

9. 비즈니스 컴포넌트 실습1

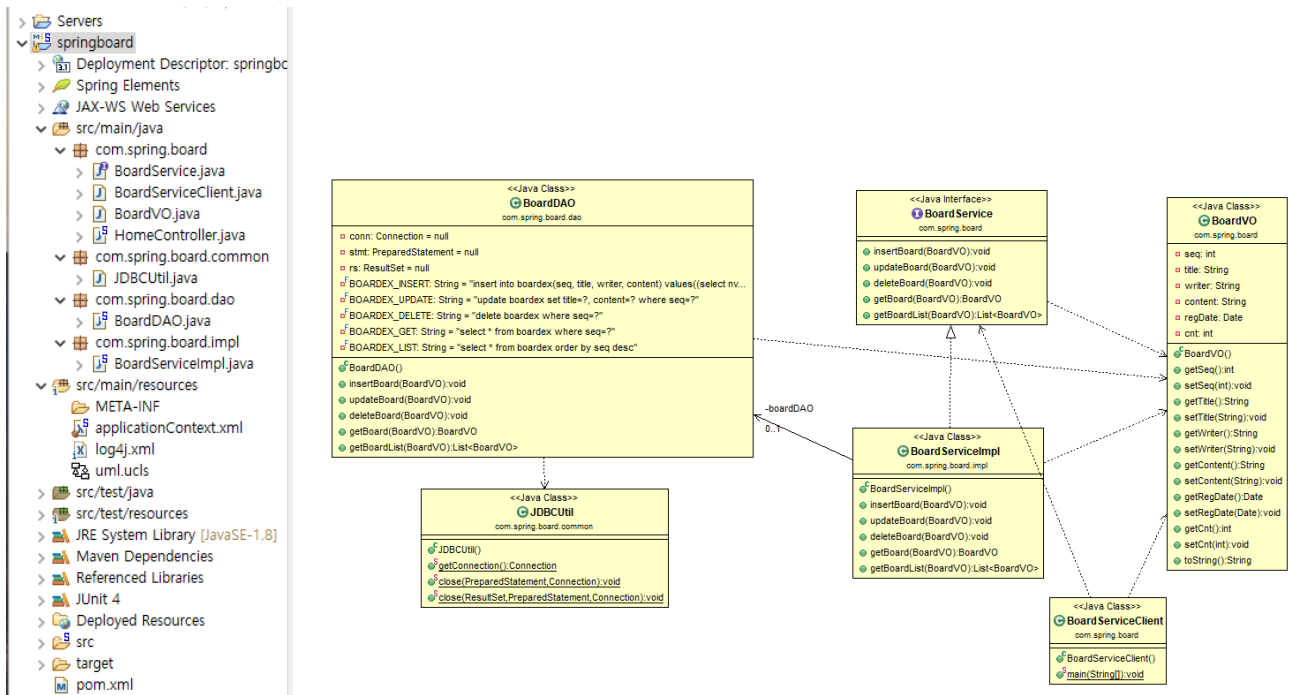
9.1 BoardService 컴포넌트 구조

일반적으로 비즈니스 컴포넌트는 네 개의 자바 파일로 구성 된다. 게시판 관련 컴포넌트를 구현하면서 비즈니스 컴포넌트의 작성 순서와 이름 규칙등을 알아본다.

BOARD 테이블과 관련된 BoardService 컴포넌트에 대한 클래스 다이어그램이며, BoardVO, BoardDAO, BoardService, BoardServiceImpl 클래스로 구현되어 있다.

