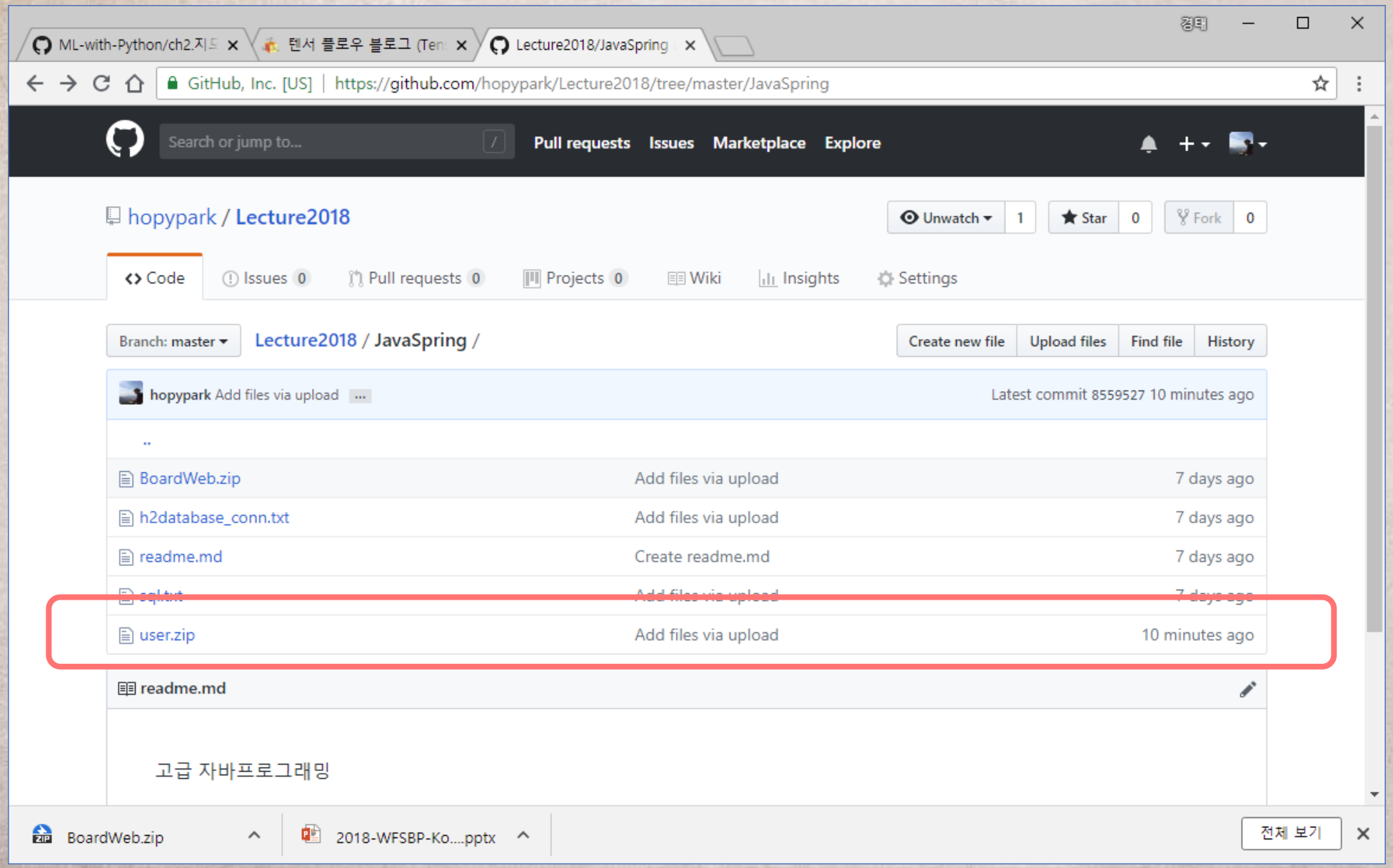
박 경 태
comsi.java@gmail.com

고급 자바 프로그래밍
: STS를 이용한 Spring 프로그래밍

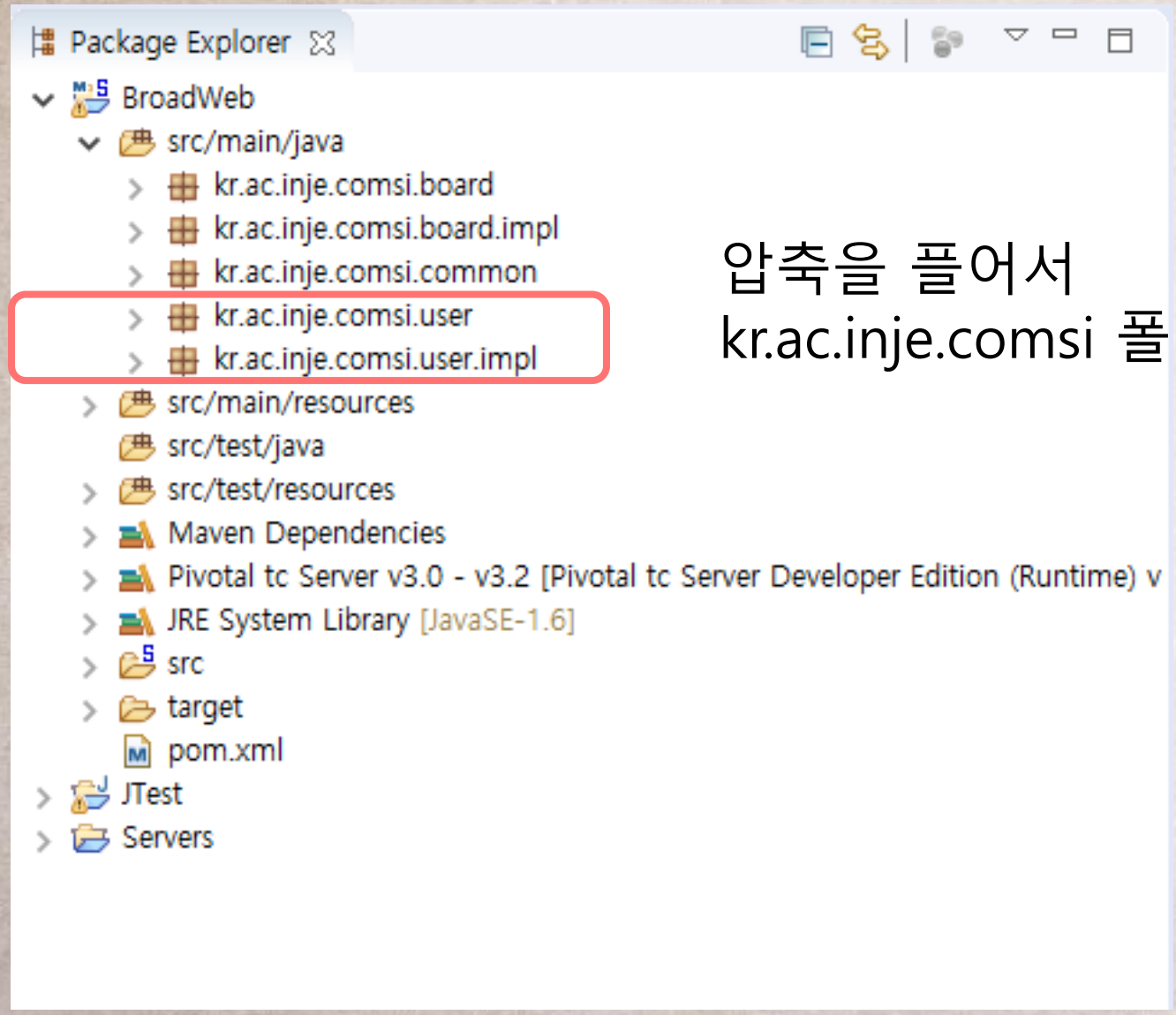
강의 내용

순서	내 용
1	• Spring IoC를 이용한 비즈니스 컴포넌트 만들기
2	• Spring AOP(Aspect Oriented Programming)를 이용한 공통 서비스 만들기 • Spring DAO(Data Access Object)를 이용한 데이터베이스 연동 및 트랜잭션 처리 • JdbcTemplate 클래스를 이용한 JDBC의 반복적 코드 제거와 SQL분리
3	• Spring MVC를 이용한 MVC 아키텍쳐 적용하기
4	• Spring MVC의 부가 기능 사용하기(파일 업로드, 다국어, 예외 처리 등)
5	• Spring과 MyBatis 연동하기 • Spring과 JPA 연동하기 • 표준 ORM(Object Relational Mapping)을 이용한 RDB와의 객체 지향적 연결과 관리

Users 테이블 처리 소스 추가하기



Users 테이블 처리 소스 추가하기



압축을 풀어서
kr.ac.inje.comsi 폴더 아래에 붙여 넣기 하면 된다.

6. 스프링 JDBC

스프링 JDBC

- JDBC(Java Database Connectivity)

➔자바에서 데이터베이스에 접속할 수 있도록 하는 자바 API이다.

➔**JDBC**는 데이터베이스에서 자료를 쿼리하거나 업데이트하는 방법을 제공

➔JDBC를 이용한 DB 연동 프로그램을 개발하면 데이터베이스에 비종속적인 DB 연동 로직을 구현할 수 있다.

  - But, JDBC 프로그램을 이용하면 개발자가 작성해야 할 코드가 너무 많다

UserDAO.java

```java
public class UserDAO {
    // JDBC 관련 변수
    private Connection conn = null;
    private PreparedStatement stmt = null;
    private ResultSet rset =null;
    // SQL 명령어들
    private final String USER_GET = "select * from users where id=? and password=?";

    // CRUD 기능의 메소드
    // 회원 등록
    public UserVO getUser(UserVO vo){
        UserVO user = null;
        try{
            System.out.println("===> JDBC로 getUser() 기능 처리");
            conn = JDBCUtil.getConnection();
            stmt = conn.prepareStatement(USER_GET);
            stmt.setString(1, vo.getId());
            stmt.setString(2, vo.getPassword());
            rset = stmt.executeQuery();

            if (rset.next()){
                user = new UserVO();
                user.setId(rset.getString("ID"));
                user.setName(rset.getString("NAME"));
                user.setPassword(rset.getString("PASSWORD"));
                user.setRole(rset.getString("ROLE"));
            }
        }catch(Exception e){
            e.printStackTrace();
        }finally{
            JDBCUtil.close(rset, stmt, conn);
        }
        return user;
    }
}
```

반복적 코드?

반복적 코드?

JDBCUtil.Java – 반복코드??

```java
public static Connection getConnection(){
    try{
        Class.forName("org.h2.Driver");
        return DriverManager.getConnection("jdbc:h2:/d:/workspace/java/h2db/myDB", "sa", "");
    }catch(Exception e){
        e.printStackTrace();
    }
    return null;
}
// close connection
public static void close(PreparedStatement stmt, Connection conn){
    if(stmt != null){
        try{
            if(!stmt.isClosed()) stmt.close();
        }catch(Exception e){
            e.printStackTrace();
        }finally{
            stmt = null;
        }
    }

    if(conn != null){
        try{
            if(!conn.isClosed()) conn.close();
        }catch(Exception e){
            e.printStackTrace();
        }finally{
            conn = null;
        }
    }

}
```

Driver 등록

Driver 연결(URL, username, password)

