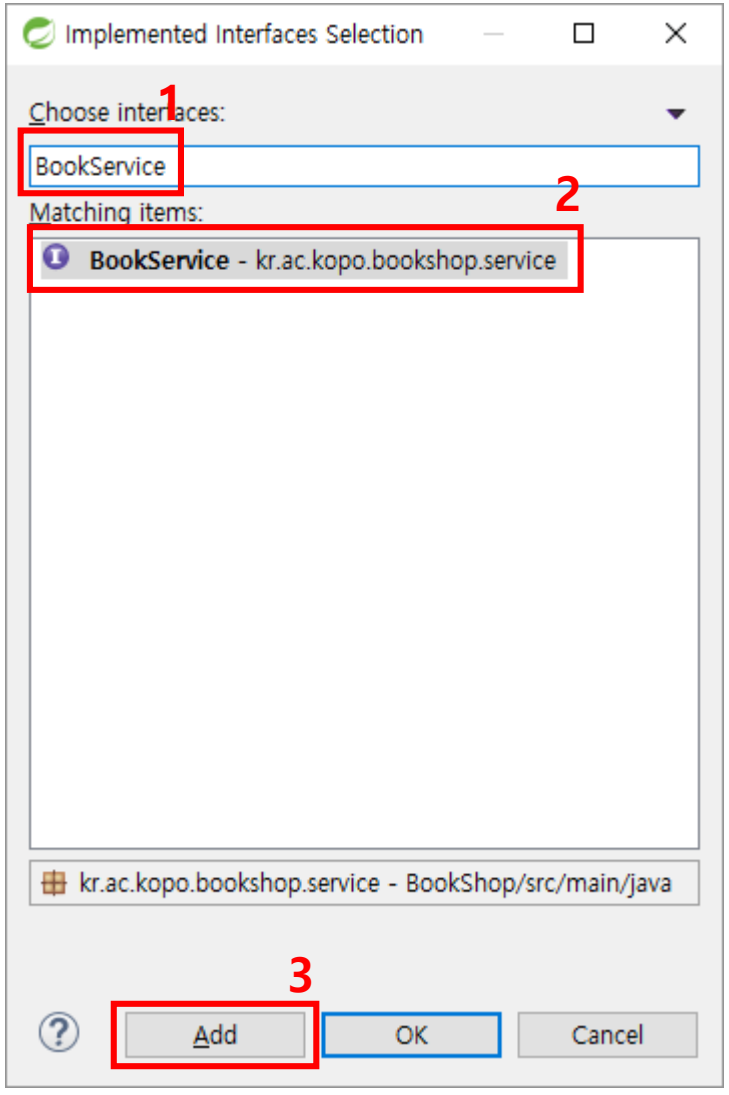
Service 클래스 생성 (2)