Help -> Install New Software -> <http://www.objectaid.com/update/current> 추가 설치

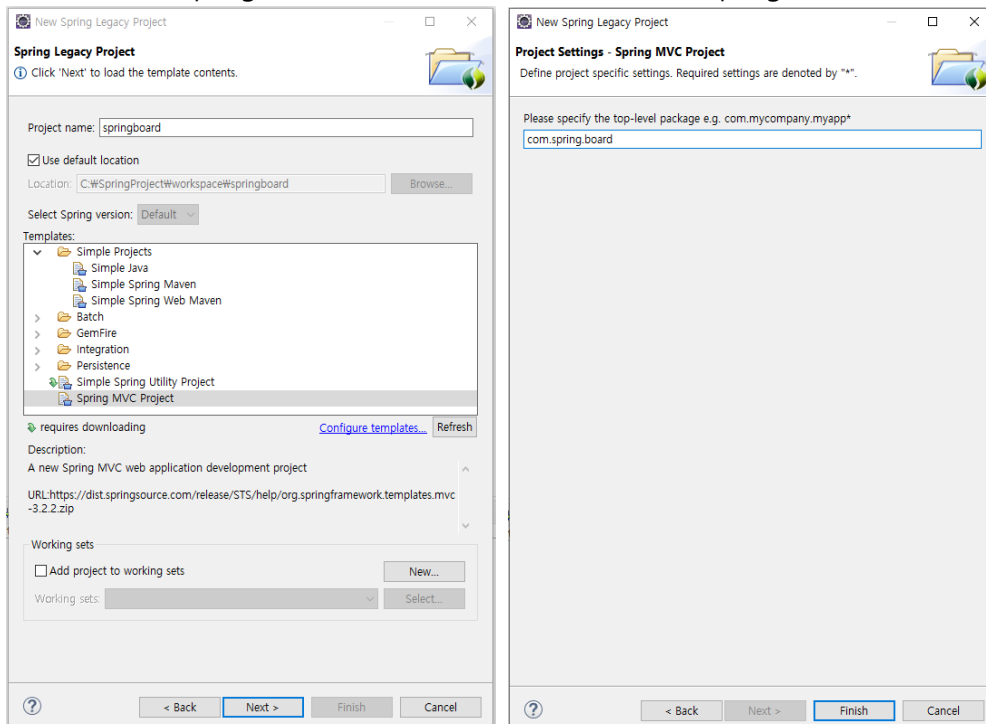
BoardService 컴포넌트 관련 파일들의 위치와 클래스 다이어그램



Spring Legacy Project 생성

프로젝트명 : springboard

패키지명 : com.spring.board



버전 변경 환경 설정

9.2 Value Object 클래스 작성

VO(Value Object)클래스는 레이어와 레이어 사이에서 관련된 데이터를 한꺼번에 주고 받을 목적으로 사용되는 클래스이다. DTO(Data Transfer Object)라 하기도 하는데, 데이터 전달을 목적으로 사용하는 객체이므로 같은 의미로 사용된다.

board.sql

```
create table boardex(
    seq number(5) primary key,
    title varchar2(200),
    writer varchar2(20),
    content varchar2(2000),
    regdate date default sysdate,
    cnt number(5) default 0
);
```

BoardVO.java 생성

```
package com.spring.board;

import java.util.Date;

// VO(Value Object)
public class BoardVO {
    private int seq;
    private String title;
    private String writer;
    private String content;
    private Date regDate;
    private int cnt;
    public int getSeq() {
        return seq;
    }
    public void setSeq(int seq) {
        this.seq = seq;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getWriter() {
        return writer;
    }
}
```

```

    public void setWriter(String writer) {
        this.writer = writer;
    }
    public String getContent() {
        return content;
    }
    public void setContent(String content) {
        this.content = content;
    }
    public Date getRegDate() {
        return regDate;
    }
    public void setRegDate(Date regDate) {
        this.regDate = regDate;
    }
    public int getCnt() {
        return cnt;
    }
    public void setCnt(int cnt) {
        this.cnt = cnt;
    }
    @Override
    public String toString() {
        return "BoardVO [seq=" + seq + ", title=" + title + ", writer=" + writer + ", content="
+ content + ", regDate=" + regDate + ", cnt=" + cnt + "]\n";
    }
}

```

toString() 메소드는 나중에 VO 객체의 값을 출력할 때 사용한다.

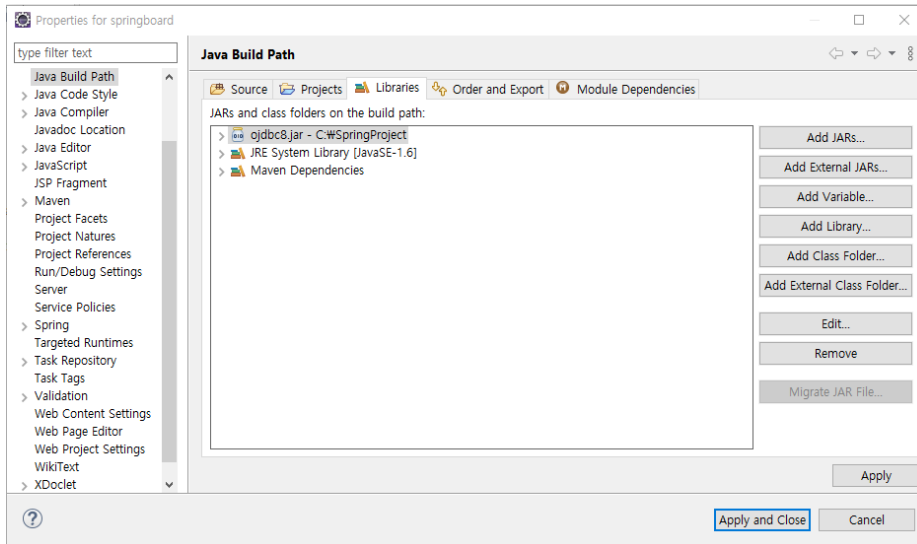
9.3 DAO 클래스 작성

DAO(Data Access Object)클래스는 데이터베이스 연동을 담당하는 클래스이다. 따라서 DAO클래스에는 CRUD(Create, Read, Update, Delete)기능의 메소드가 구현되어야 한다. 이때 데이터베이스에서 제공하는 JDBC 드라이버 파일이 필요하다.