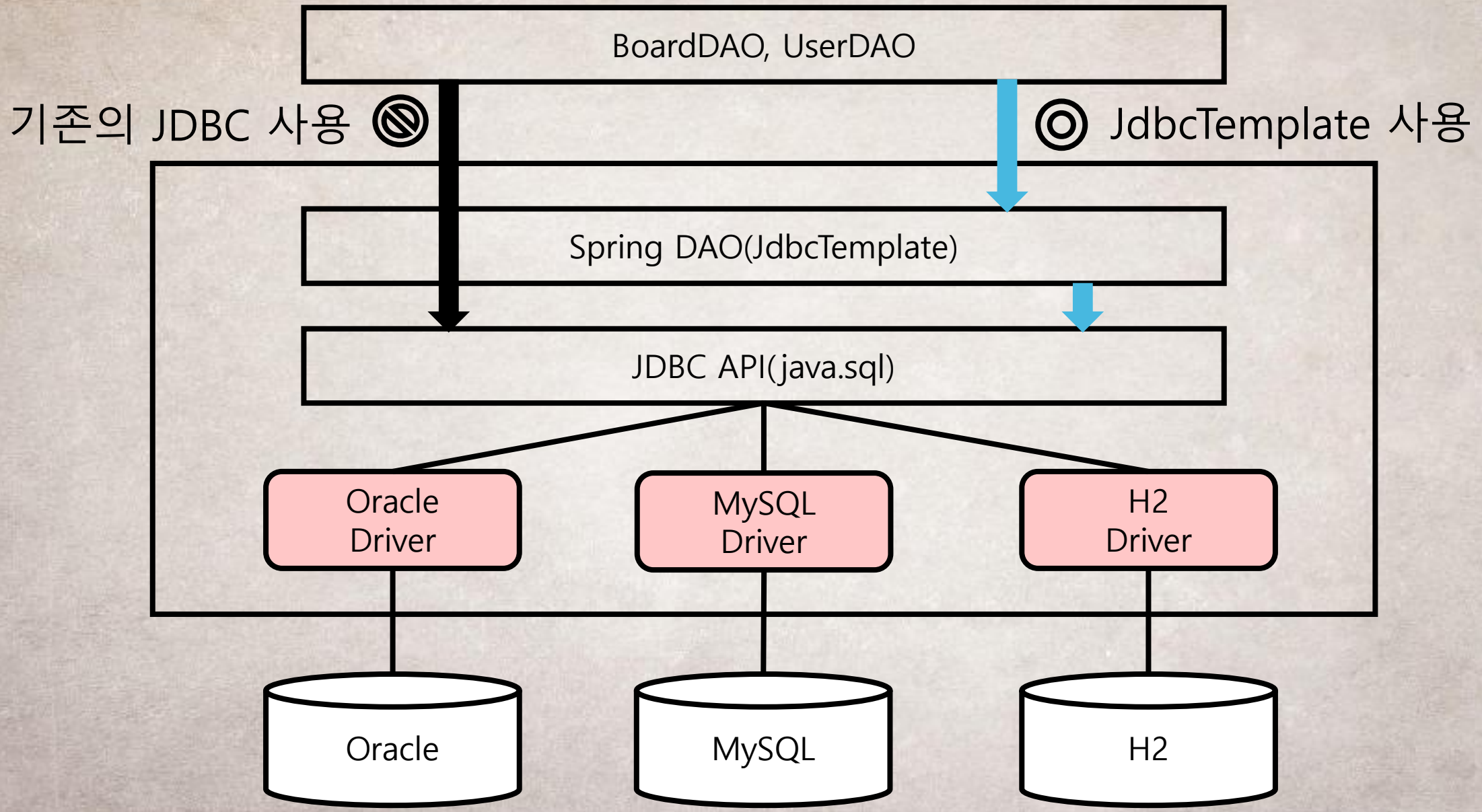
JdbcTemplate 클래스

- 스프링에서 JDBC 기반의 DB 연동 프로그램을 쉽게 개발할 수 있도록 jdbcTemplate 클래스 지원

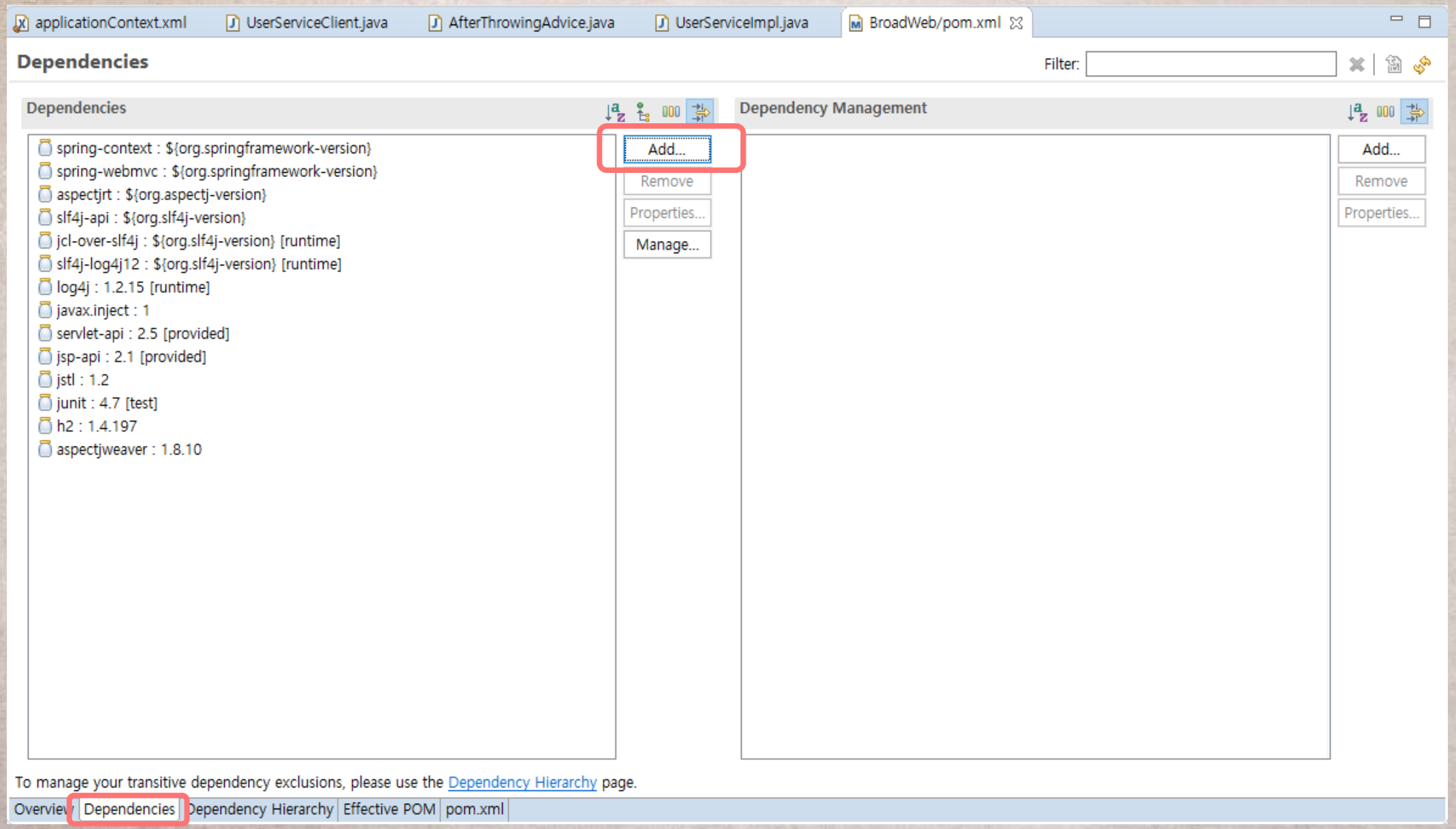
- JdbcTemplate은 Gof 디자인 패턴 중 템플릿 메소드 패턴이 적용된 클래스
  - 템플릿 메소드 패턴은 **복잡하고 반복되는 알고리즘을 캡슐화**해서 **재사용**하는 패턴으로 정의

- JDBC처럼 코딩 순서가 정형화된 기술에서 유용하게 사용
  - 반복되는 DB 연동 로직은 JdbcTemplate 클래스의 템플릿 메소드가 제공
  - **개발자는 달라지는 SQL 구문과 설정 값 만 신경 씀**

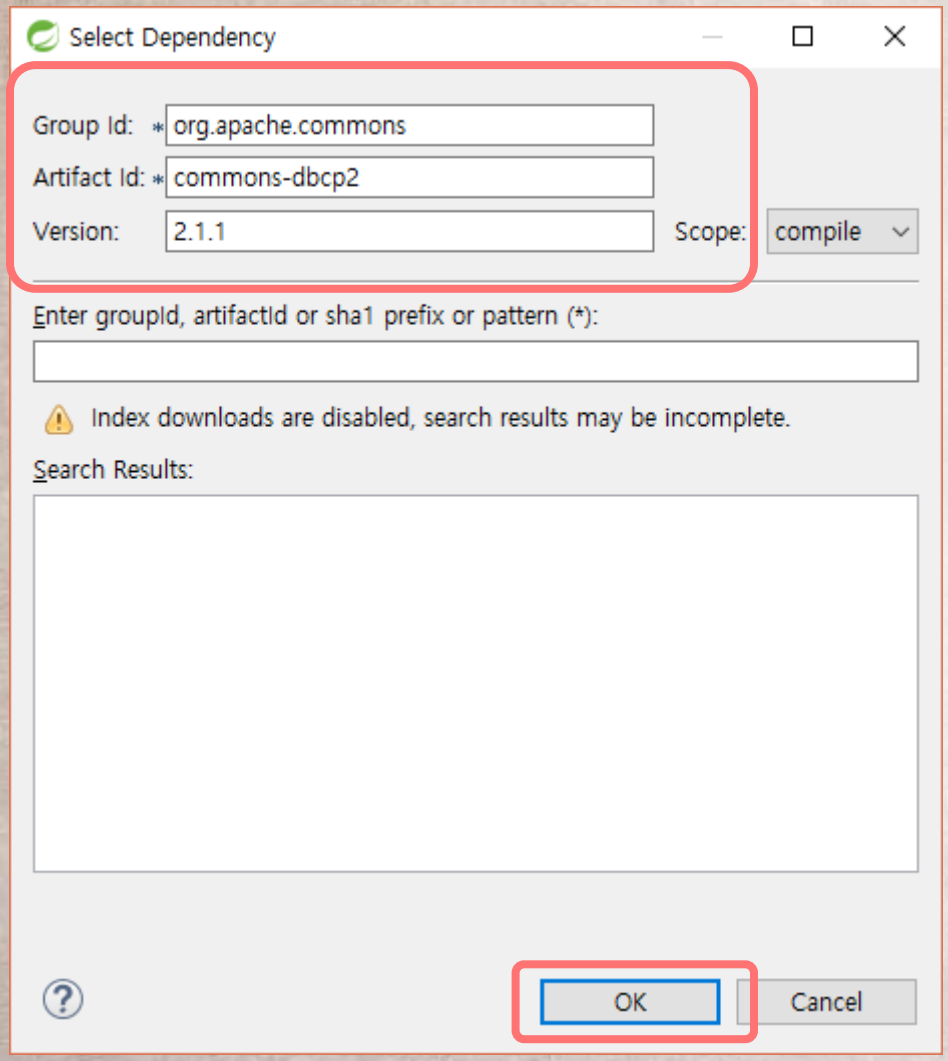
JdbcTemplate의 위치와 역할

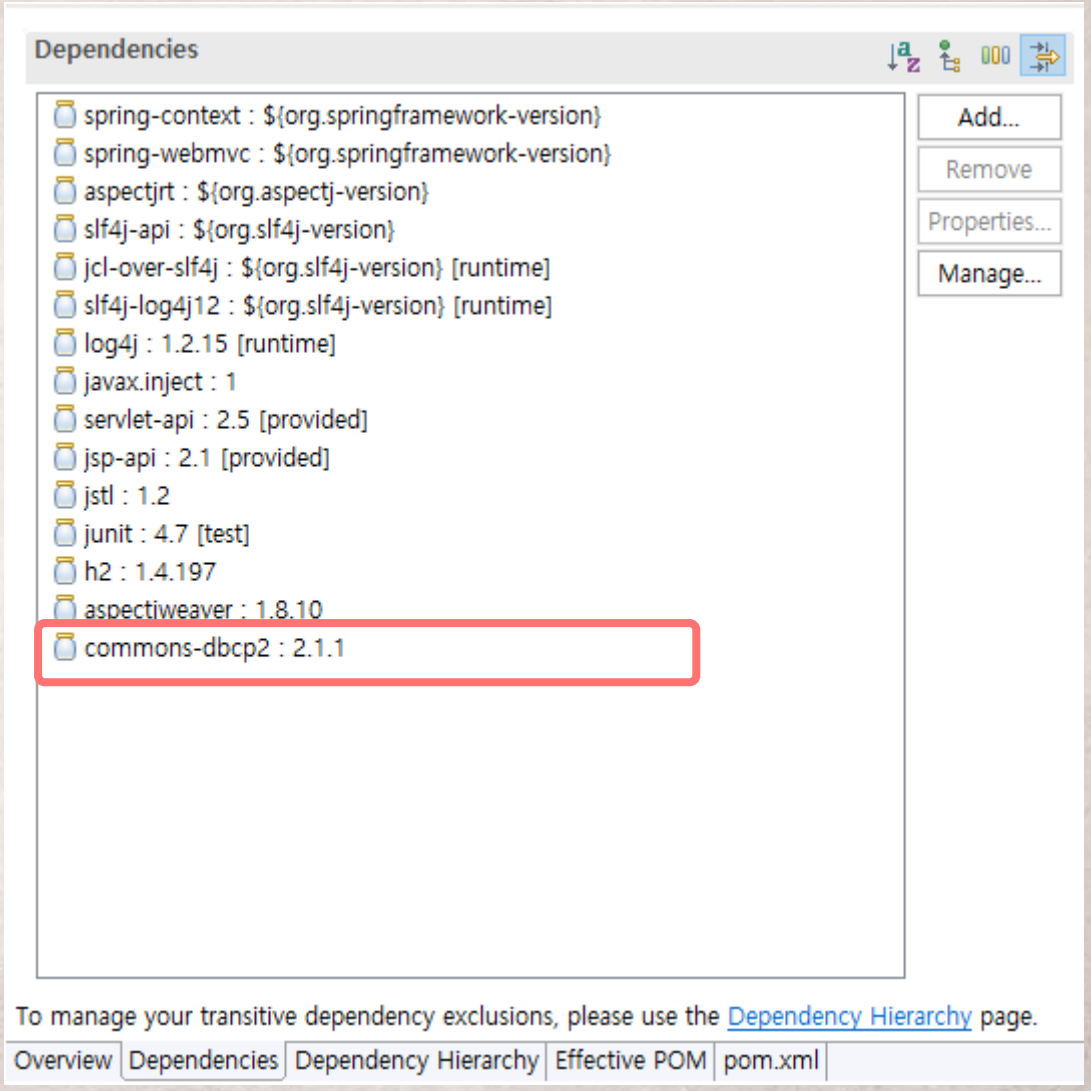


DataSource Dependencies 추가(pom.xml)



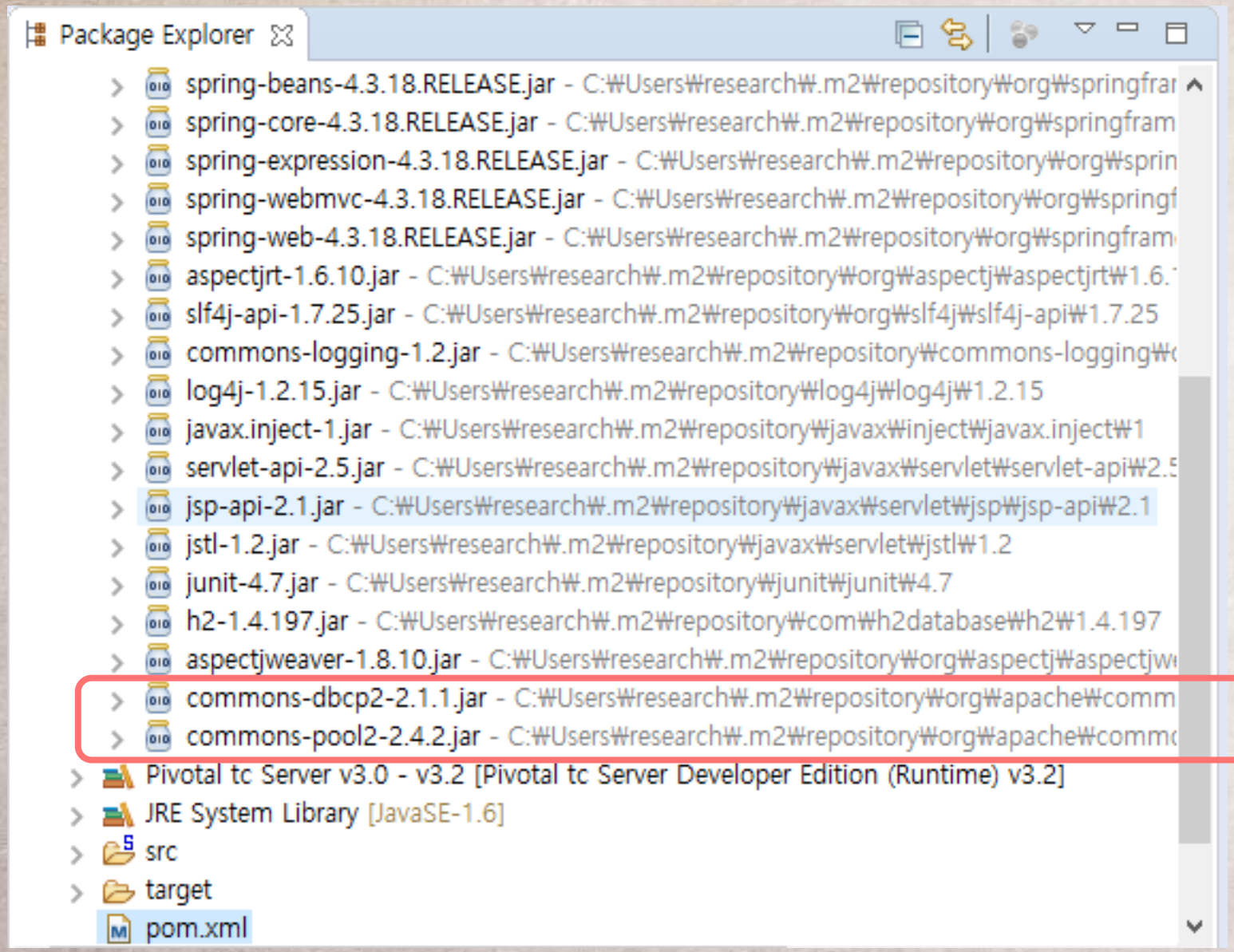
DataSource Dependencies 추가(pom.xml)





pom.xml 파일이 수정되었으므로 저장해야 함.

