

JAVA로 DB 연동 프로그래밍

- 1) 독립 어플리케이션 : JAVA
- 2) 웹 프로그래밍 : JSP + MyBatis
- 3) Spring + MyBatis, Spring + JPA

참고 : JAVA JDK 다운로드

방법1) <https://www.oracle.com/java/technologies/downloads/>

방법2) <https://github.com/ojdkbuild/ojdkbuild>

방법3) <https://bell-sw.com/pages/downloads/#jdk-17-lts>

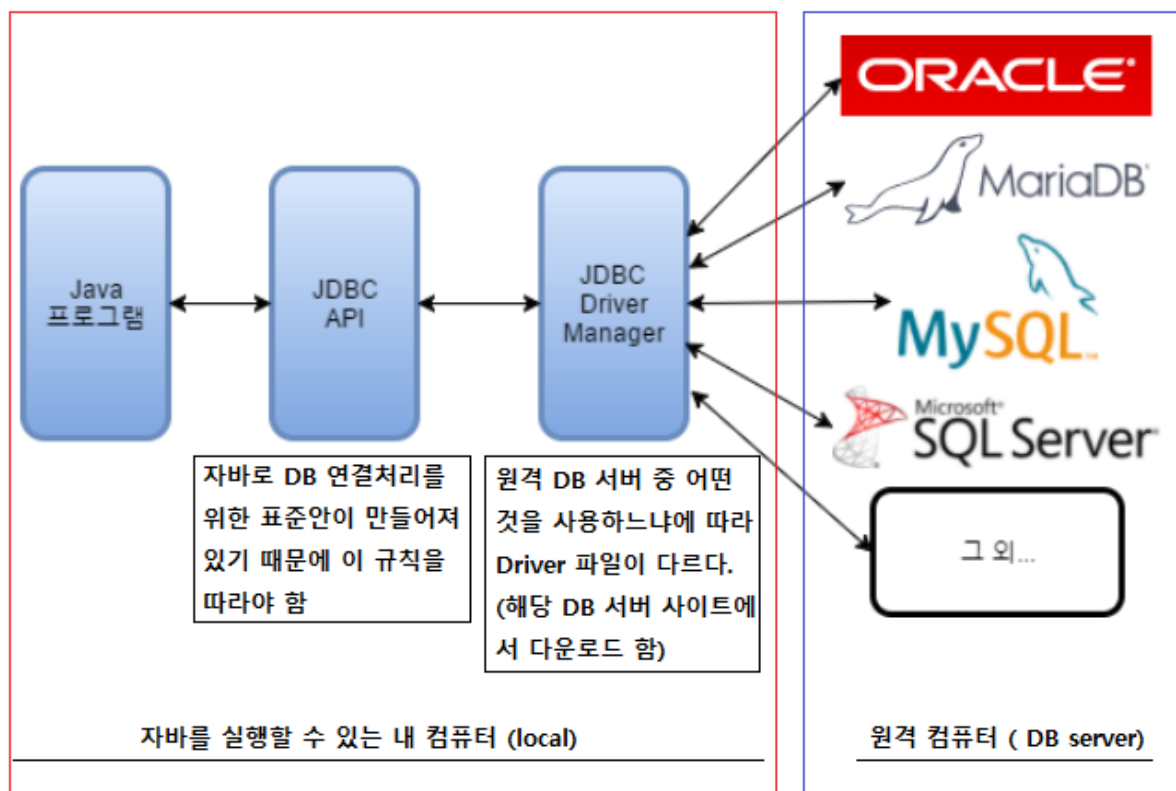
참고 : Lombok 이란? <https://needjarvis.tistory.com/696>

Java 기반의 프로젝트를 개발할 때 getter, setter, toString 등 반복적으로 작성하는 코드들이 존재한다. 이런 코드를 줄여주는 라이브러리이다. 즉, Lombok은 어노테이션 기반으로 Getter, Setter, Equals, ToString, Constructor 등과 같은 코드를 자동완성 해준다.

참고 : JAVA로 DB 연동은 JDBC API를 사용

<https://cafe.daum.net/flowlife/HqLo/17>

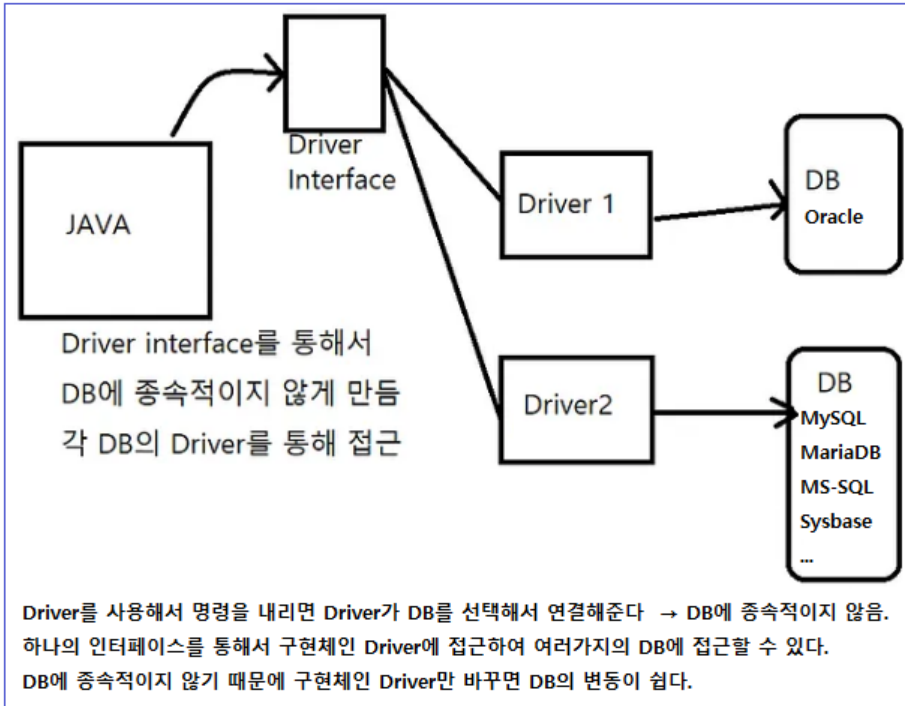
<https://devlog-wjdrbs96.tistory.com/139>