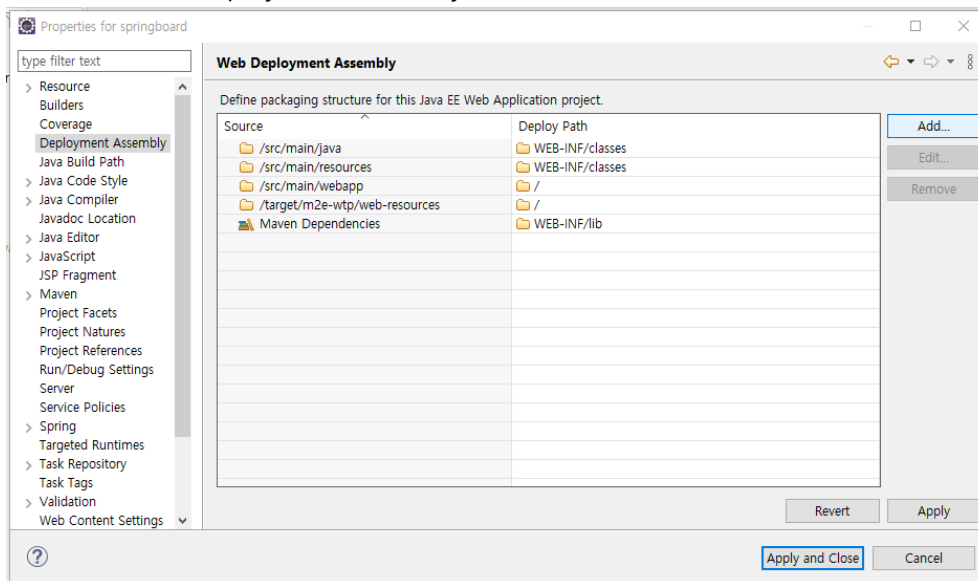
(1) Oracle 드라이버 설정

프로젝트명 마우스 오른쪽 버튼 클릭 Build Path 클릭 Libraries 탭 선택

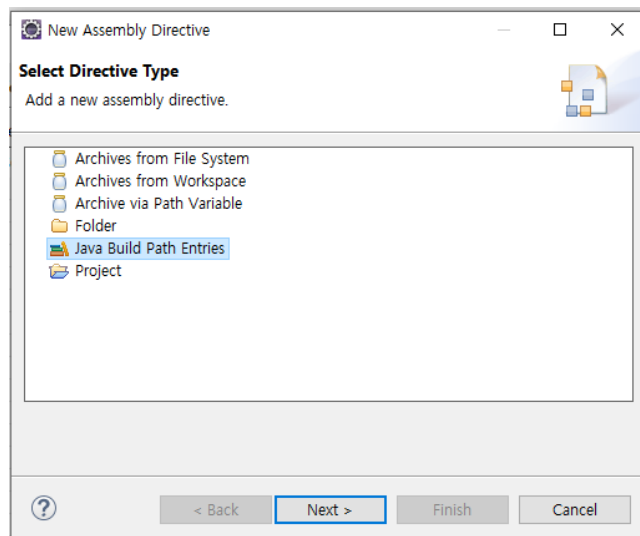
Add External JARs... 버튼 클릭 ojdbc8.jar 파일 추가 하고 Apply 버튼 클릭



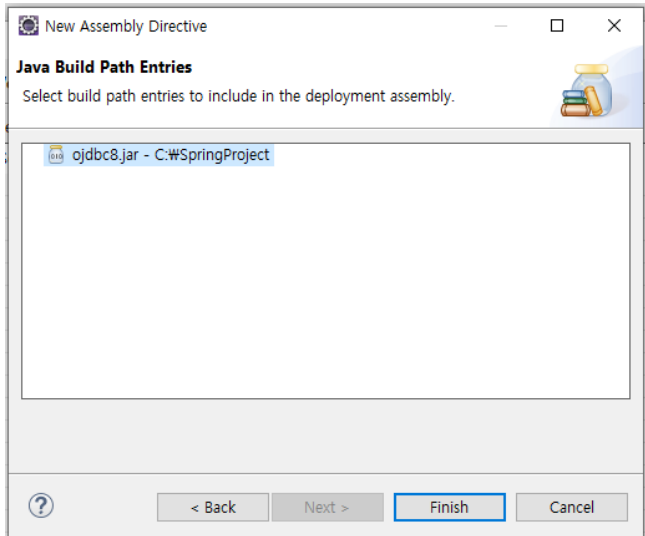
왼쪽 메뉴에서 Deployment Assembly 선택 -> Add 버튼클릭