1) 구현할 서비스 인터페이스명을 입력

2) 검색 된 인터페이스 중에서 해당 인터페이스를 선택

3) Add 버튼을 클릭하여 인터페이스 지정

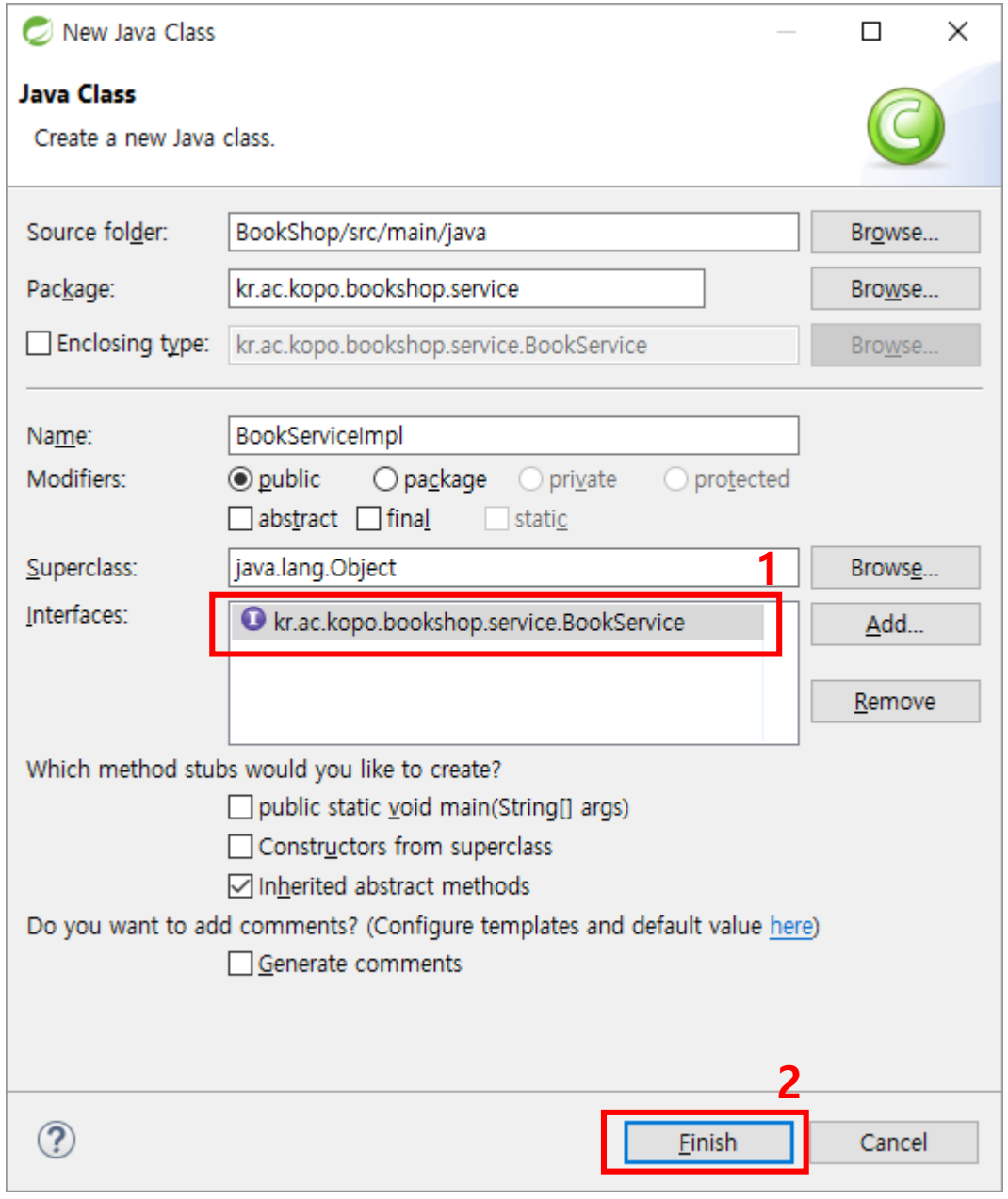
4) OK



Service 클래스 생성 (3)

1) 인터페이스 확인

2) Finish



Service 클래스 생성 (4)

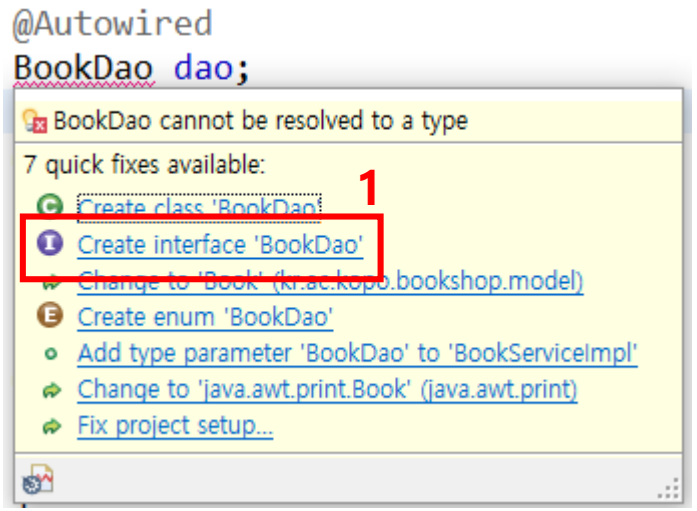
1) @Service

2) @Autowired
BookDao dao;

```java
@Service
public class BookServiceImpl implements BookService {
	@Autowired
	BookDao dao;

	@Override
	public List<Book> getList() {
		return dao.getList();
	}

	@Override
	public void delete(int bookid) {
		dao.delete(bookid);
	}

	@Override
	public void add(Book item) {
		dao.add(item);
	}

	@Override
	public Book getItem(int bookid) {
		return dao.getItem(bookid);
	}

	@Override
	public void update(Book item) {
		dao.update(item);
	}
}
```