DataSource – apache의 commons-dbcp2



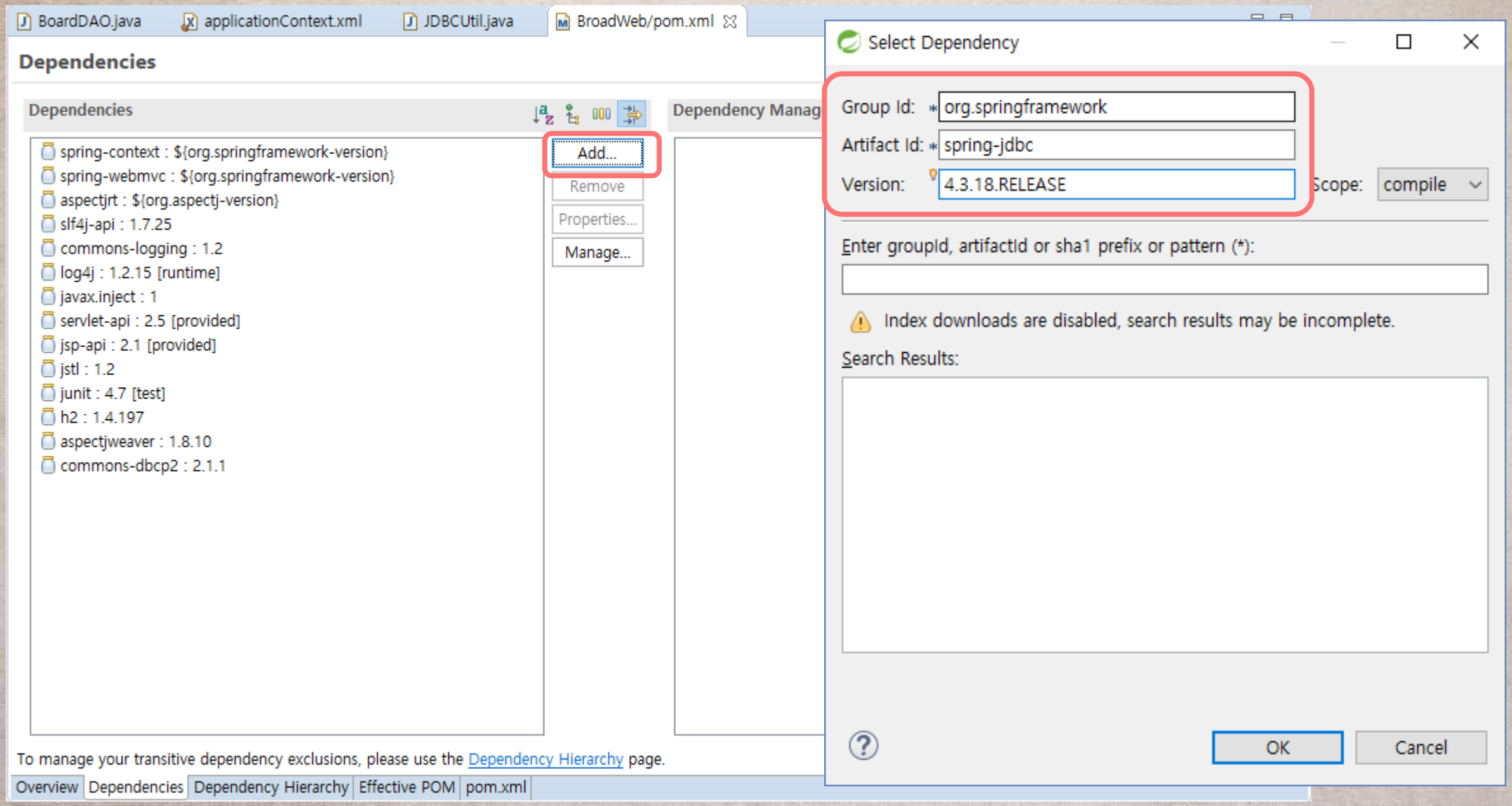
Bean 등록을 통한 DataSource 추가 설정

BoardDAO.java | applicationContext.xml | JDBCUtil.java

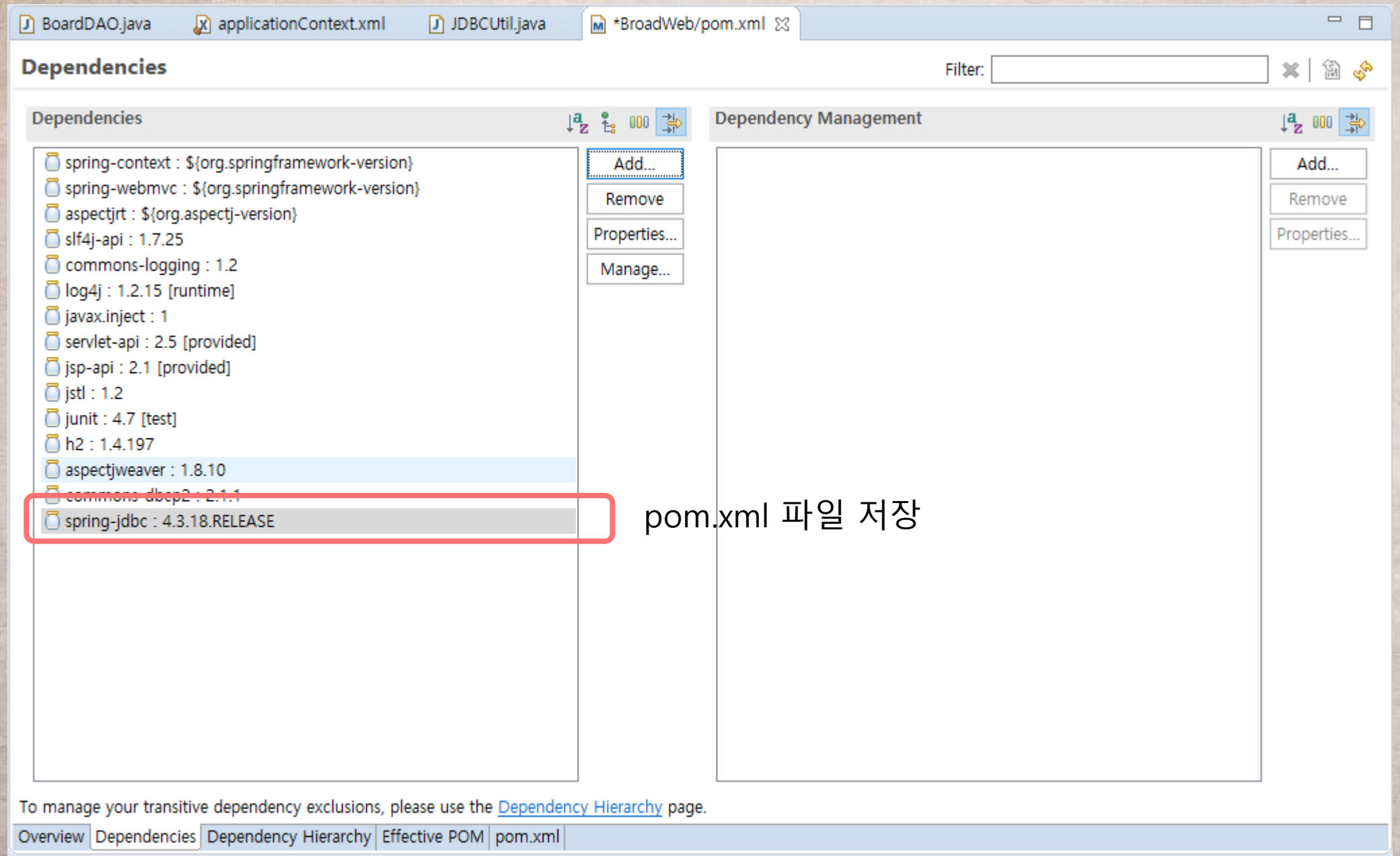
```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <beans xmlns="http://www.springframework.org/schema/beans"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xmlns:p="http://www.springframework.org/schema/p"
5      xmlns:context="http://www.springframework.org/schema/context"
6      xmlns:aop="http://www.springframework.org/schema/aop"
7      xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
8          http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd
9          http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.3.xsd">
10
11     <context:component-scan base-package="kr.ac.inje.comsi"></context:component-scan>
12
13
14     <bean id="dataSource"
15         class="org.apache.commons.dbcp2.BasicDataSource"
16         autowire-candidate="false" destroy-method="close">
17         <property name="driverClassName" value="org.h2.Driver"></property>
18         <property name="url" value="jdbc:h2:D:/workspace/h2db/db"></property>
19         <property name="username" value="sa"></property>
20         <property name="password" value=""></property>
21     </bean>
22 </beans>
23
```

기존 Bean 설정은 제거 한 후 JDBC를 이용한 DataSource 설정 추가됨

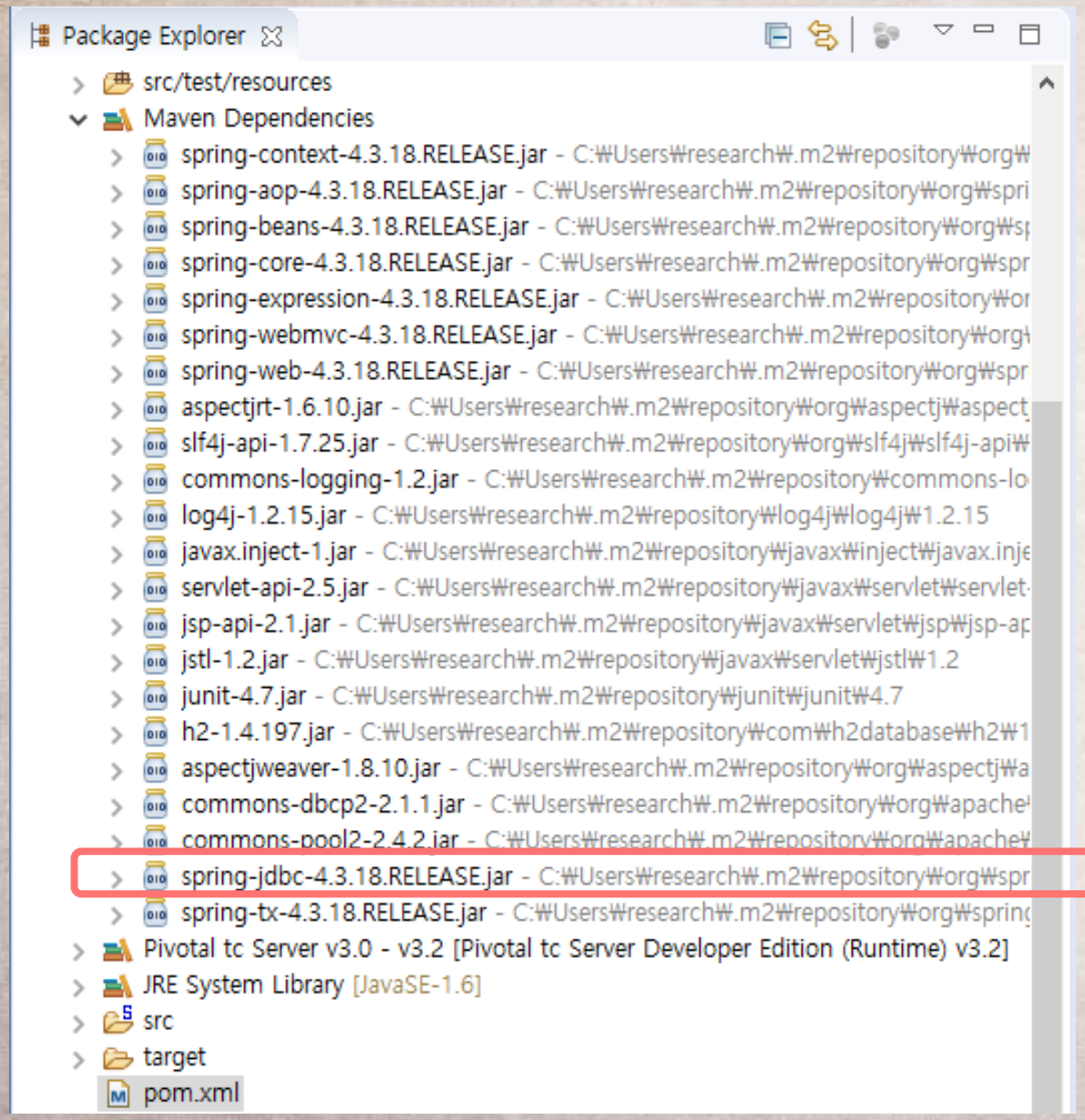
Spring JDBC API 추가하기



Spring JDBC API 추가하기



Spring JDBC API 추가하기



JdbcTamplate 클래스

- JDBC core 패키지에서 가장 중요한 클래스
- 리소스의 생성과 해지를 처리해서 연결을 닫는 걸 까먹는 등의 일반적인 오류를 피하도록 도와준다.
  - 리소스는 ResultSet, Statement(PreparedStatement), Connection 등
- 스테이트먼트(statement)의 생성과 실행같은 JDBC 핵심 작업 흐름의 기본적인 작업을 수행해주므로 어플리케이션 코드가 SQL을 제공하고 결과를 받도록 한다.
- SQL 조회, 업데이트문, 저장 프로시저 호출, ResultSet의 반복 수행, 반환된 파라미터 값의 추출 등을 실행한다.

JdbcTemplate 메소드

- **update() 메소드**: "?"에 값을 설정하는 방식에 따라 두 가지 형태
  - 첫번째: SQL 구문에 설정된 "?"수만큼 값들을 나열하는 방식

메소드	int update(String sql, Object ... args)

사용 예	// 글 수정
	public void updateBoard(BoardVO vo) {
	String BOARD_UPDATE = "update board set title=?, content=? where seq=?";
	int cnt = jdbcTemplate.update(BOARD_UPDATE, vo.getTitle(), vo.getContent(), vo.getSeq());
	System.out.println(cnt + "건 데이터 수정");
	}

  - 두번째: Object 배열 객체에 SQL 구문에 설정된 "?" 수만큼의 값들을 세팅하여 배열 객체를 두번째 인자로 전달하는 방식

메소드	int update(String sql, Object ... args)

사용 예	// 글 수정
	public void updateBoard(BoardVO vo) {
	String BOARD_UPDATE = "update board set title=?, content=? where seq=?";
	Object[] args = {vo.getTitle(), vo.getContent(), vo.getSeq() };
	int cnt = jdbcTemplate.update(BOARD_UPDATE,args);
	System.out.println(cnt + "건 데이터 수정");
	}

- **queryForInt()**메소드
  - SELECT 구문으로 검색된 정숫값을 받을 때 사용하며 매개변수의 의미는 앞에서 본 update()메소드와 같다.

메소드	**int queryForInt(String sql)** **int queryForInt(String sql, Object... args)** **int queryForInt(String sql, Object[] args)**

사용 예	// 전체 게시글 수 조회
	public int getBoardTotalCount(BoardVO vo) {
	String BOARD_TOT_COUNT = "select count(*) from board";
	
	int cnt = jdbcTemplate.queryForInt(BOARD_TOT_COUNT);
	System.out.println("전체 게시글 수: "+ cnt + "건");
	}

- **queryForObject()**메소드
  - SELECT 구문의 실행 결과를 특정 자바 객체(Value Object)로 매핑하여 리턴 받을 때 사용한다.
  - 검색 결과가 없거나 검색 결과가 두 개 이상이면 예외(IncorrectResultSizeDataAccessException)를 발생시킨다.
  - 중요한 것은 검색 결과를 자바 객체(Value Object)로 매핑한 **RowMapper 객체**를 지정해야 한다.

메소드	**int queryForObject(String sql)** **int queryForObject(String sql, RowMapper<T> rowMapper)** **int queryForObject(String sql, Object[] args, RowMapper<T> rowMapper)**

사용 예	// 글 상세 조회
	public BoardVO getBoard(BoardVO vo) {
	String BOARD_GET = "select * from board where seq=?";
	Object[] args = {vo.getSeq() };
	return jdbcTemplate.queryForObject(BOARD_GET, args, new BoardRowMapper());
	}

➔ 자세한 내용은 JdbcTemplate의 Javadoc을 참고해라.

RowMapper 객체

- 검색 결과를 특정 자바 VO(Value Object) 객체에 매핑하여 리턴하려면 RowMapper 인터페이스를 구현한 RowMapper 클래스가 필요

  ➔RowMapper 클래스는 테이블당 하나씩 필요함

- RowMapper 인터페이스는 mapRow() 메소드가 있어서 검색 결과로 얻어낸 Row 정보를 어떤 VO에 어떻게 매핑할 것인지 구현하면 된다.