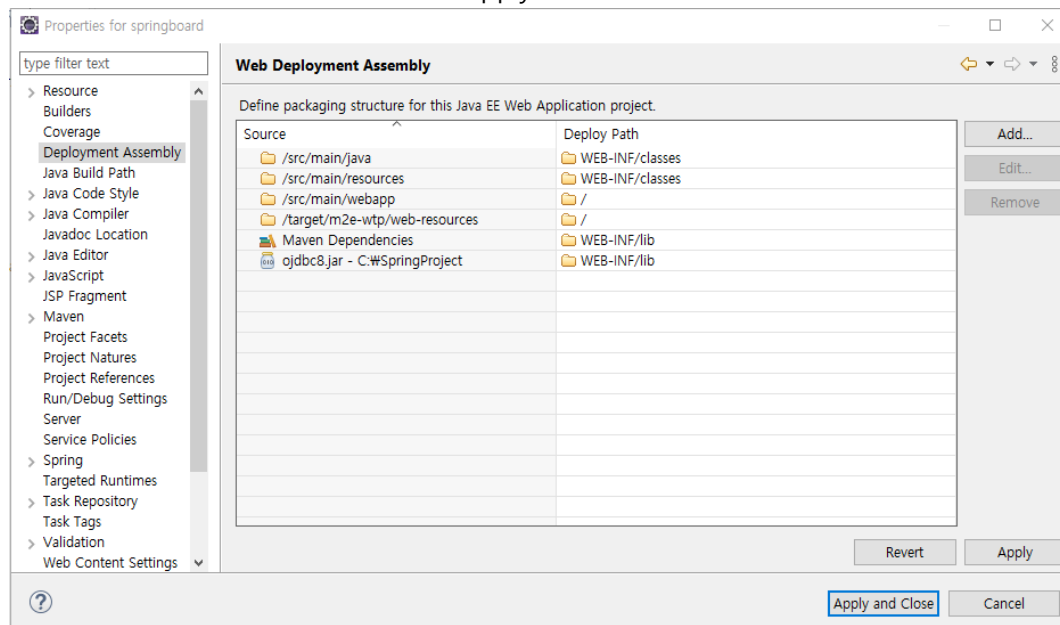
Java Build Path Entries 선택 Next 클릭



Finish 클릭



드라이브 파일 추가된 내용 확인 후 Apply and Close 클릭



JDBC 테스트 코드

scr/test/java

OracleJDBCTest.java

```
package com.spring.board;

import java.sql.Connection;
import java.sql.DriverManager;
import org.junit.Test;

public class OracleJDBCTest {
    private static final String DRIVER = "oracle.jdbc.OracleDriver";
    private static final String URL = "jdbc:oracle:thin:@localhost:1521:XE";
```

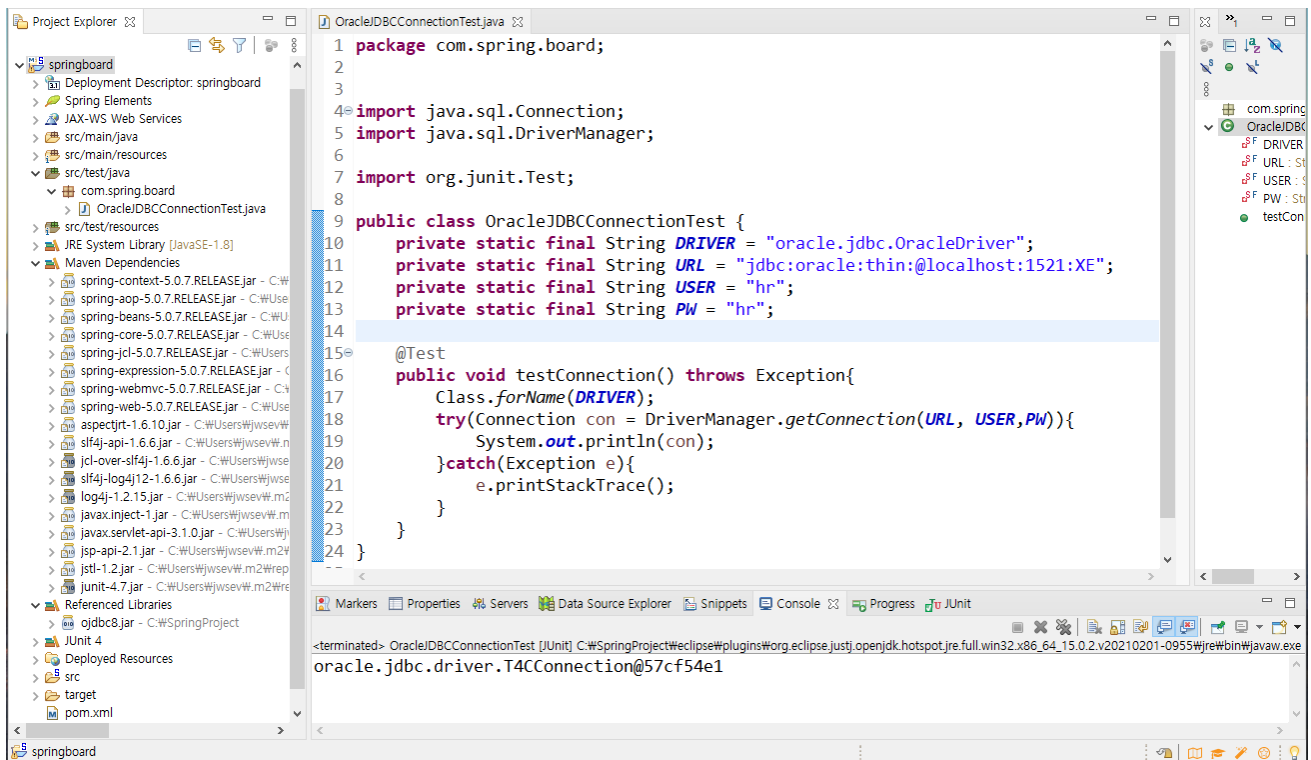
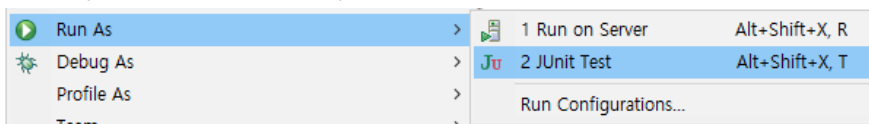
```

private static final String USER = "hr";
private static final String PW = "hr";

@Test
public void testConnection() throws Exception{
    Class.forName(DRIVER);
    try {
        Connection con = DriverManager.getConnection(URL, USER,PW);
        System.out.println(con);
    }catch(Exception e){
        e.printStackTrace();
    }
}
}