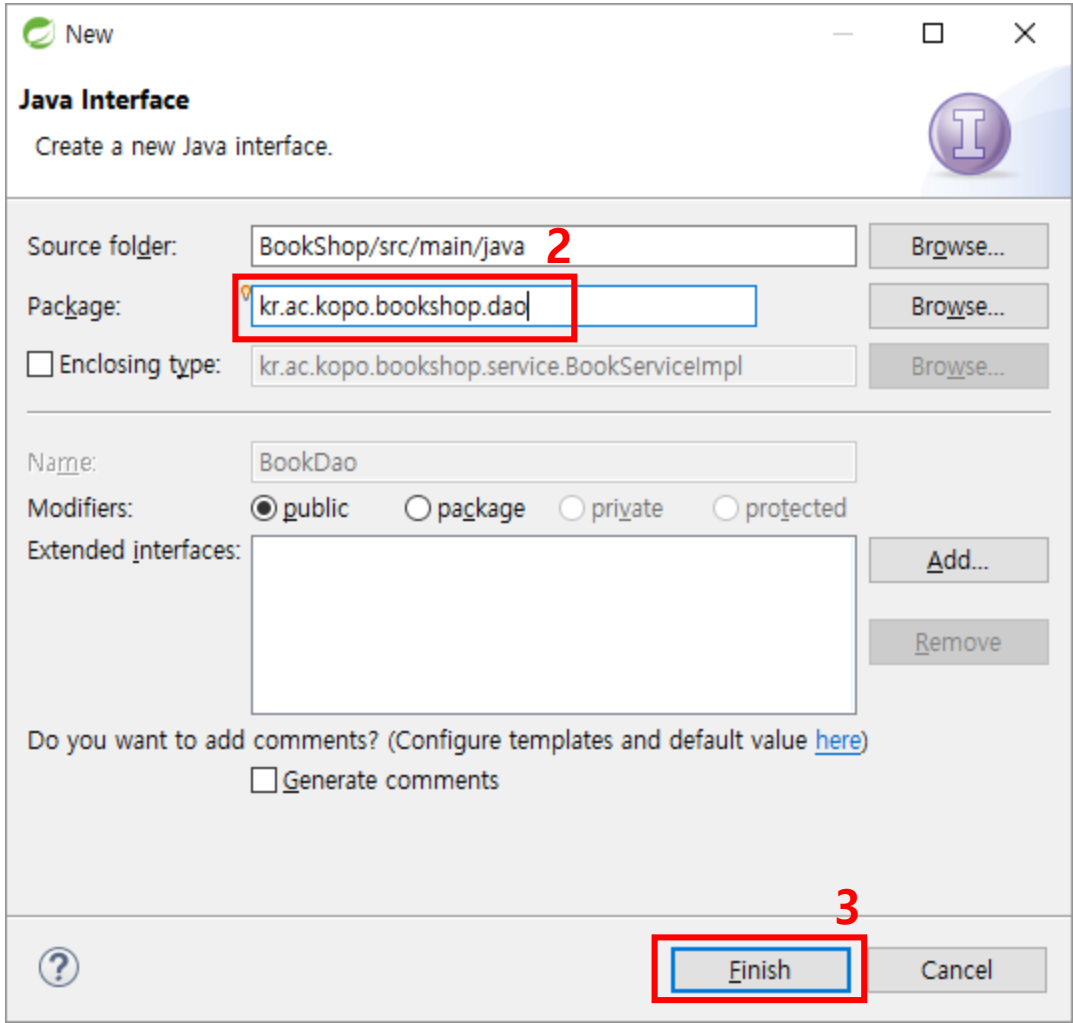
Dao 인터페이스 생성

1) BookServiceImpl 파일에서 BookDao 에 마우스를 위치하여 "Create interface 'BookDao'" 를 클릭하여 BookDao 인터페이스 추가