SELECT * FROM BOARD;

SEQ	TITLE	WRITER	CONTENT	REGDATE	CNT
1	가입인사	관리자	잘 부탁드립니다....	2018-04-08	0
2	임시 제목	홍길동	임시 내용	2018-05-12	0

(2 행, 5 ms)

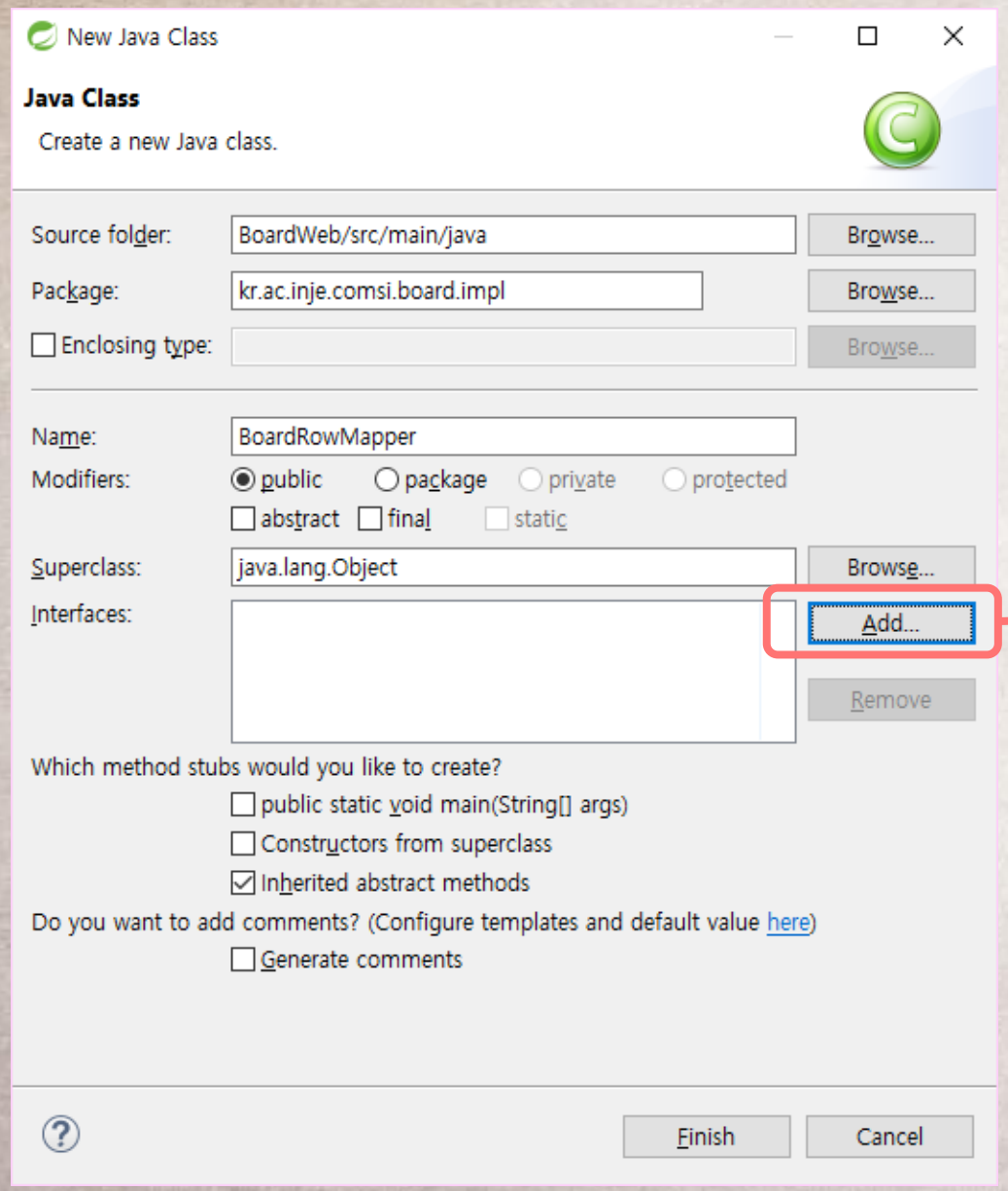
→ BoardVO 객체로 매핑

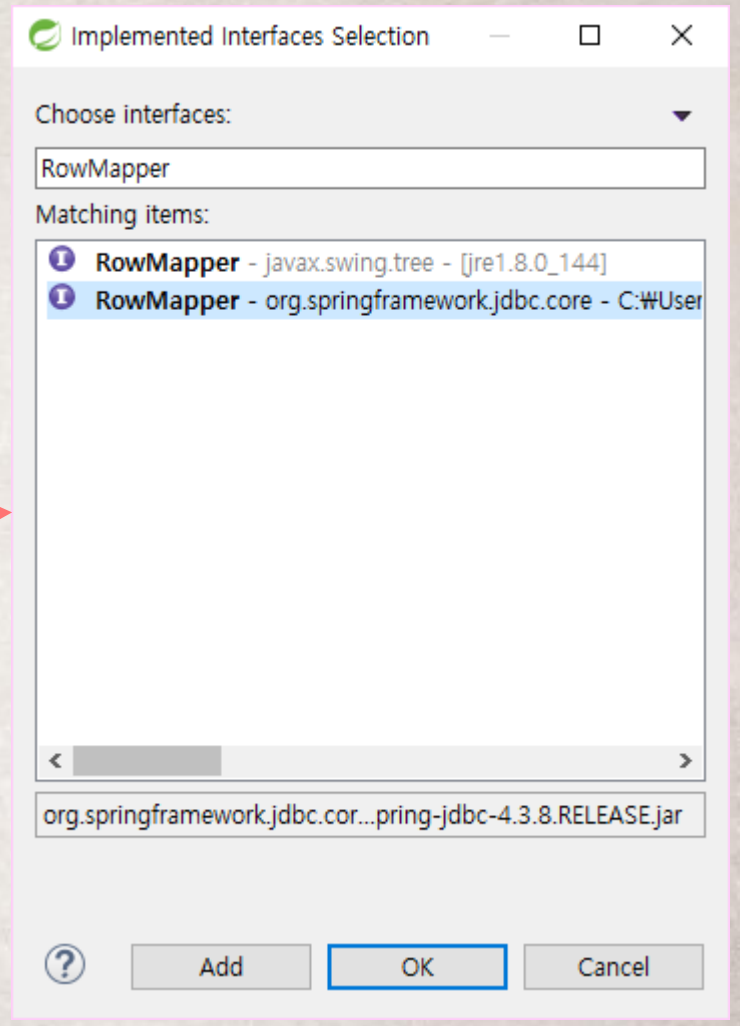
완성된 RowMapper 클래스 – BoardRowMapper 클래스

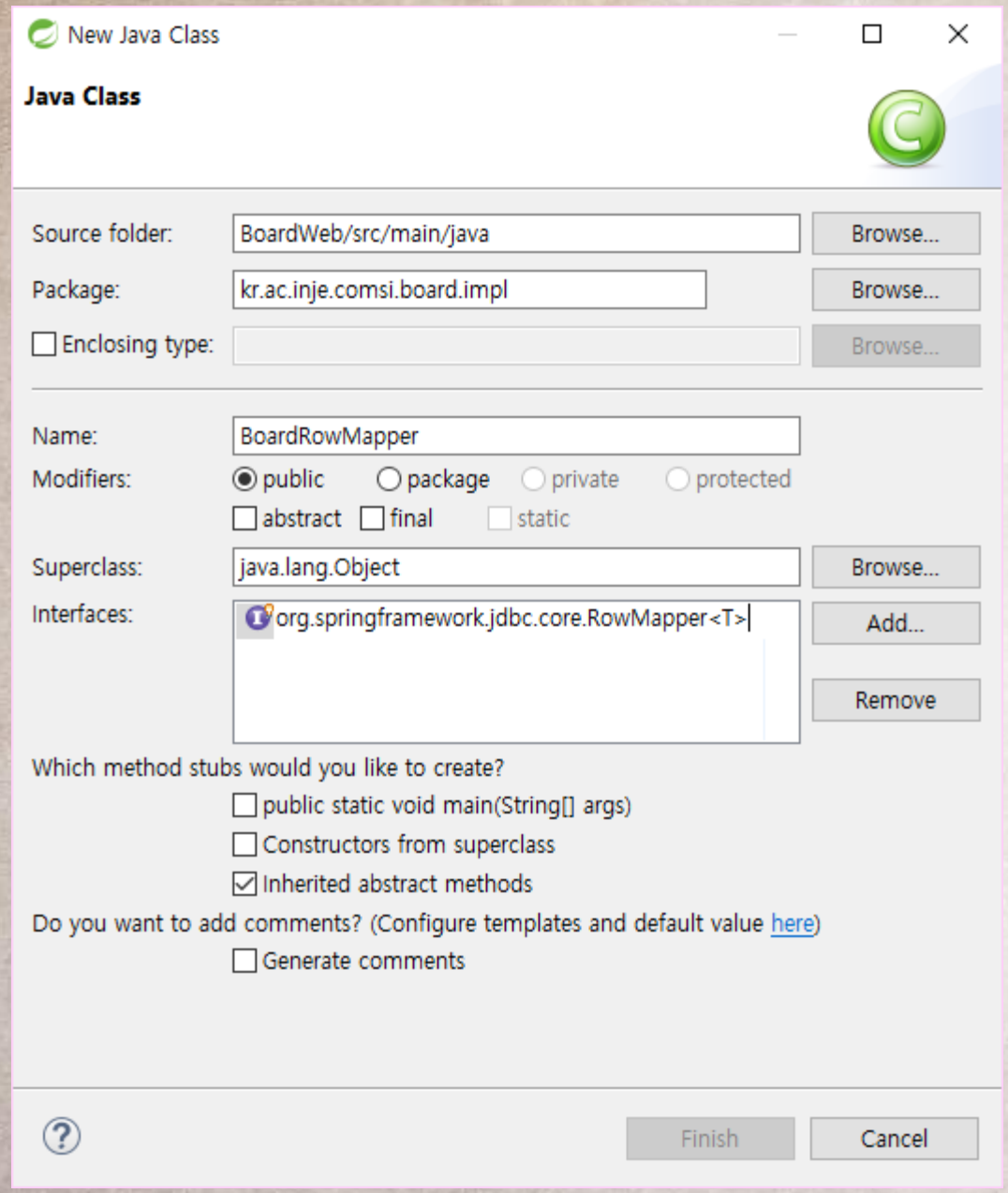
applicationC... | UserDAO.java | build.gradle | BoardServic... | BoardServic... | BoardDAO.java | BoardRowMap...

```java
package kr.ac.inje.comsi.board.impl;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;

import kr.ac.inje.comsi.board.BoardVO;

public class BoardRowMapper implements RowMapper<BoardVO> {

    @Override
    public BoardVO mapRow(ResultSet rs, int rowNum) throws SQLException {

        BoardVO board = new BoardVO();
        board.setSeq(rs.getInt("SEQ"));
        board.setTitle(rs.getString("TITLE"));
        board.setWriter(rs.getString("WRITER"));
        board.setContent(rs.getString("CONTENT"));
        board.setRegDate(rs.getDate("REGDATE"));
        board.setCnt(rs.getInt("CNT"));

        return board;
    }

}
```

RowMapper<T> 인터페이스 구현 클래스 만들기







- 제네릭
  - 특정 타입만 다루지 않고, 여러 종류의 타입으로 변신할 수 있도록 클래스나 메소드를 일반화시키는 기법
    - <E>, <K>, <V> : 타입 매개 변수
      - 요소 타입을 일반화한 타입
  - 제네릭 클래스 사례
    - 제네릭 벡터 : Vector<E>
    - E에 특정 타입으로 구체화
    - 정수만 다루는 벡터 Vector<Integer>
    - 문자열만 다루는 벡터 Vector<String>

RowMapper 인터페이스를 구현한 BoardRowMapper

applicationC... | UserDAO.java | build.gradle | BoardServic... | BoardServic... | BoardDAO.java | *BoardRowMa...

```
1  package kr.ac.inje.comsi.board.impl;
2
3  import org.springframework.jdbc.core.RowMapper;
4
5  public class BoardRowMapper implements RowMapper<T> {
6
7  }
8
```

applicationContext.xml | UserServiceClient.java | AfterThrowingAdvice.java | UserServiceImpl.java | *BoardRowMapper.java

```
1  package kr.ac.inje.comsi.board.impl;
2
3  import org.springframework.jdbc.core.RowMapper;
4
5  public class BoardRowMapper implements RowMapper<BoardVO> {
6
7  }
8
```

The type BoardRowMapper must implement the inherited abstract method RowMapper<BoardVO>.mapRow(ResultSet, int)

2 quick fixes available:

- Add unimplemented methods
- Make type 'BoardRowMapper' abstract

Press 'F2' for focus

mapRow() 함수 override하기

mapRow() 메소드 implements

applicationC... | UserDAO.java | build.gradle | BoardServic... | BoardServic... | BoardDAO.java | *BoardRowMa...

```java
package kr.ac.inje.comsi.board.impl;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;

public class BoardRowMapper implements RowMapper<BoardVO> {

    @Override
    public BoardVO mapRow(ResultSet rs, int rowNum) throws SQLException {
        // TODO Auto-generated method stub
        return null;
    }

}
```

ResultSet 결과로부터 BoardVO 객체로 매핑

완성된 RowMapper 클래스 – BoardRowMapper 클래스

applicationC... | UserDAO.java | build.gradle | BoardServic... | BoardServic... | BoardDAO.java | BoardRowMap...

```java
package kr.ac.inje.comsi.board.impl;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;

import kr.ac.inje.comsi.board.BoardVO;

public class BoardRowMapper implements RowMapper<BoardVO> {

    @Override
    public BoardVO mapRow(ResultSet rs, int rowNum) throws SQLException {

        BoardVO board = new BoardVO();
        board.setSeq(rs.getInt("SEQ"));
        board.setTitle(rs.getString("TITLE"));
        board.setWriter(rs.getString("WRITER"));
        board.setContent(rs.getString("CONTENT"));
        board.setRegDate(rs.getDate("REGDATE"));
        board.setCnt(rs.getInt("CNT"));

        return board;
    }

}
```

- **query 메소드**
  - queryForObject()가 SELECT문으로 객체 하나를 검색할 때 사용하는 메소드라면 query() 메소드는 SELECT문의 실행 결과가 목록일 때 사용한다.
  - query() 메소드가 실행되면 여러 건의 row 정보가 검색되며,row 수 만큼 RowMapper 객체의 mapRow() 메소드가 실행

메소드	**int query(String sql)** **int query(String sql, RowMapper<T> rowMapper)** **int query(String sql, Object[] args, RowMapper<T> rowMapper)**

사용 예	// 글 상세 조회
	public List<BoardVO> getBoardList(BoardVO vo) {
	String BOARD_LIST = "select * from board order by seq desc";
	return jdbcTemplate.query(BOARD_LIST, args, new BoardRowMapper());
	}

  - row 정보가 매핑된 VO 객체 여러 개가 List 컬렉션에 저장되어 리턴된다.