```

테스트(Run As -> JUnit Test)



위와 같은 결과가 나오며는 오라클과 연결이 성공한 것이다.

@Test에서 JUnit 버전은 JUnit 4 버전을 선택한다. pom.xml 에서 JUnit 4.7 버전을 JUnit 4.12 버전으로 수정한다.

pom.xml

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

(2) JDBC Utility 클래스

MyBatis 프레임워크를 사용하기 전까지는 데이터베이스 연동 처리를 JDBC로 한다. 모든 DAO 클래스에서 공통으로 사용할 JDBCUtil 클래스를 작성하여 Connection 획득과 해제를 공통으로 처리한다.

JDBCUtil.java

```
package com.spring.board.common;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class JDBCUtil {
    public static Connection getConnection() {
        try {
            Class.forName("oracle.jdbc.OracleDriver");
            return DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE", "hr", "hr");
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    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;
    }
}

}

public static void close(ResultSet rs, PreparedStatement stmt, Connection conn) {
    if(rs != null) {
        try {
            if(!rs.isClosed()) rs.close();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            rs = null;
        }
    }

    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;
        }
    }
}

```



```

    }
    }
}

```

(3) DAO 클래스

BoardVO 객체를 매개변수와 리턴 타입으로 사용하면서 BOARD 테이블과 CRUD 기능을 처리할 BoardDAO 클래스를 작성한다.

BoardDAO.java

```

package com.spring.board.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;
import org.springframework.stereotype.Repository;
import com.spring.board.BoardVO;
import com.spring.board.common.JDBCUtil;

// DAO(Data Access Object)
@Repository("boardDAO")
public class BoardDAO {
    // JDBC 관련 변수
    private Connection conn = null;
    private PreparedStatement stmt = null;
    private ResultSet rs = null;

    // SQL 명령어들
    private final String BOARDEX_INSERT = "insert into boardex(seq, title, writer, content)
values((select nvl(max(seq), 0)+1 from boardex),?,?,?)";
    private final String BOARDEX_UPDATE = "update boardex set title=?, content=? where seq=?";
    private final String BOARDEX_DELETE = "delete boardex where seq=?";
    private final String BOARDEX_GET = "select * from boardex where seq=?";
    private final String BOARDEX_LIST = "select * from boardex order by seq desc";

    // CRUD 기능의 메소드 구현
    // 글 등록
    public void insertBoard(BoardVO vo) {
        System.out.println("==> JDBC로 insertBoard() 기능 처리");
        try {
            conn = JDBCUtil.getConnection();

```

```

        stmt = conn.prepareStatement(BOARDEX_INSERT);
        stmt.setString(1, vo.getTitle());
        stmt.setString(2, vo.getWriter());
        stmt.setString(3, vo.getContent());
        stmt.executeUpdate();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        JDBCUtil.close(stmt, conn);
    }
}

// 글 수정
public void updateBoard(BoardVO vo) {
    System.out.println("==> JDBC로 updateBoard() 기능 처리");
    try {
        conn = JDBCUtil.getConnection();
        stmt = conn.prepareStatement(BOARDEX_UPDATE);
        stmt.setString(1, vo.getTitle());
        stmt.setString(2, vo.getContent());
        stmt.setInt(3, vo.getSeq());
        stmt.executeUpdate();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        JDBCUtil.close(stmt, conn);
    }
}

// 글 삭제
public void deleteBoard(BoardVO vo) {
    System.out.println("==> JDBC로 deleteBoard() 기능 처리");
    try {
        conn = JDBCUtil.getConnection();
        stmt = conn.prepareStatement(BOARDEX_DELETE);
        stmt.setInt(1, vo.getSeq());
        stmt.executeUpdate();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        JDBCUtil.close(stmt, conn);
    }
}