2) 팝업창에서 패키지를 기본값에 dao 를 추가하여 작성

3) Finish



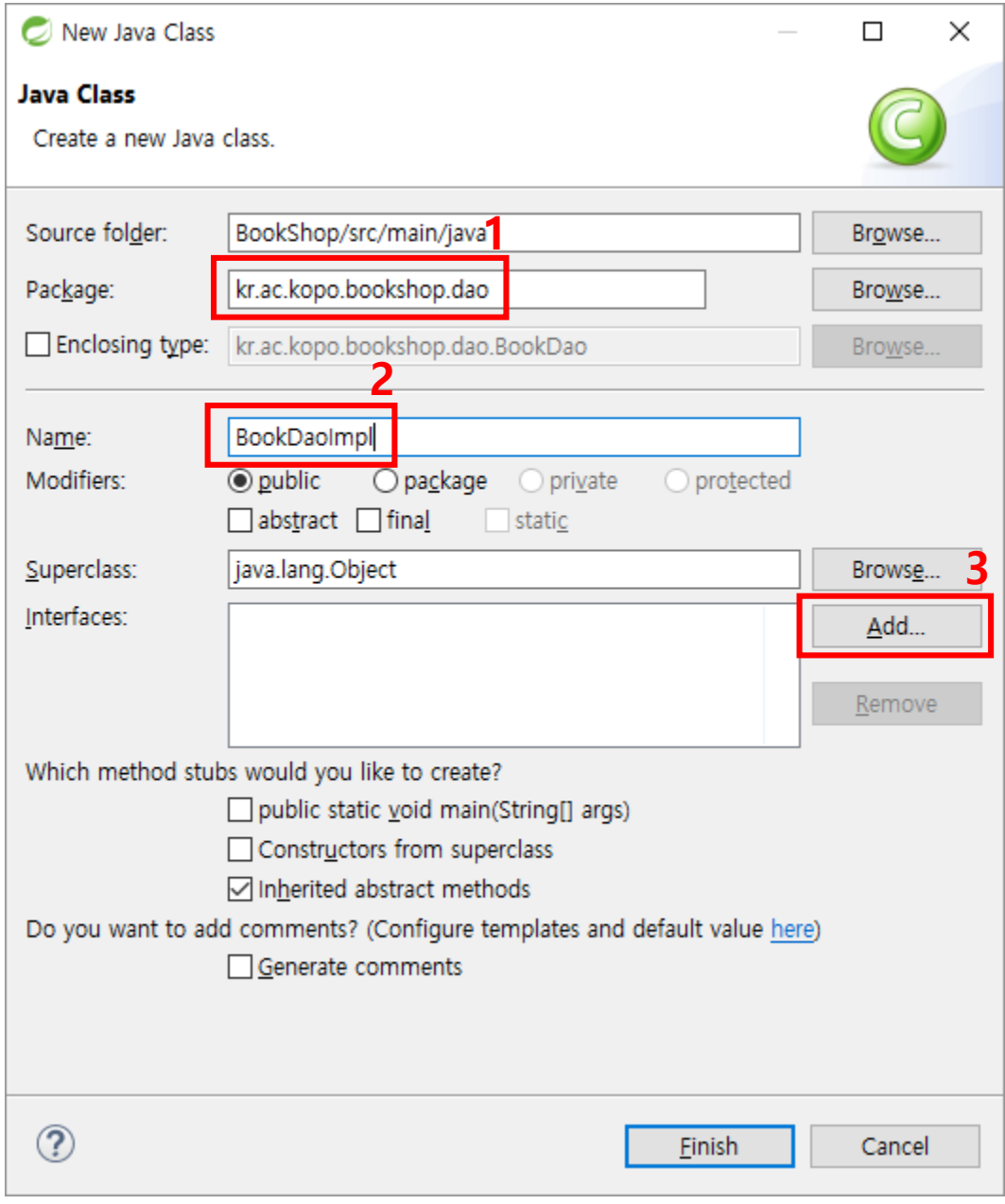
```
public interface BookDao {

        List<Book> getList();

        void delete(int bookid);

        void add(Book item);

        Book getItem(int bookid);

        void update(Book item);

}
```

Dao 클래스 생성 (1)