- 요소(element)라고 불리는 가변 개수의 객체들의 모음
  - 요소의 개수에 따라 컬렉션은 자동 크기 조절
- 다양한 객체들의 삽입, 삭제, 검색 등을 관리하기 용이

JdbcTemplate를 이용한 DAO

- 스프링 JDBC 사용을 위한 설정이 마무리되었고
- JdbcTemplate를 이용한 DAO 클래스 구현(2가지 방법)
  - jdbcDaoSupport 클래스를 상속
  - **jdbcTemplate 클래스를 <bean> 등록, 의존성 주입**

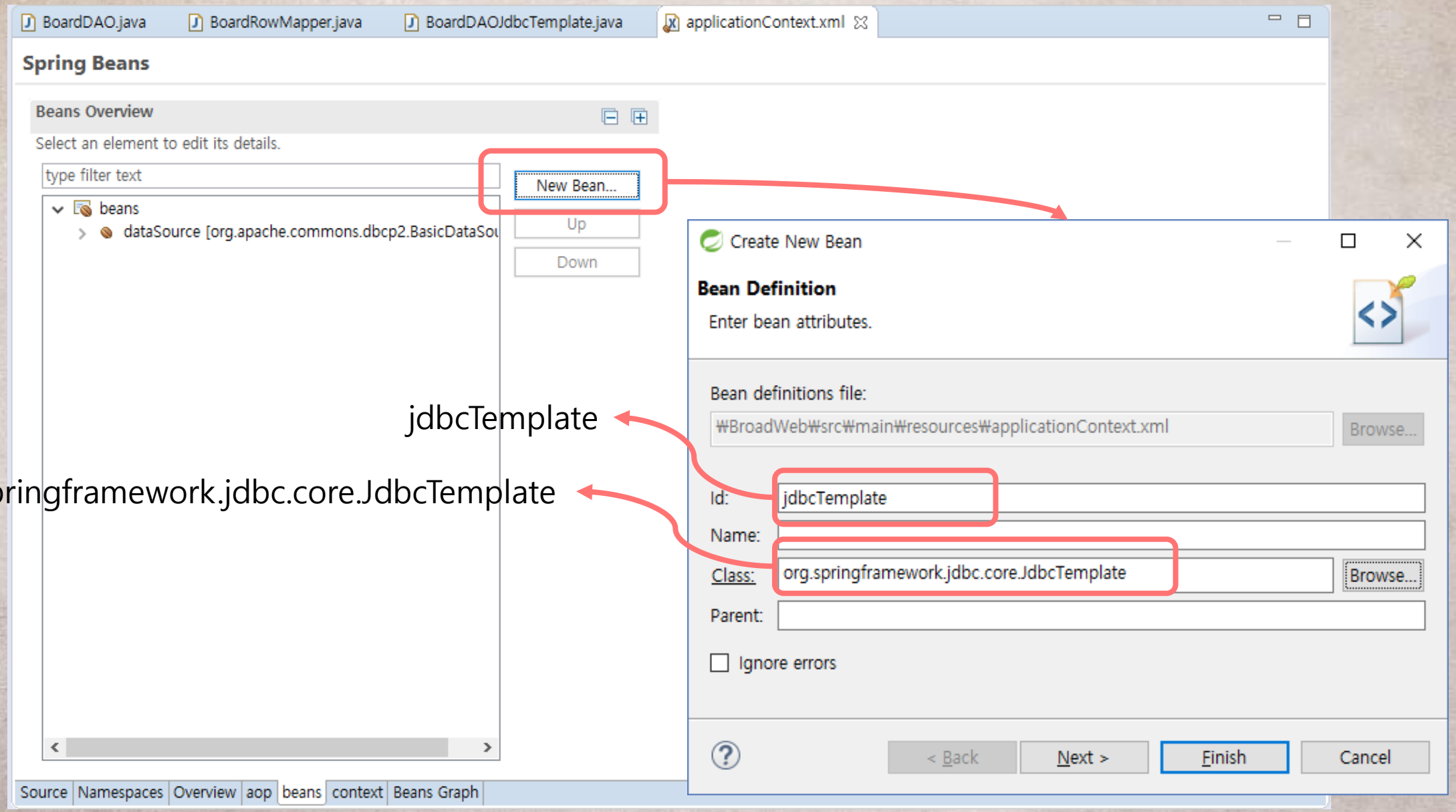
JdbcTemplate 클래스를 <bean> 등록 및 의존성 주입

applicationContext.xml | BoardRowMapper.java

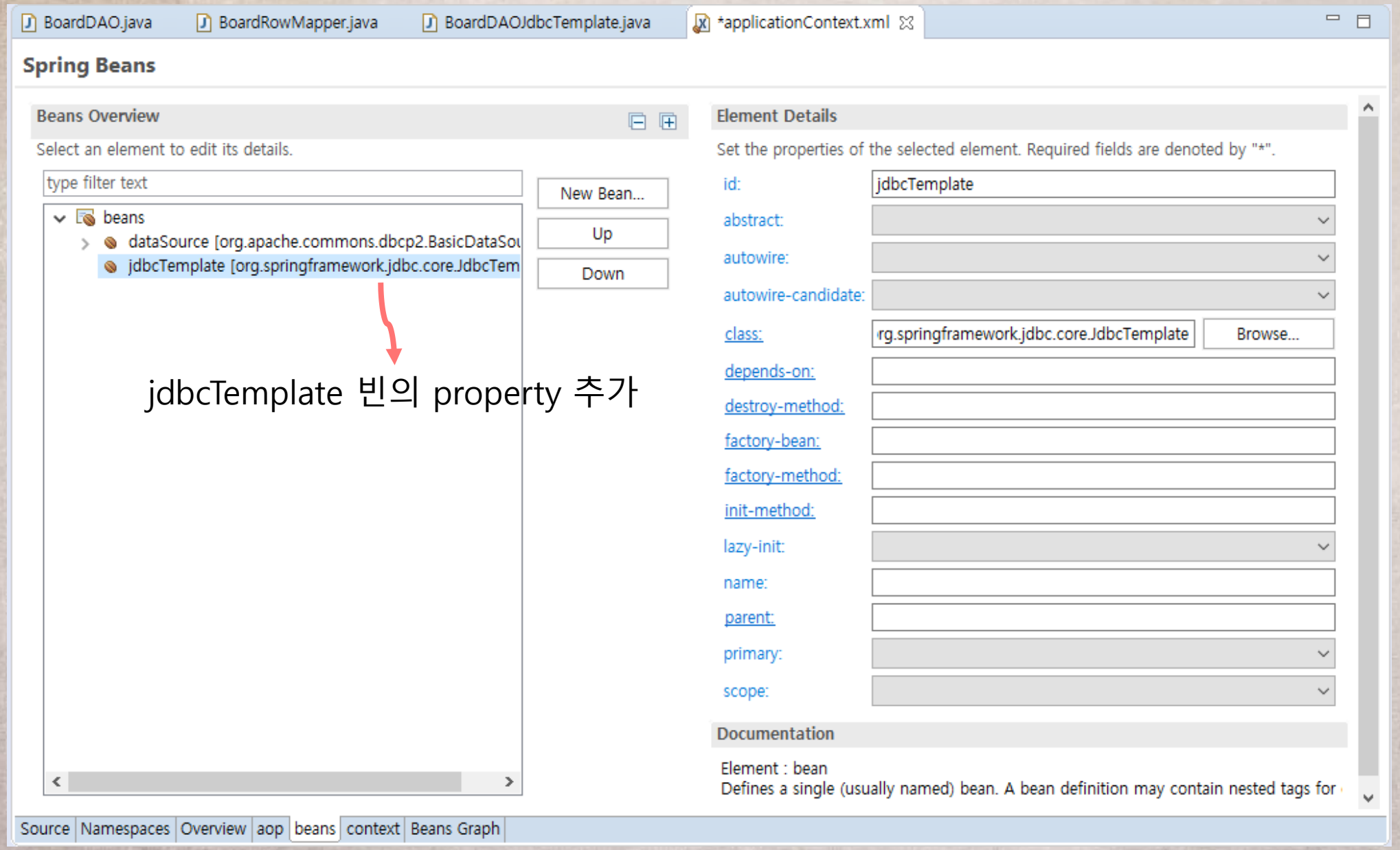
```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:p="http://www.springframework.org/schema/p"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-bean
        http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd
        http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.3.xsd">

    <context:component-scan base-package="kr.ac.inje.comsi"></context:component-scan>

    <!-- DataSouce 설정 -->
    <bean id="dataSource" class="org.apache.commons.dbcp2.BasicDataSource" destroy-method="close">
        <property name="driverClassName" value="org.h2.Driver"/>
        <property name="url" value="jdbc:h2:/d:/workspace/java/h2db/myDB"/>
        <property name="username" value="sa"/>
        <property name="password" value=""/>
    </bean>
    <!-- Spring JDBC 설정 -->
    <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource" ref="dataSource"/>
    </bean>
</beans>
```

추가시킴
(다음 페이지)