```

```

}

// 글 상세 조회
public BoardVO getBoard(BoardVO vo) {
    System.out.println("==> JDBC로 getBoard() 기능 처리");
    BoardVO boardex = null;
    try {
        conn = JDBCUtil.getConnection();
        stmt = conn.prepareStatement(BOARDEX_GET);
        stmt.setInt(1, vo.getSeq());
        rs = stmt.executeQuery();
        if (rs.next()) {
            boardex = new BoardVO();
            boardex.setSeq(rs.getInt("SEQ"));
            boardex.setTitle(rs.getString("TITLE"));
            boardex.setWriter(rs.getString("WRITER"));
            boardex.setContent(rs.getString("CONTENT"));
            boardex.setRegDate(rs.getDate("REGDATE"));
            boardex.setCnt(rs.getInt("CNT"));
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        JDBCUtil.close(rs, stmt, conn);
    }
    return boardex;
}

// 글 목록 조회
public List<BoardVO> getBoardList(BoardVO vo) {
    System.out.println("==> JDBC로 getBoardList() 기능 처리");
    List<BoardVO> boardList = new ArrayList<BoardVO>();
    try {
        conn = JDBCUtil.getConnection();
        stmt = conn.prepareStatement(BOARDEX_LIST);
        rs = stmt.executeQuery();
        while (rs.next()) {
            BoardVO boardex = new BoardVO();
            boardex.setSeq(rs.getInt("SEQ"));
            boardex.setTitle(rs.getString("TITLE"));
            boardex.setWriter(rs.getString("WRITER"));
            boardex.setContent(rs.getString("CONTENT"));
        }
    }

```

```

        boardex.setRegDate(rs.getDate("REGDATE"));
        boardex.setCnt(rs.getInt("CNT"));
        boardList.add(boardex);
    }
} catch (Exception e) {
    e.printStackTrace();
} finally {
    JDBCUtil.close(rs, stmt, conn);
}
return boardList;
}
}

```

위 클래스 객체를 스프링 컨테이너가 생성할 수 있도록 클래스 선언부에 @Repository 어노테이션을 설정한다. @Component를 사용해도 되지만 DAO 기능의 클래스에는 @Repository를 사용하는 것이 적합하다.

CRUD (Creat,Read,Update,Delete) 기능의 메소드 이름

기능	메소드 이름
등록	insert테이블명
수정	update테이블명
삭제	delete테이블명
상세 조회	get테이블명(혹은 select테이블명)
목록 검색	get테이블명List(혹은 select테이블명List)

9.4 Service 인터페이스

BoardService 인터페이스는 BoardServiceImpl 클래스가 구현한다. BoardDAO 클래스는 독립된 클래스로 구현한다.

BoardService.java

```

package com.spring.board;
import java.util.List;
public interface BoardService {
    // CRUD 기능의 메소드
    // 글 등록
    void insertBoard(BoardVO vo);
    // 글 수정
    void updateBoard(BoardVO vo);
    // 글 삭제
    void deleteBoard(BoardVO vo);
    // 글 상세 조회
    BoardVO getBoard(BoardVO vo);
    // 글 목록 조회
    List<BoardVO> getBoardList(BoardVO vo);
}

```

```
}
```

9.5 Service 구현 클래스

BoardService 인터페이스를 구현한 BoardServiceImpl 클래스의 비즈니스 메소드를 구현할 때, 멤버 변수로 선언된 BoardDAO를 이용한다.

BoardServiceImpl.java

```
package com.spring.board.impl;

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.spring.board.BoardService;
import com.spring.board.BoardVO;
import com.spring.board.dao.BoardDAO;

@Service("boardService")
public class BoardServiceImpl implements BoardService {

    @Autowired
    private BoardDAO boardDAO;

    @Override
    public void insertBoard(BoardVO vo) {
        boardDAO.insertBoard(vo);
    }

    @Override
    public void updateBoard(BoardVO vo) {
        boardDAO.updateBoard(vo);
    }

    @Override
    public void deleteBoard(BoardVO vo) {
        boardDAO.deleteBoard(vo);
    }

    @Override
    public BoardVO getBoard(BoardVO vo) {
        return boardDAO.getBoard(vo);
    }

    @Override
    public List<BoardVO> getBoardList(BoardVO vo) {
        return boardDAO.getBoardList(vo);
    }
}
```