1) 패키지명은 기본값에 dao 를 추가하여 작성

2) 클래스명은 BookDaoImpl
   - BookDao 인터페이스를 구현했다는 의미

3) Add 버튼을 클릭하여 인터페이스 지정



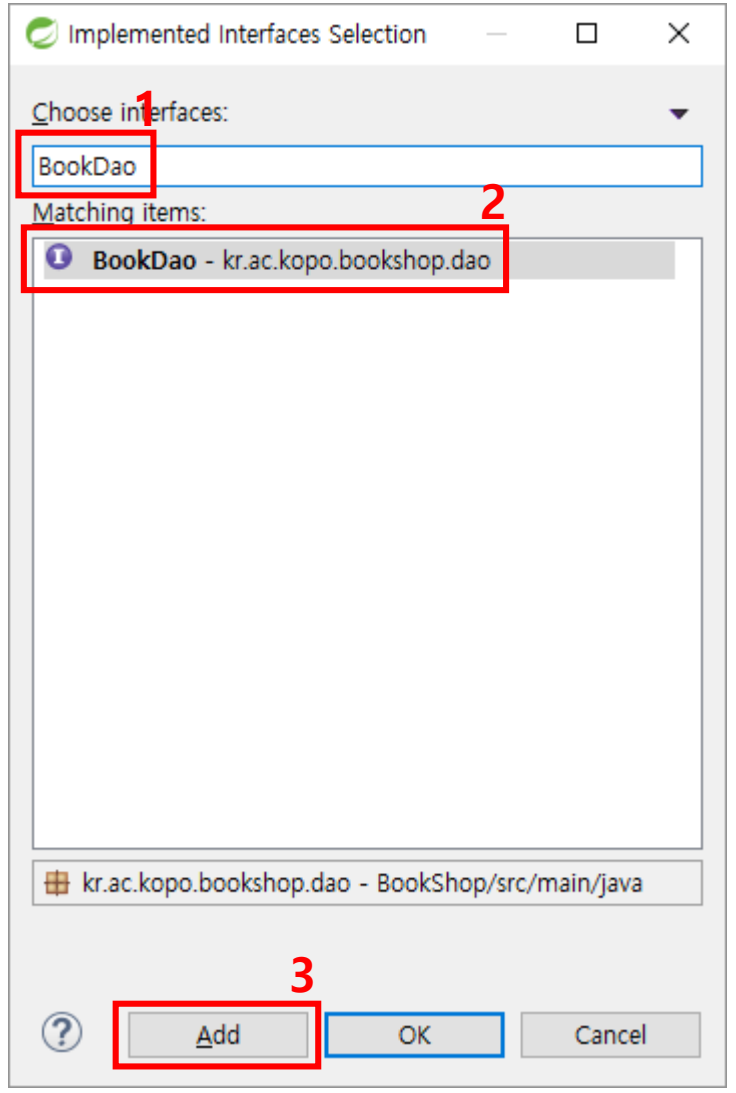
Dao 클래스 생성 (2)