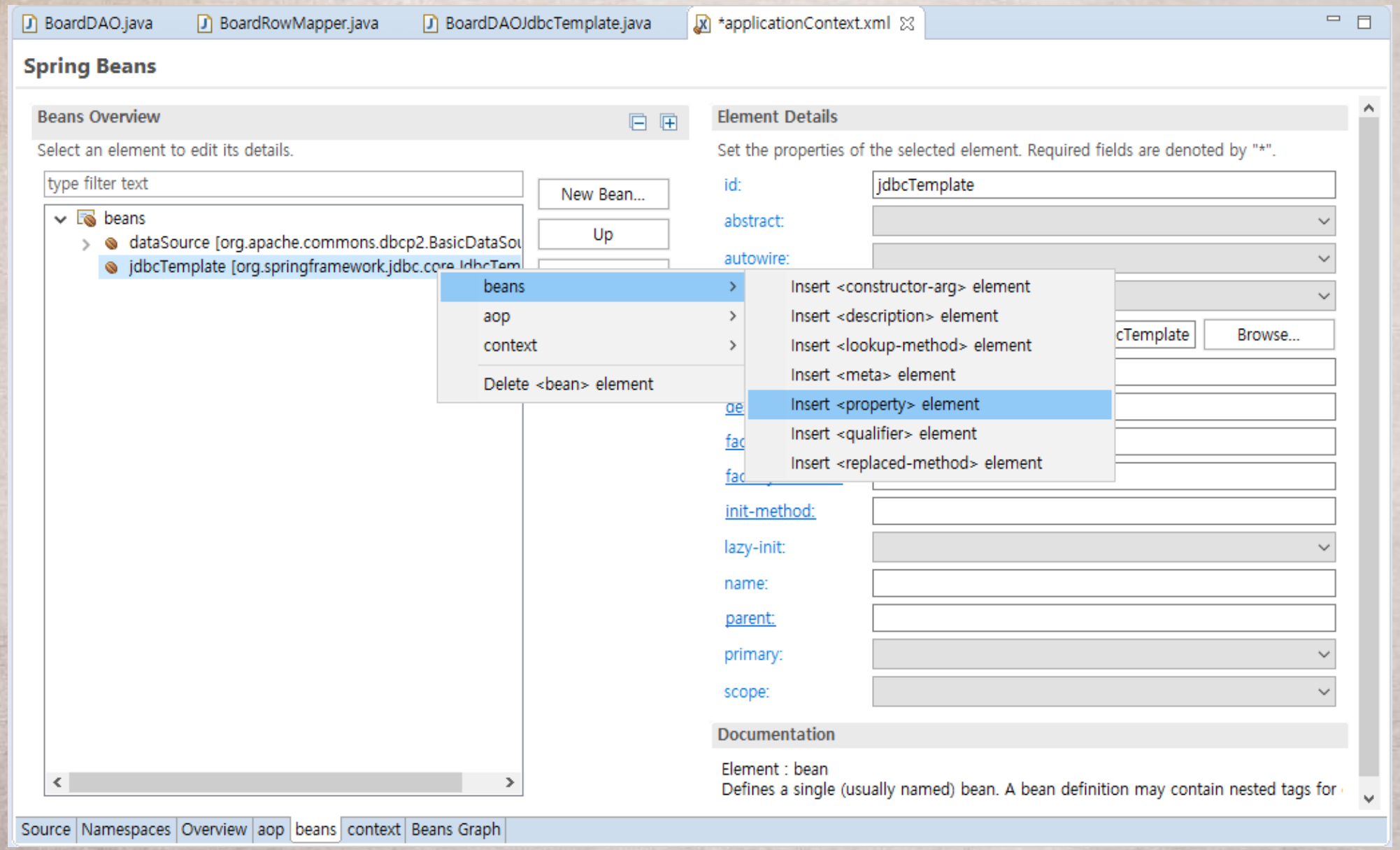
applicationContext.xml 수정



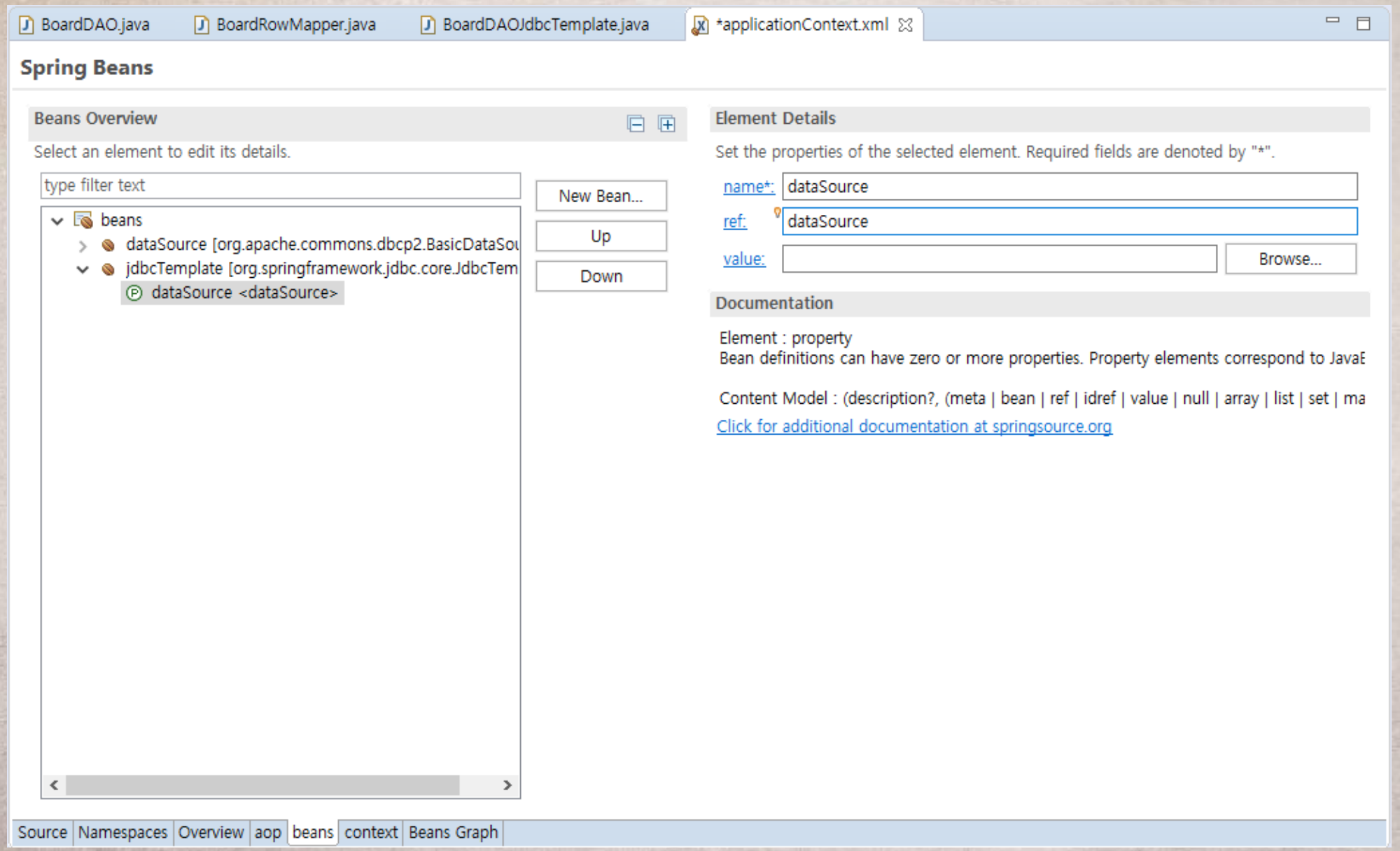
applicationContext.xml 수정



applicationContext.xml 수정

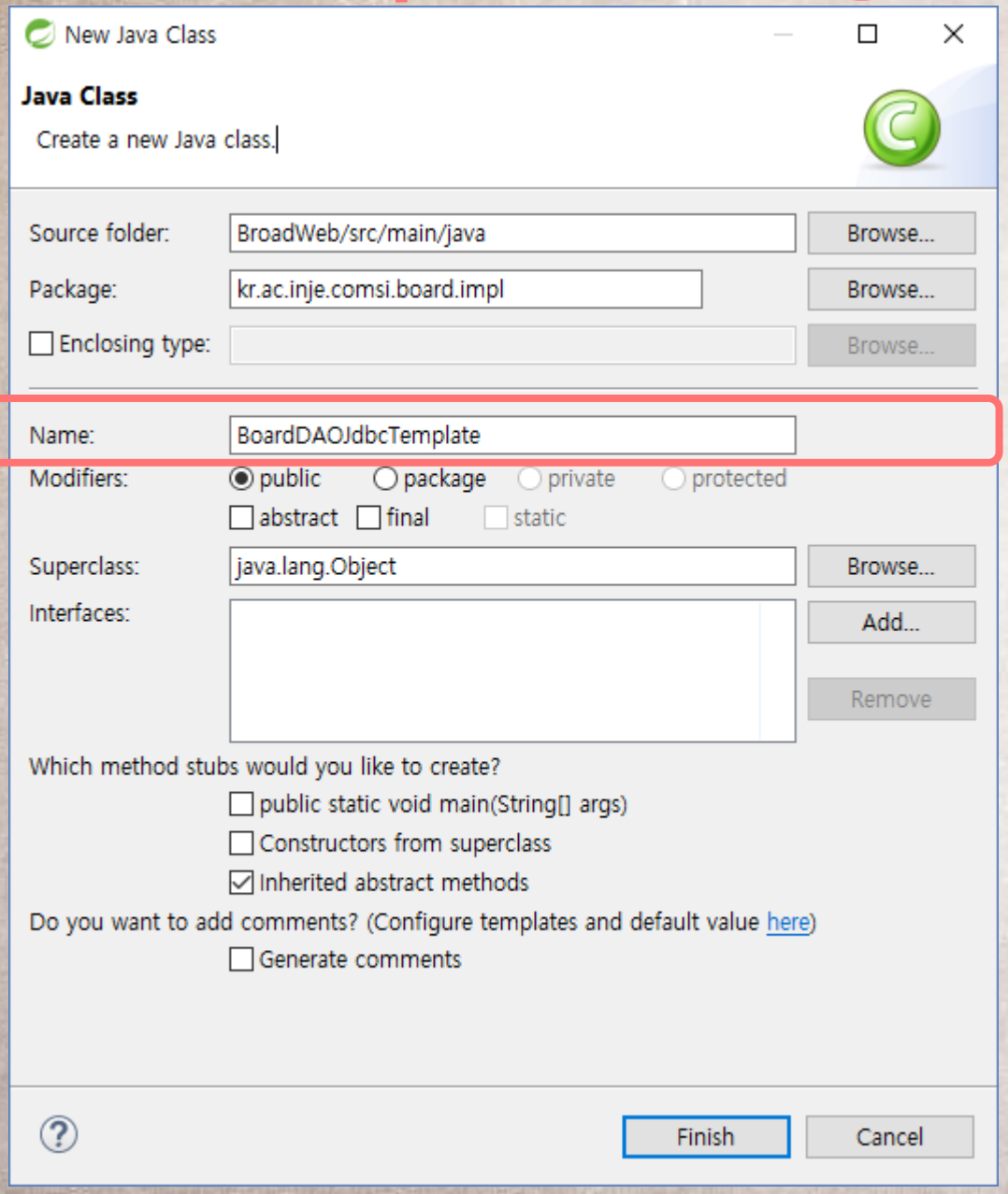


applicationContext.xml 수정



완료

JdbcTemplate를 이용한 DAO 클래스 만들기



클래스 이름-BoardDAOJdbcTemplate

BoardDAO클래스를 대체할 BoardDAOJdbcTemplate 추가

BoardDAO.java | BoardRowMapper.java | BoardDAOJdbcTemplate.java

```java
package kr.ac.inje.comsi.board.impl;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Repository;

import kr.ac.inje.comsi.board.BoardVO;

@Repository("boardDAOJdbcTemplate")
public class BoardDAOJdbcTemplate {

    @Autowired
    private JdbcTemplate jdbcTemplate;

    // SQL 명령어
    private final String BOARD_INSERT = "insert into board(seq, title, writer, "
            + "content) values((select nvl(max(seq),0)+1 from board),?,?,?)";
    private final String BOARD_UPDATE = "update board set title=?, content=? where seq=?";
    private final String BOARD_DELETE = "delete board where seq=?";
    private final String BOARD_GET = "select * from board where seq=?";
    private final String BOARD_LIST = "select * from board order by seq desc";

    // CRUD 기능의 메소드 구현
    // 글 등록
    public void insertBoard(BoardVO vo){
        System.out.println("===> JDBC로 insertBoard() 기능 처리");
        jdbcTemplate.update(BOARD_INSERT, vo.getTitle(), vo.getWriter(), vo.getContent());
    }

```

applicationContext.xml에 등록된
id="jdbcTemplate"인 bean

```
32      // 글 수정
33      public void updateBoard(BoardVO vo){
34          System.out.println("===> JDBC로 updateBoard() 기능 처리");
35          jdbcTemplate.update(BOARD_UPDATE, vo.getTitle(), vo.getContent(), vo.getSeq());
36      }
37
38      // 글 삭제
39      public void deleteBoard(BoardVO vo){
40          System.out.println("===> JDBC로 deleteBoard() 기능 처리");
41          jdbcTemplate.update(BOARD_DELETE, vo.getSeq());
42      }
43
44      // 글 상세 조회
45      public BoardVO getBoard(BoardVO vo){
46          System.out.println("===> JDBC로 getBoard() 기능 처리");
47          Object[] args = {vo.getSeq()};
48          return jdbcTemplate.queryForObject(BOARD_GET, args, new BoardRowMapper());
49      }
50
51      // 글 목록 조회
52      public List<BoardVO> getBoardList(BoardVO vo){
53          System.out.println("===> JDBC로 getBoardList() 기능 처리");
54          return jdbcTemplate.query(BOARD_LIST, new BoardRowMapper());
55      }
56
57  }
```

BoardServiceImpl 클래스 수정

BoardDAOJdbcTemplate.java | BoardServiceImpl.java

```
1  package kr.ac.inje.comsi.board.impl;
2
3  import java.util.List;
4
5  import org.springframework.beans.factory.annotation.Autowired;
6  import org.springframework.stereotype.Service;
7
8  import kr.ac.inje.comsi.board.BoardService;
9  import kr.ac.inje.comsi.board.BoardVO;
10
11 @Service("boardService")
12 public class BoardServiceImpl implements BoardService {
13
14 //  @Autowired
15 //  private BoardDAO boardDAO;
16
17     @Autowired
18     private BoardDAOJdbcTemplate boardDAOJdbcTemplate;
19
20     @Override
21     public void insertBoard(BoardVO vo) {
22         //boardDAO.insertBoard(vo);
23         boardDAOJdbcTemplate.insertBoard(vo);
24     }
25
26     @Override
27     public void updatetBoard(BoardVO vo) {
28         //boardDAO.updatetBoard(vo);
29         boardDAOJdbcTemplate.updateBoard(vo);
30     }
```

주석처리

추가하기

모든 boardDAO에서 boardDAOJdbcTemplate로 변경

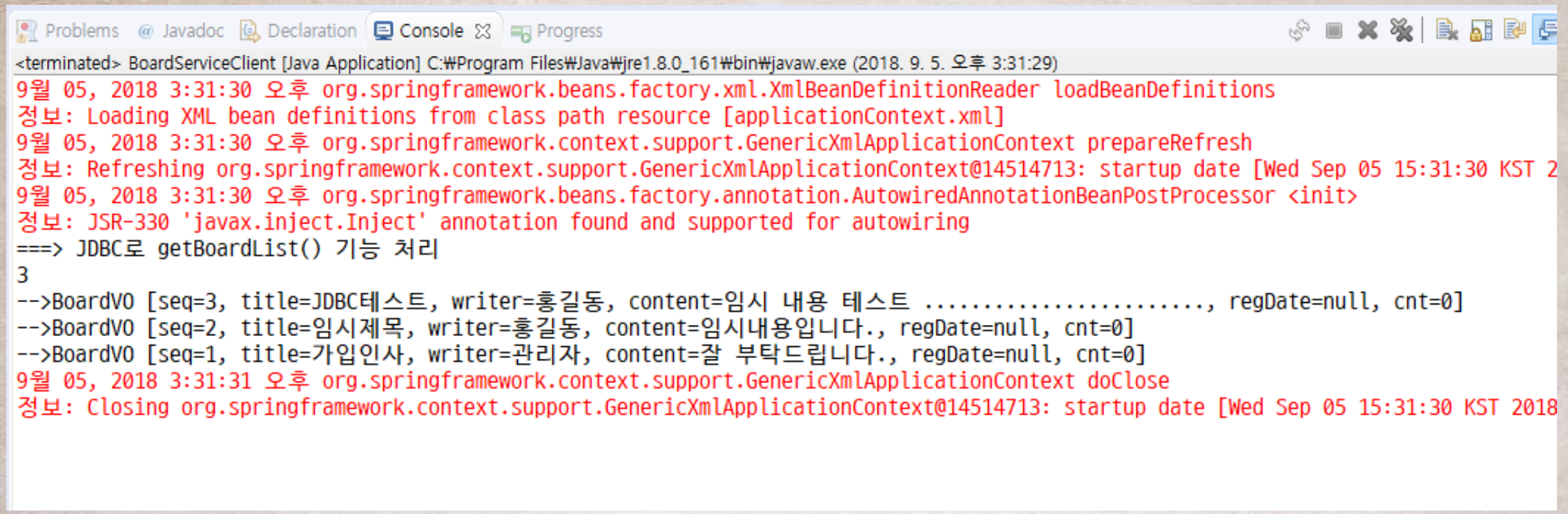
```
31
32      @Override
33      public void deleteBoard(BoardVO vo) {
34          //boardDAO.deleteBoard(vo);
35          boardDAOJdbcTemplate.deleteBoard(vo);
36      }
37
38      @Override
39      public BoardVO getBoard(BoardVO vo) {
40          // return boardDAO.getBoard(vo);
41          return boardDAOJdbcTemplate.getBoard(vo);
42      }
43
44      @Override
45      public List<BoardVO> getBoardList(BoardVO vo) {
46          //return boardDAO.getBoardList(vo);
47          return boardDAOJdbcTemplate.getBoardList(vo);
48      }
49
50  }
51
```

BoardServiceClient.java 수정

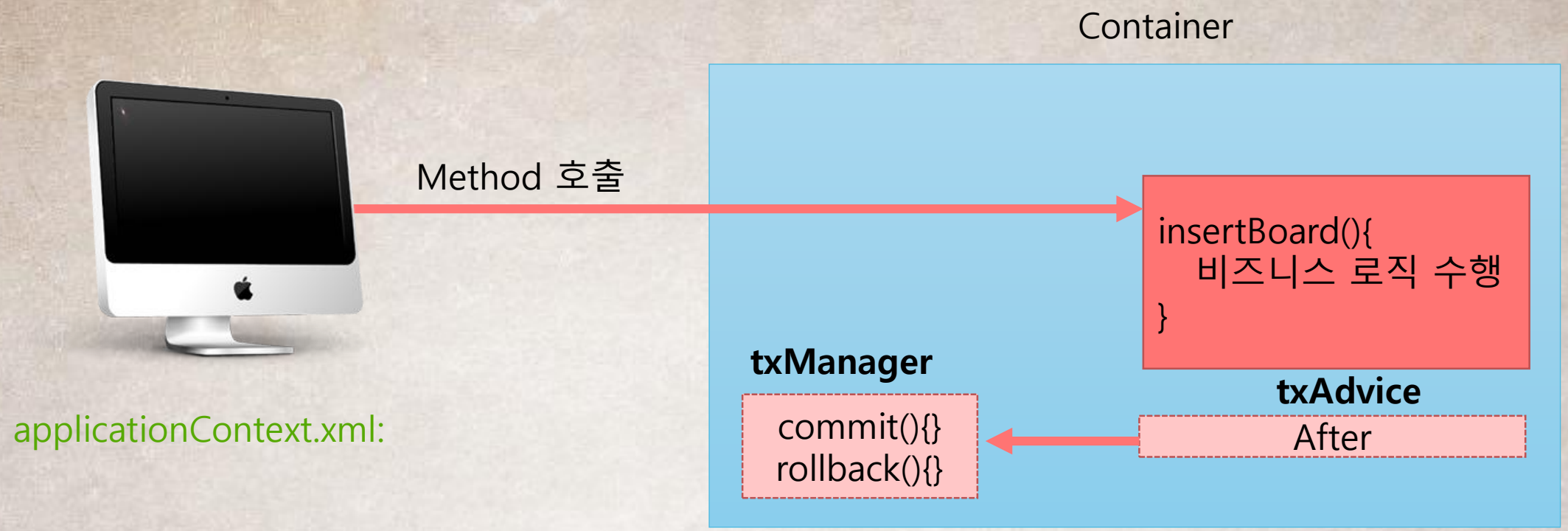
BoardDAOJdbcTemplate.java | BoardServiceImpl.java | BoardServiceClient.java

```java
package kr.ac.inje.comsi.board;

import java.util.List;

import org.springframework.context.support.AbstractApplicationContext;
import org.springframework.context.support.GenericXmlApplicationContext;

public class BoardServiceClient {

    public static void main(String[] args) {

        // 1. Spring 컨테이너 구동
        AbstractApplicationContext container = new GenericXmlApplicationContext("applicationContext.xml");

        // 2. Spring 컨테이너로부터 BoardServiceImpl 객체를 Lookup
        BoardService boardService = (BoardService) container.getBean("boardService");

//      // 3. 글 동록 기능 테스트
        BoardVO vo = new BoardVO();
        vo.setTitle("JDBCTemplate테스트");
        vo.setWriter("홍길동");
        vo.setContent("임시 내용 테스트 ......................");
//      boardService.insertBoard(vo);

        // 4. 글 목록 검색 기능 테스트
        List<BoardVO> boardList = boardService.getBoardList(vo);
        System.out.println(boardList.size());
        for(BoardVO board: boardList) {
            System.out.println("-->" + board.toString());
        }

        // 5. Spring 컨테이너 종료
        container.close();
    }
}
```

실행 결과


Problems
Javadoc
Declaration
Console
Progress
<terminated> BoardServiceClient [Java Application] C:₩Program Files₩Java₩jre1.8.0_161₩bin₩javaw.exe (2018. 9. 5. 오후 3:31:29)
9월 05, 2018 3:31:30 오후 org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
정보: Loading XML bean definitions from class path resource [applicationContext.xml]
9월 05, 2018 3:31:30 오후 org.springframework.context.support.GenericXmlApplicationContext prepareRefresh
정보: Refreshing org.springframework.context.support.GenericXmlApplicationContext@14514713: startup date [Wed Sep 05 15:31:30 KST 2
9월 05, 2018 3:31:30 오후 org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor <init>
정보: JSR-330 'javax.inject.Inject' annotation found and supported for autowiring
===> JDBC로 getBoardList() 기능 처리
3
-->BoardVO [seq=3, title=JDBC테스트, writer=홍길동, content=임시 내용 테스트 ....., regDate=null, cnt=0]
-->BoardVO [seq=2, title=임시제목, writer=홍길동, content=임시내용입니다., regDate=null, cnt=0]
-->BoardVO [seq=1, title=가입인사, writer=관리자, content=잘 부탁드립니다., regDate=null, cnt=0]
9월 05, 2018 3:31:31 오후 org.springframework.context.support.GenericXmlApplicationContext doClose
정보: Closing org.springframework.context.support.GenericXmlApplicationContext@14514713: startup date [Wed Sep 05 15:31:30 KST 2018