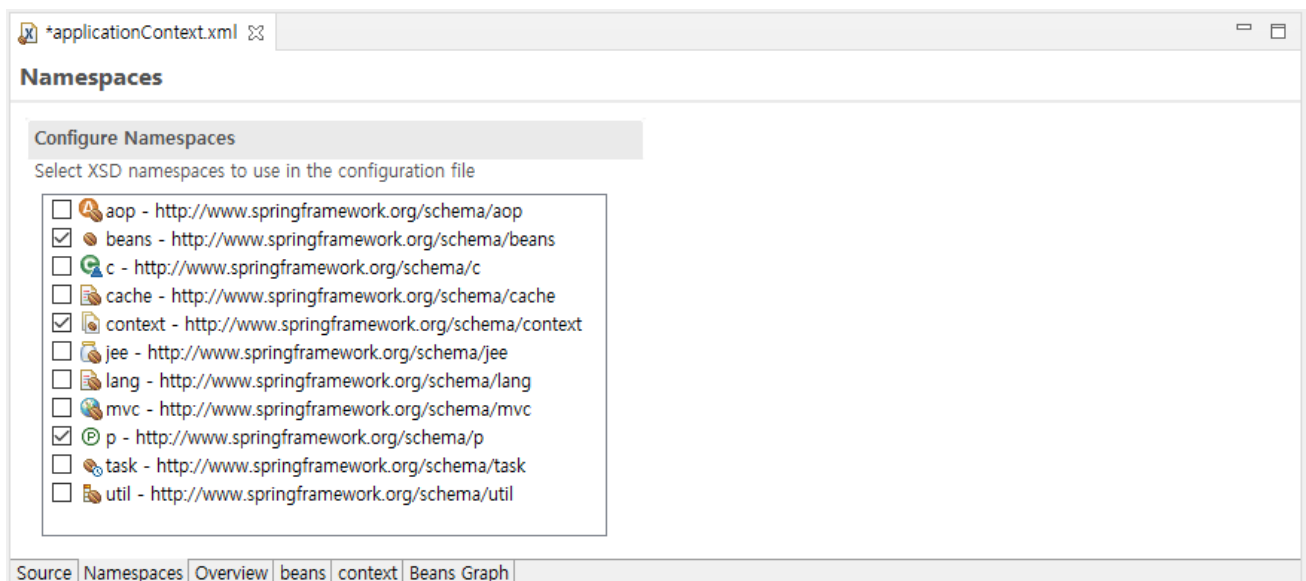
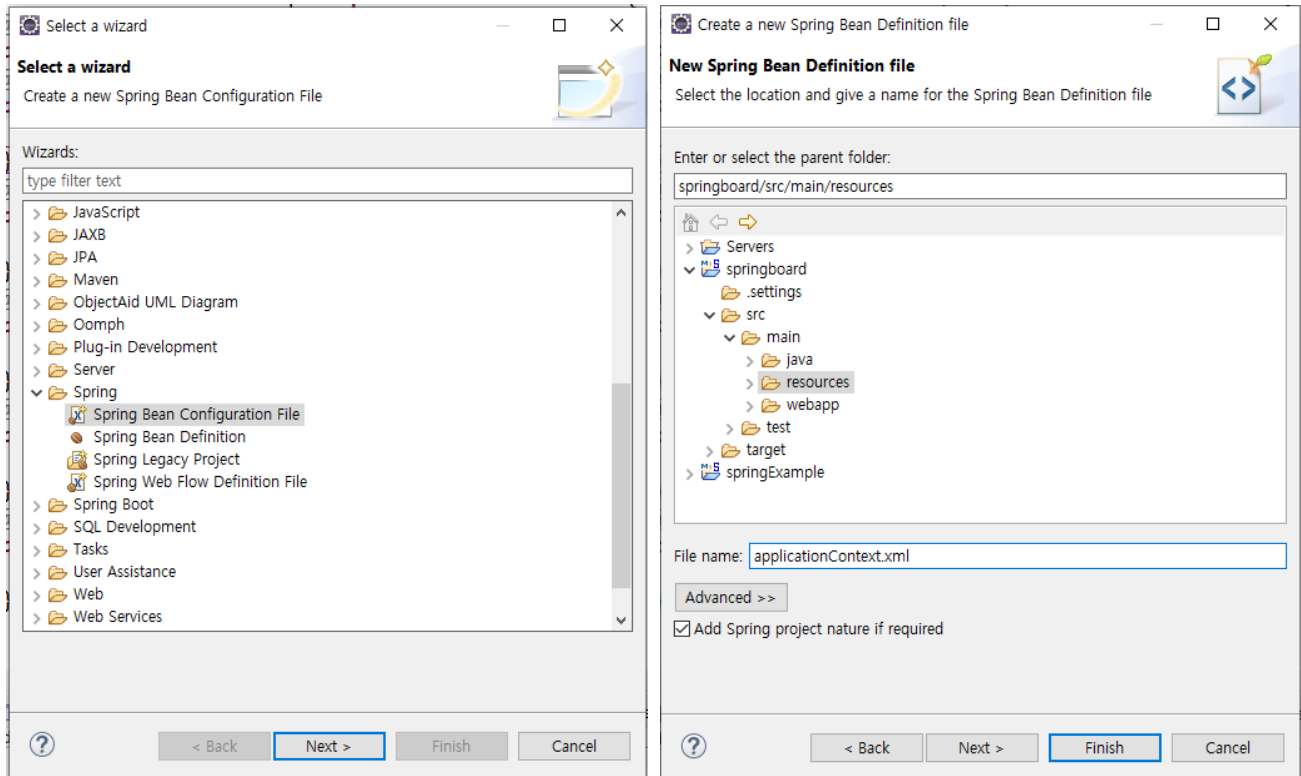
BoardServiceImpl 클래스 선언부에 객체 생성을 위한 @Service가 선언되어 있으며, 클라이언트 프로그램에서 boardService라는 이름으로 객체를 요청할 수 있도록 아이디도 설정한다.