1) 구현할 서비스 인터페이스명을 입력

2) 검색 된 인터페이스 중에서 해당 인터페이스를 선택

3) Add 버튼을 클릭하여 인터페이스 지정

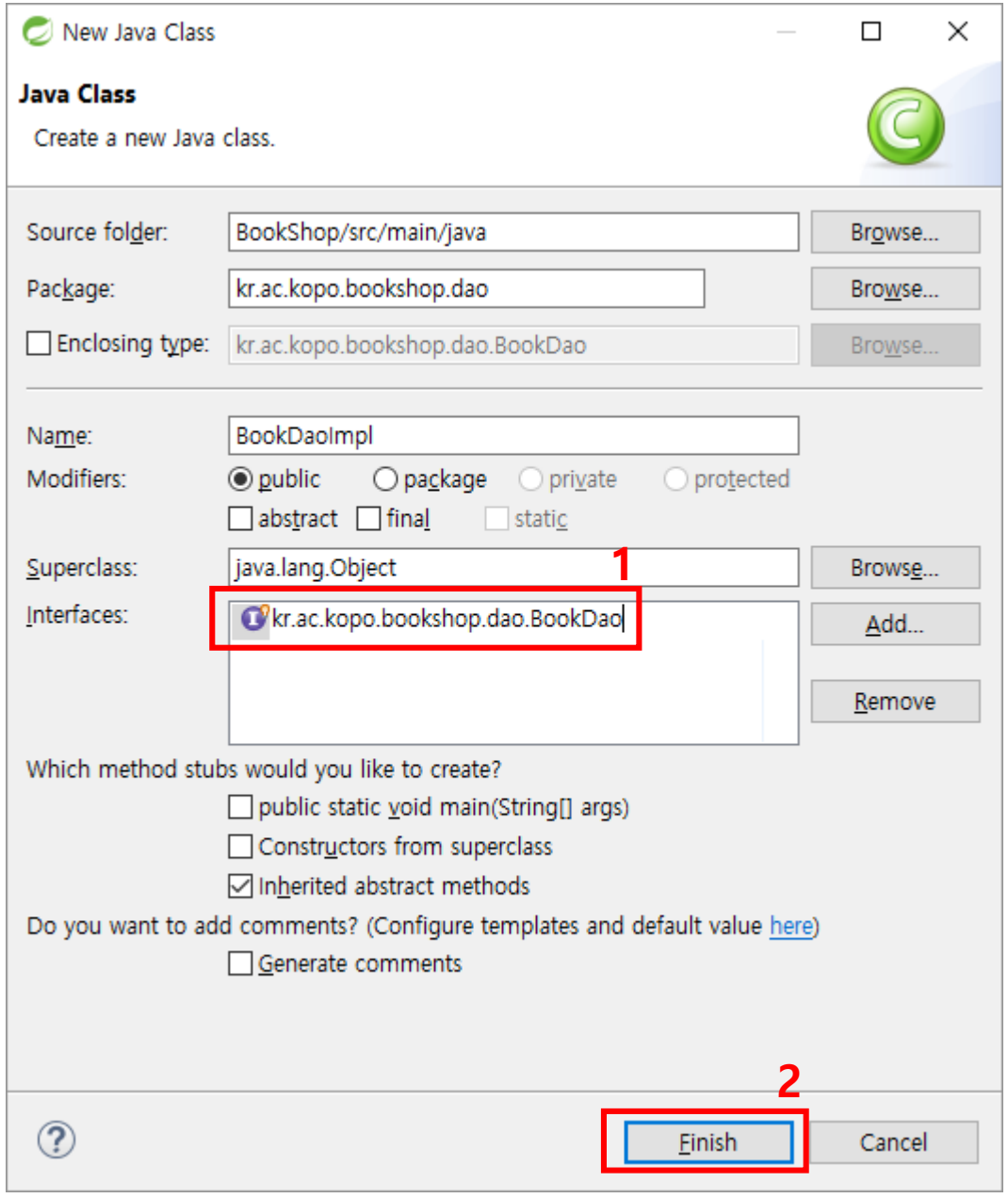
4) OK



Dao 클래스 생성 (3)

1) 인터페이스 확인

2) Finish



Dao 클래스 생성 (4)

1) @Repository

2) @Autowired
SqlSession sql;

```java
@Repository
public class BookDaoImpl implements BookDao {

	@Autowired
	SqlSession sql;

	@Override
	public List<Book> getList() {
		return sql.selectList("book.getList");
	}

	@Override
	public void delete(int bookid) {
		sql.delete("book.delete", bookid);
	}

	@Override
	public void add(Book item) {
		sql.insert("book.add", item);
	}

	@Override
	public Book getItem(int bookid) {
		return sql.selectOne("book.getItem", bookid);
	}

	@Override
	public void update(Book item) {
		sql.update("book.update", item);
	}
}
```

list.jsp 생성

1) src > main > webapp > WEB-INF > views 에 book 폴더에 생성

2) JSTL 설정
   - <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

```
<c:choose>
    <c:when test="${list.size() > 0}">
        <c:forEach var="item" items="${list}">
            <tr>
                <td>${item.bookid}</td>
                <td>${item.bookname}</td>
                <td>${item.publisher}</td>
                <td>${item.price}</td>
                <td><a href="update?bookid=${item.bookid}">변경</a>  <a href="delete?bookid=${item.bookid}">삭제</a></td>
            </tr>
        </c:forEach>
    </c:when>
    <c:otherwise>
        <tr>
            <td colspan="5">등록 된 도서가 없습니다</td>
        </tr>
    </c:otherwise>
</c:choose>
```

add.jsp 생성

1) src > main > webapp > WEB-INF > views 에 book 폴더에 생성

```
<form method="post" action="add">
        <div>
                <label>도서번호:</label>
                <input name="bookid" type="text">
        </div>
        <div>
                <label>도서명:</label>
                <input name="bookname" type="text">
        </div>
        <div>
                <label>출판사:</label>
                <input name="publisher" type="text">
        </div>
        <div>
                <label>가격:</label>
                <input name="price" type="text">
        </div>
        <div>
                <input type="submit" value="등록">
        </div>
</form>
```

update.jsp 생성

1) src > main > webapp > WEB-INF > views 에 book 폴더에 생성

```
<form method="post" action="update">
	<div>
		<label>도서번호:</label>
		<input name="bookid" type="text" value="${item.bookid}" readonly>
	</div>
	<div>
		<label>도서명:</label>
		<input name="bookname" type="text" value="${item.bookname}">
	</div>
	<div>
		<label>출판사:</label>
		<input name="publisher" type="text" value="${item.publisher}">
	</div>
	<div>
		<label>가격:</label>
		<input name="price" type="text"  value="${item.price}">
	</div>
	<div>
		<input type="submit" value="변경">
	</div>
</form>
```