7. 트랜잭션 처리



- 클라이언트가 BoardServiceImpl 객체의 insertBoard() 메소드 호출
- insertBoard() 메소드의 비즈니스 로직이 실행
- insertBoard() 메소드 실행 중에 문제가 발생하면
- txAdvce에 등록한 Advice가 동작하며,
  참조하는 txManager의 rollback() 메소드가 호출
- 문제가 없다면
  txManager의 commit() 메소드가 호출

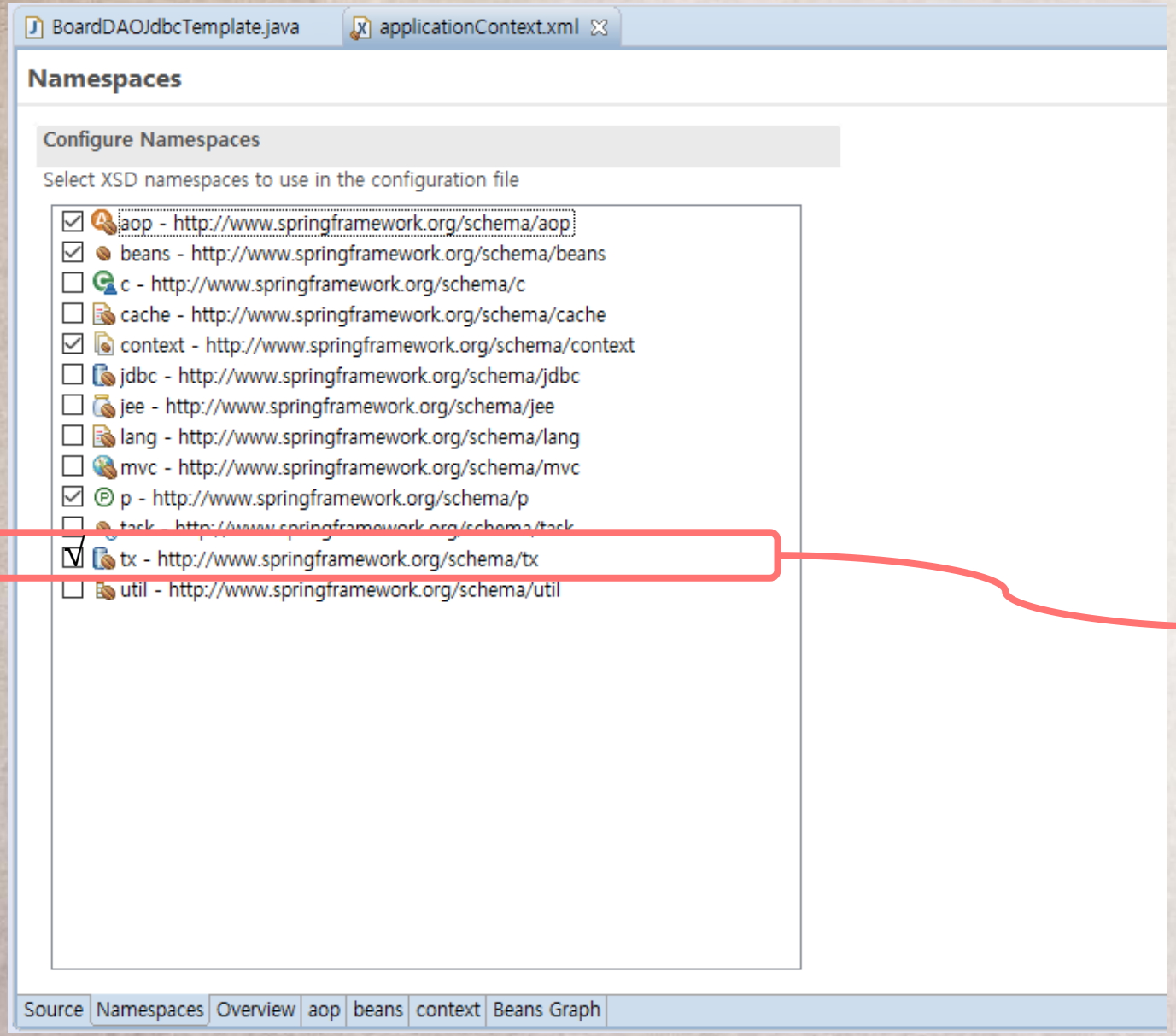
트랜잭션 처리를 위한 namespace와 bean 추가

applicationContext.xml | BoardRowMapper.java | BoardServiceClient.java | BroadWeb/pom.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <beans xmlns="http://www.springframework.org/schema/beans"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xmlns:p="http://www.springframework.org/schema/p"
5      xmlns:context="http://www.springframework.org/schema/context"
6      xmlns:aop="http://www.springframework.org/schema/aop"
7      xmlns:tx="http://www.springframework.org/schema/tx"
8      xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
9          http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd
10         http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.3.xsd
11         http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.3.xsd">
12
13     <context:component-scan base-package="kr.ac.inje.comsi"></context:component-scan>
14
15     <bean id="dataSource" class="org.apache.commons.dbcp2.BasicDataSource" destroy-method="close">
16         <property name="driverClassName" value="org.h2.Driver"></property>
17         <property name="url" value="jdbc:h2:d:/workspace/h2db/db"></property>
18         <property name="username" value="sa"></property>
19         <property name="password" value=""></property>
20     </bean>
21
22     <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
23         <property name="dataSource" ref="dataSource"></property>
24     </bean>
25
26     <bean id="txManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
27         <property name="dataSource" ref="dataSource"></property>
28     </bean>
29 </beans>
30
```

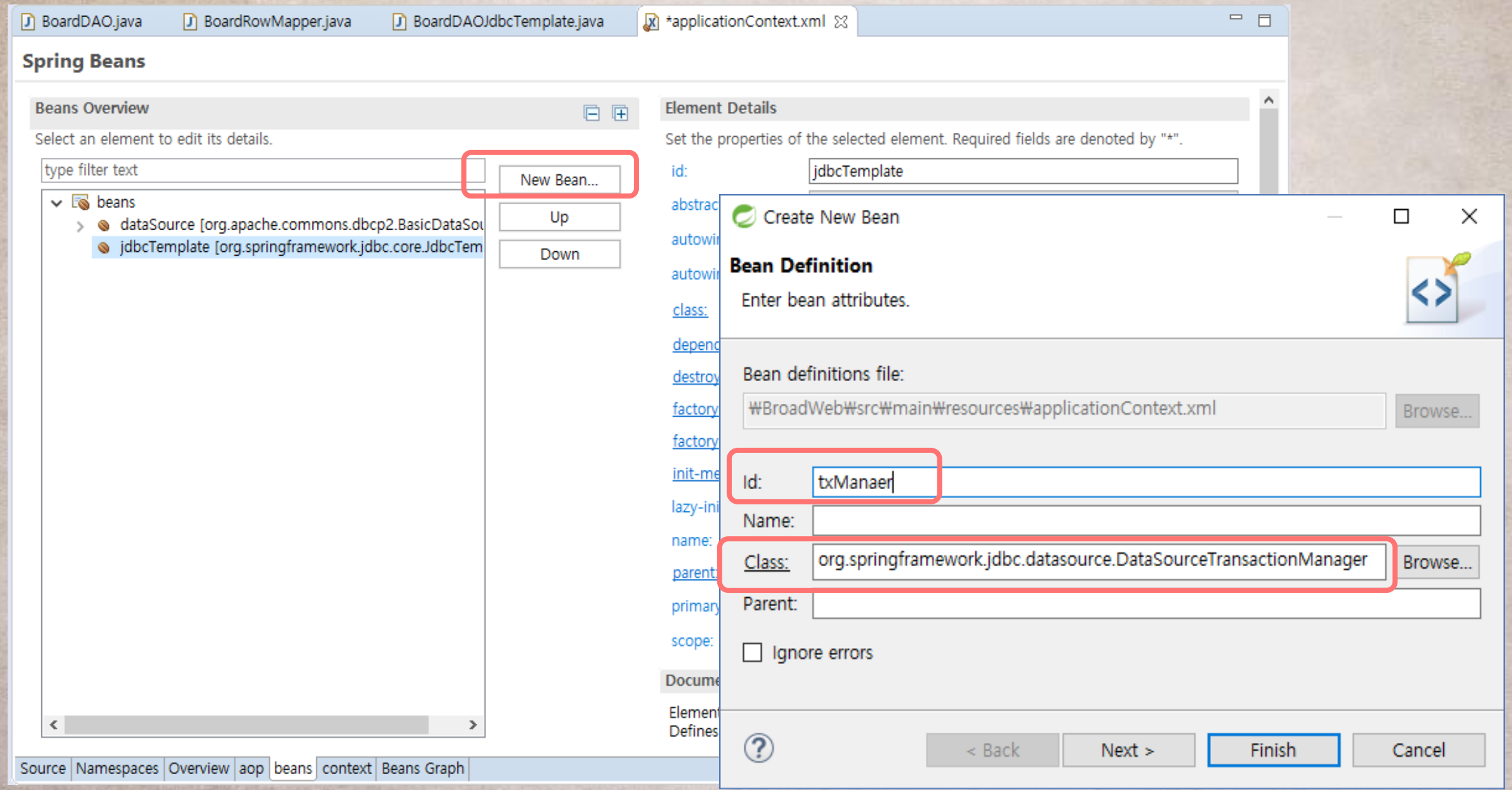
트랜잭션을 위한 Bean

트랜잭션 처리를 위한 namespace

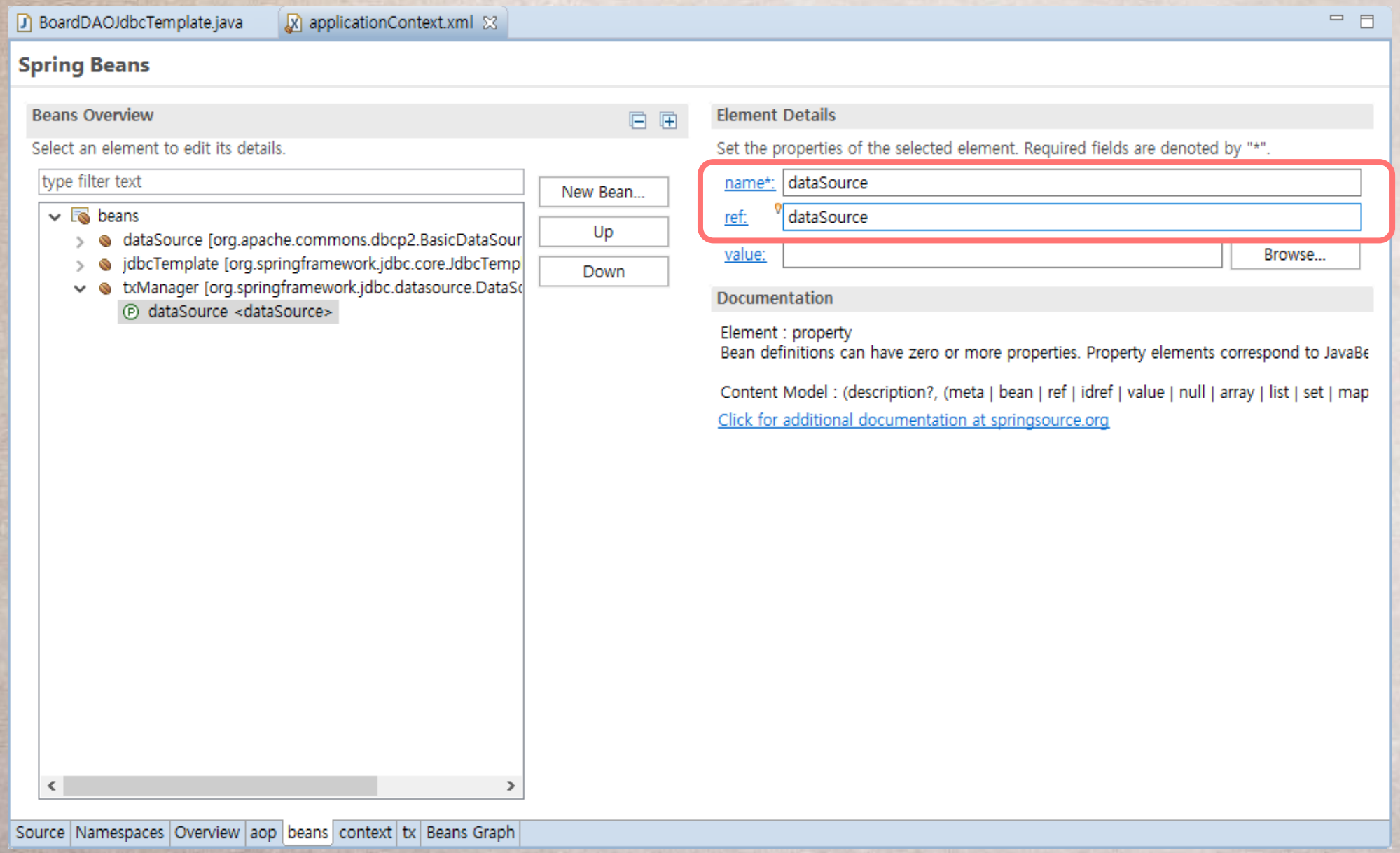


체크함

트랜잭션 처리를 위한 Manager Bean 추가



트랜잭션 처리를 위한 Manager Bean 추가



트랙잭션 Advice 설정 및 AOP 추가