그리고 BoardServiceImpl은 데이터베이스 연동이 포함된 비즈니스로직 처리를 위해서 BoardDAO 타입의 객체를 멤버 변수로 가진다. 이 변수에 BoardDAO 타입의 객체를 의존성 주입하기 위해서 @Autowired를 설정한다.

9.6 BoardService 컴포넌트 테스트

(1) 스프링 설정 파일 수정

BoardService 컴포넌트를 스프링 기반으로 테스트하려면 우선 스프링 설정 파일에 <context:component-scan> 설정을 수정한다.



src/main/resources/applicationContext.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:context="http://www.springframework.org/schema/context"
       xmlns:p="http://www.springframework.org/schema/p"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
                           http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.3.xsd">

    <context:component-scan base-package="com.spring">
    </context:component-scan>

</beans>
```

컴포넌트 스캔의 범위를 'com.spring' 패키지로 지정하면 BoardServiceImpl 클래스와 BoardDAO 클래스가 스캔 범위에 포함되어 객체가 생성된다. 그리고 의존성 주입도 처리된다.

(2) 클라이언트 작성 및 실행

마지막으로 스프링 컨테이너를 구동하고 BoardService 컴포넌트를 사용하는 클라이언트 프로그램을 다음과 같이 작성하여 글 등록 기능과 글 목록 검색 기능을 테스트 한다.

(/src/test/java)

BoardServiceClient.java

```
package com.spring.board;

import java.util.List;

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

public class BoardServiceClient {

    @Test
    public void boardTest() {
        // 1. Spring 컨테이너를 등록한다.
        AbstractApplicationContext container = new
        GenericXmlApplicationContext("applicationContext.xml");

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

        // 3. 글 등록 기능 테스트
```

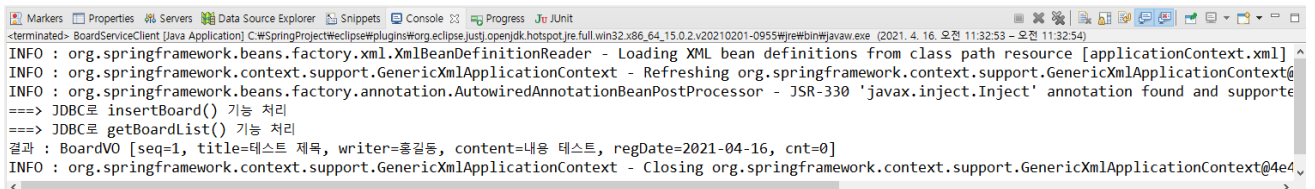
```

        BoardVO vo = new BoardVO();
        vo.setTitle("테스트 제목");
        vo.setWriter("홍길동");
        vo.setContent("내용 테스트");
        boardService.insertBoard(vo);

        // 4. 글 목록 검색 기능 테스트
        List<BoardVO> boardList = boardService.getBoardList(vo);
        for (BoardVO board : boardList) {
            System.out.println("결과 : " + board.toString());
        }
        // 5. Spring 컨테이너 종료
        container.close();
    }
}

```

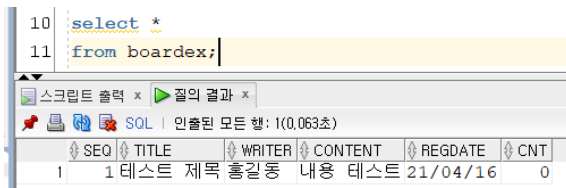
Run As -> JUnit Test



```

-terminated- BoardServiceClient (Java Application) C:\SpringProject\workspace\spring-junit\src\main\java\com\example\board\BoardServiceClient.java (2021. 4. 16. 오전 11:32:53 - 오전 11:32:54)
INFO : org.springframework.beans.factory.xml.XmlBeanDefinitionReader - Loading XML bean definitions from class path resource [applicationContext.xml]
INFO : org.springframework.context.support.GenericXmlApplicationContext - Refreshing org.springframework.context.support.GenericXmlApplicationContext@4e4...
INFO : org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor - JSR-330 'javax.inject.Inject' annotation found and supported
==> JDBC로 insertBoard() 기능 처리
==> JDBC로 getBoardList() 기능 처리
결과 : BoardVO [seq=1, title=테스트 제목, writer=홍길동, content=내용 테스트, regDate=2021-04-16, cnt=0]
INFO : org.springframework.context.support.GenericXmlApplicationContext - Closing org.springframework.context.support.GenericXmlApplicationContext@4e4...

```



```

10 select *
11 from boardex;

```

SEQ	TITLE	WRITER	CONTENT	REGDATE	CNT
1	1 테스트 제목	홍길동	내용 테스트	21/04/16	0