```
26      <bean id="txManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
27          <property name="dataSource" ref="dataSource"></property>
28      </bean>
29
30      <!-- 트랜잭션 Advice 추가 -->
31      <tx:advice id="txAdvice" transaction-manager="txManager">
32          <tx:attributes>
33              <tx:method name="get*" read-only="true" />
34              <tx:method name="*" />
35          </tx:attributes>
36      </tx:advice>
37
```

- txManager를 이용해 트랜잭션을 관리한다.
- <tx:method> 엘리먼트를 이용해 트랜잭션을 적용할 메소드 지정
  - Get으로 시작하는 메소드는 읽기 전용으로 제외
  - 나머지 메소드들은 트랜잭션 관리에 적용

트랙잭션 Advice 설정 및 AOP 추가

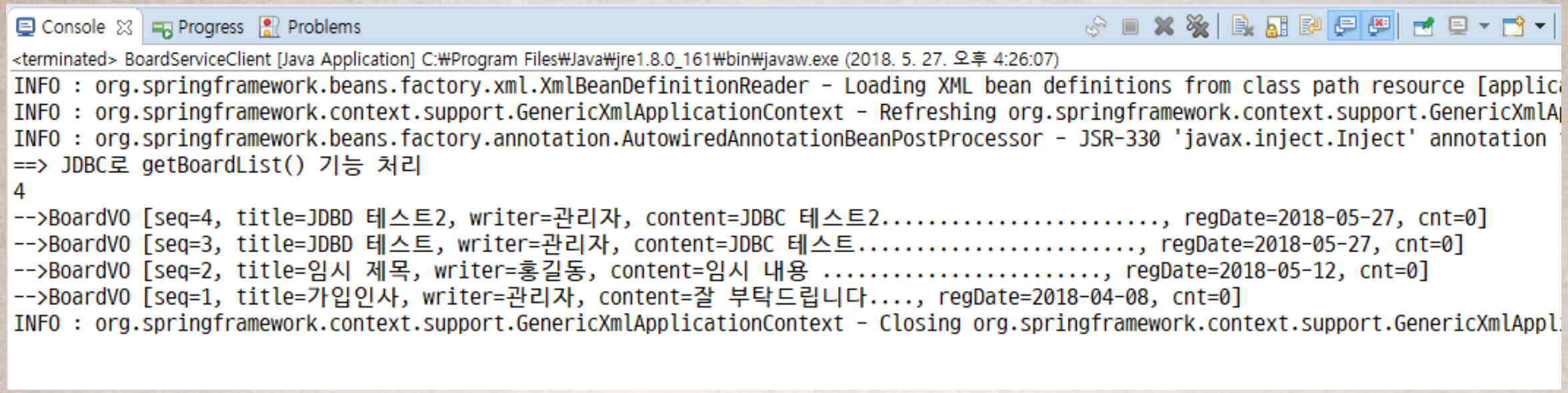
```
26      <bean id="txManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
27          <property name="dataSource" ref="dataSource"></property>
28      </bean>
29
30      <!-- 트랜잭션 Advice 추가 -->
31      <tx:advice id="txAdvice" transaction-manager="txManager">
32          <tx:attributes>
33              <tx:method name="get*" read-only="true" />
34              <tx:method name="*" />
35          </tx:attributes>
36      </tx:advice>
37
38      <aop:config> <!-- 트랜잭션 AOP 추가 -->
39          <aop:pointcut expression="execution(* kr.ac.inje.comsi..*(..))" id="txPointcut" />
40          <aop:advisor advice-ref="txAdvice" pointcut-ref="txPointcut" />
41      </aop:config>
42
43  </beans>
44
```

- 트랙잭션 처리는 txAdvice를 이용하여 처리한다.
  - 트랜잭션 처리 대상은 모든 함수이나 get*으로 시작하는 함수는 read-only로 설정하여 제외
- AOP를 이용하여 횡단관심사를 처리
  - 적용할 대상(pointcut)은 모든 종류의 함수

AOP(Aspect Oriented Programming) - 용어

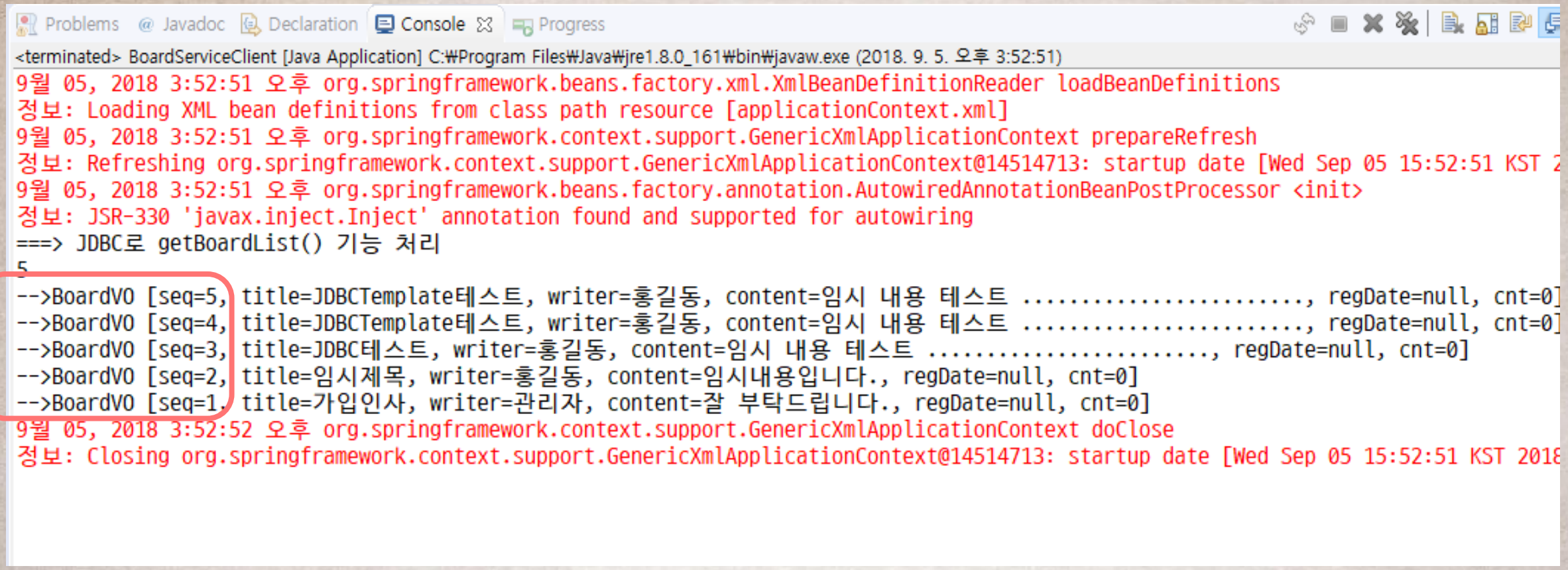
용어	내용
어드바이스 (Advice)	실질적인 비즈니스 로직을 구현하고 있는 "핵심 비즈니스 로직 구현부"에 추가적인 기능을 제공하는 것(횡단관심사)
조인포인트 (Joinpoint)	클래스의 인스턴스 생성 시점, 메소드를 호출하는 시점, Exception이 발생하는 시점과 같이 어플리케이션을 실행할 때, 특정 작업이 실행되는 시점(비즈니스 매소드, **핵심관심사**)을 의미한다.
포인트컷 (Pointcut)	분리된 기능(횡단관심사, 로깅 ...)들을 결합시키기 위한 규칙(패턴)이 필요한 데, 이와 같은 패턴을 포인터컷이라고 한다.
위빙 (Weaving)	"엮는다, 실로 짠다" 분리된 기능들을 하나로 결합시키는 것을 의미. 분리하여 개발된 기능들을 Weaving하는 작업을 도와주는 것이 AOP 툴이 하는 역할이다.
어스펙트 (Aspect)	어드바이스와 포인트컷을 합쳐서 하나의 Aspect라고 한다. 즉, 일정한 패턴을 가지는 클래스에 어드바이스를 적용하도록 지원할 수 있는 것을 Aspect라고 한다. 혹은 어드바이저(Advisor)라고 한다.
어드바이저 (Advisor)	어드바이스와 포인트컷을 하나로 묶어 다루는 것을 말한다.

현재 오류 없이 실행된 결과화면



오류를 추가해서 트랜잭션 기능 보기

실행 결과-트랜잭션 테스트 전



seq가 5까지 되어 있음

트랜잭션 처리 확인을 위하여 SQL문 수정 및 중복 삽입 처리

BoardServiceClient.java | BoardServiceImpl.java | BoardDAOJdbcTemplate.java

```
1  package kr.ac.inje.comsi.board.impl;
2
3  import java.util.List;
10
11 @Repository("boardDAOJdbcTemplate")
12 public class BoardDAOJdbcTemplate {
13
14     @Autowired
15     private JdbcTemplate jdbcTemplate;
16
17     // SQL 명령어
18 //  private final String BOARD_INSERT = "insert into board(seq, title, writer, "
19 //          + "content) values((select nvl(max(seq),0)+1 from board),?,?,?)";
20     private final String BOARD_INSERT = "insert into board(seq, title, writer, "
21             + "content) values(?,?,?,?)";
22     private final String BOARD_UPDATE = "update board set title=?, content=? where seq=?";
23     private final String BOARD_DELETE = "delete board where seq=?";
24     private final String BOARD_GET = "select * from board where seq=?";
25     private final String BOARD_LIST = "select * from board order by seq desc";
26
27     // CRUD 기능의 메소드 구현
28     // 글 등록
29     public void insertBoard(BoardVO vo){
30         System.out.println("===> JDBC로 insertBoard() 기능 처리");
```

Seq 번호를 생성하는 것이 아니라
사용자의 입력을 받아서 처리하도록 수정

- Seq가 자동으로 증가하도록 한 소스를 수동으로 입력하도록 수정

BoardServiceClient.java | BoardDAOSpring.java | BoardServiceImpl.java

```java
package kr.ac.inje.comsi.board;

import java.util.List;

import org.springframework.context.support.AbstractApplicationContext;
import org.springframework.context.support.GenericXmlApplicationContext;

public class BoardServiceClient {

    public static void main(String[] args) {
        // 1. Spring 컨테이너 구동
        AbstractApplicationContext container = new GenericXmlApplicationContext("applicationContext.xml");
        // 2. Spring 컨테이너로부터 BoardServiceImpl 객체를 Lookup
        BoardService boardService = (BoardService) container.getBean("boardService");
//      // 3. 글 동록 기능 테스트
        BoardVO vo = new BoardVO();
        vo.setSeq(100);
        vo.setTitle("JDBD 테스트2");
        vo.setWriter("관리자");
        vo.setContent("JDBC 테스트2.....................");
        boardService.insertBoard(vo);
        // 4. 글 목록 검색 기능 테스트
        List<BoardVO> boardList = boardService.getBoardList(vo);
        System.out.println(boardList.size());
        for(BoardVO board: boardList) {
            System.out.println("-->" + board.toString());
        }
        // 5. Spring 컨테이너 종료
        container.close();
    }
}
```

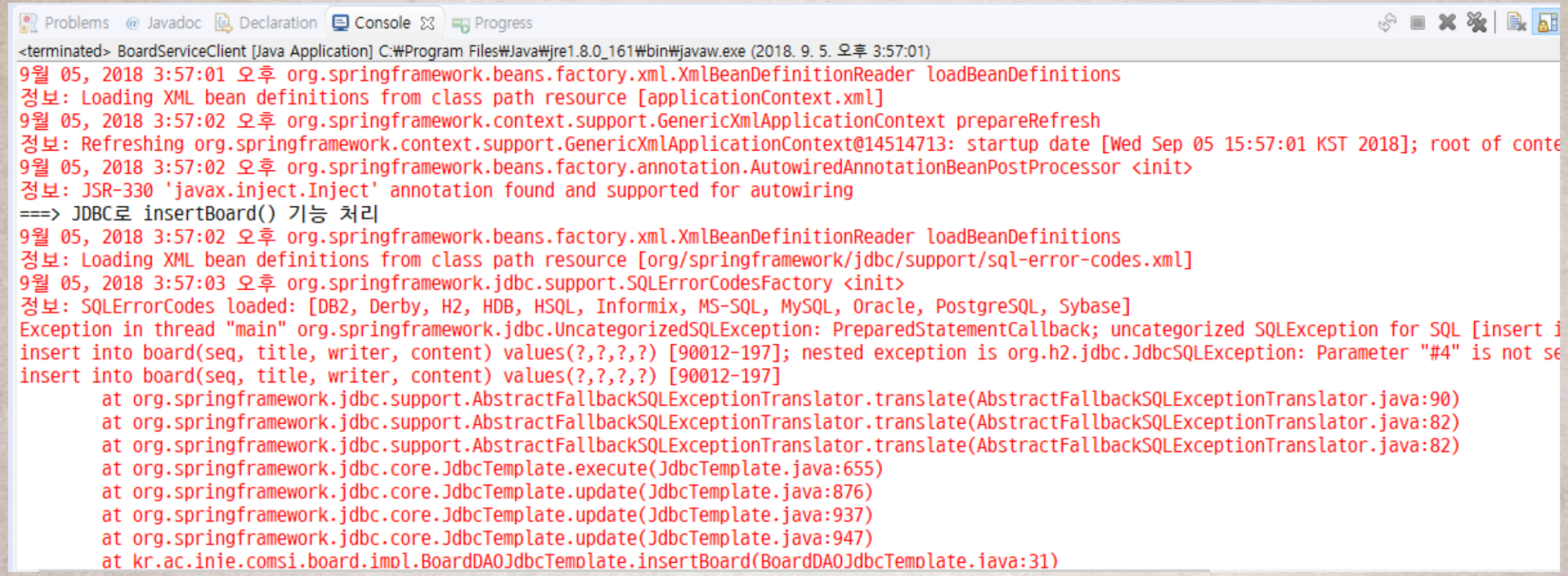
Seq=100으로 고정

이중 삽입 코드 추가

BoardServiceClient.java | BoardServiceImpl.java

```
1  package kr.ac.inje.comsi.board.impl;
2
3  import java.util.List;
10
11 @Service("boardService")
12 public class BoardServiceImpl implements BoardService {
13
14 //  @Autowired
15 //  private BoardDAO boardDAO;
16
17     @Autowired
18     private BoardDAOJdbcTemplate boardDAOJdbcTemplate;
19
20     @Override
21     public void insertBoard(BoardVO vo) {
22         //boardDAO.insertBoard(vo);
23         boardDAOJdbcTemplate.insertBoard(vo);
24         boardDAOJdbcTemplate.insertBoard(vo);
25     }
26
27     @Override
28     public void updatetBoard(BoardVO vo) {
29         //boardDAO.updatetBoard(vo);
30         boardDAOJdbcTemplate.updateBoard(vo);
31     }
32
```

트랜잭션 처리하도록 오류 추가 후 실행 결과



BoardServiceClient.java | BoardServiceImpl.java | BoardDAOJdbcTemplate.java

```java
package kr.ac.inje.comsi.board;

import java.util.List;

public class BoardServiceClient {

    public static void main(String[] args) {

        // 1. Spring 컨테이너 구동
        AbstractApplicationContext container = new GenericXmlApplicationContext("applicationContext.xml");

        // 2. Spring 컨테이너로부터 BoardServiceImpl 객체를 Lookup
        BoardService boardService = (BoardService) container.getBean("boardService");

//      // 3. 글 등록 기능 테스트
        BoardVO vo = new BoardVO();
        vo.setSeq(100); // 100으로 고정
        vo.setTitle("JDBCTemplate테스트2");
        vo.setWriter("홍길동");
        vo.setContent("임시 내용 테스트 ......................");
//
//      boardService.insertBoard(vo);

        // 4. 글 목록 검색 기능 테스트
```

seq=100이 입력되지 않음

Problems | Javadoc | Declaration | Console | Progress

<terminated> BoardServiceClient [Java Application] C:₩Program Files₩Java₩jre1.8.0_161₩bin₩javaw.exe (2018. 9. 5. 오후 3:58:51)

```
9월 05, 2018 3:58:51 오후 org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
정보: Loading XML bean definitions from class path resource [applicationContext.xml]
9월 05, 2018 3:58:51 오후 org.springframework.context.support.GenericXmlApplicationContext prepareRefresh
정보: Refreshing org.springframework.context.support.GenericXmlApplicationContext@14514713: startup date [Wed Sep 05 15:58:51 KST 2018]; root of context
9월 05, 2018 3:58:51 오후 org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor <init>
정보: JSR-330 'javax.inject.Inject' annotation found and supported for autowiring
===> JDBC로 getBoardList() 기능 처리
5
-->BoardVO [seq=5, title=JDBCTemplate테스트, writer=홍길동, content=임시 내용 테스트 ......................, regDate=null, cnt=0]
-->BoardVO [seq=4, title=JDBCTemplate테스트, writer=홍길동, content=임시 내용 테스트 ......................, regDate=null, cnt=0]
-->BoardVO [seq=3, title=JDBC테스트, writer=홍길동, content=임시 내용 테스트 ......................, regDate=null, cnt=0]
-->BoardVO [seq=2, title=임시제목, writer=홍길동, content=임시내용입니다., regDate=null, cnt=0]
-->BoardVO [seq=1, title=가입인사, writer=관리자, content=잘 부탁드립니다., regDate=null, cnt=0]
9월 05, 2018 3:58:52 오후 org.springframework.context.support.GenericXmlApplicationContext doClose
정보: Closing org.springframework.context.support.GenericXmlApplicationContext@14514713: startup date [Wed Sep 05 15:58:51 KST 2018]; root of context hie
```