oracle : ojdbc 참고 사이트 <https://mant81.tistory.com/296>

mysql : connector/J 참고 사이트 <https://appsnnuri.tistory.com/495>

mariadb : connector/J (mariadb-java-client.jar) 참고 사이트 <https://cafe.daum.net/flowlife/HqLk/63>



* 실습용 MariaDB 또는 MySQL로 테이블 작성하기 - sangdata

```
create table sangdata(code int primary key,sang varchar(20),su int,dan int);
```

참고 : 자료 입력 후 select로 볼 때 한글이 깨질 경우 ... dan int)charset=utf8;

* Oracle로 실습용 테이블 작성

```
create table sangdata(code number primary key,sang varchar2(20),su number,dan number);
```

```
insert into sangdata values(1,'장갑',3,10000);
insert into sangdata values(2,'병어리장갑',2,12000);
insert into sangdata values(3,'가죽장갑',10,50000);
insert into sangdata values(4,'가죽점퍼',5,650000);
```

--- 오라클에서 테이블 작성 후 code(pk) 자동 증가 시키기 ---

```
CREATE SEQUENCE tmp_seq START WITH 1 INCREMENT BY 1 ;
INSERT INTO sangdata values(tmp_seq.NEXTVAL, '신상1', 5, 1000);
commit;
select * from sangdata;
```

-- MariaDB, MySQL 에서 code 자동 증가 시키기 ---

```
create table sangdata(code int primary key auto_increment,sang varchar(20),su int,dan int);
```

참고 : MariaDB는 MySQL 기반으로 만들어졌기 때문(fork한 서비스)에 쿼리를 비롯한 전반적인 사용법은 MySQL 과 유사하다. 비슷한 사용법 외에도 MariaDB는 MySQL 대비 더 좋은 장점이 있다.

MariaDB 개발이 좀 더 개방적이고 활발한 커뮤니티를 지님, 빠르고 투명한 보안패치 릴리즈, 다양한 스토리지 엔진, 더 나은 성능 및 기능, 호환성과 쉬운 마이그레이션, Linux의 표준 database임. 과거에는 My-SQL이었으나 오라클이 인수함으로 인해서 현재는 무료 제품인 MariaDB를 더 많이 사용하게 되었다.

실습 1) 독립 어플리케이션(응용 프로그램) : **JAVA**

DbTest1.java : Statement 클래스 사용

```
package pack;
```

```
import java.sql.*;
```

```
public class DbTest1 {
```

```
    Connection conn; // DB 연결
```

```
    Statement stmt; // SQL문 실행, 동일 query 수행 시 DB 서버의 효율성 저하. Sql문 확장에 제한적
```

```
    ResultSet rs; // select의 결과에 접근
```

```
    public DbTest1() {
```

```
        try {
```

```
            // Driver 로딩
```

```
            // 방법1 : MariaDB용 Driver 파일을 이용
```

```
            Class.forName("org.mariadb.jdbc.Driver");
```

```
            // 방법2 : MySQL용 Driver 파일을 이용
```

```
            // Class.forName("com.mysql.jdbc.Driver");
```

```
            // 방법3 : Oracle용 Driver 파일을 이용
```

```
            // Class.forName("oracle.jdbc.driver.OracleDriver");
```

```
        } catch (Exception e) {
```

```
            System.out.println("드라이버 로딩 실패:" + e);
```

```
            return;
```

```
        }
```

```
        connect();
```

```
    }
```

```
    private void connect() {
```

```
        try {
```

```
            // DB 연결
```

```
            // 방법1 : MariaDB
```

```
            conn = DriverManager.getConnection("jdbc:mariadb://localhost:3306/test", "root", "123");
```

```
            //conn = DriverManager.getConnection("jdbc:mariadb://127.0.0.1:3306/test", "root",
```

```
"123"); //localhost와 동일
```

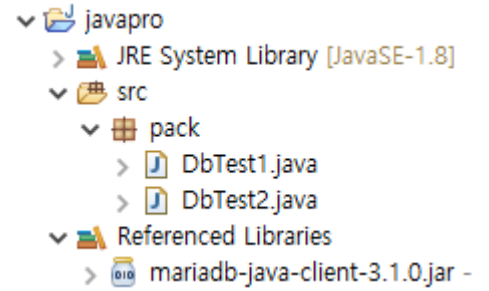
```
            //conn = DriverManager.getConnection("jdbc:mariadb://211.123.100.56:3306/test", "root",
```

```
"123"); //현재 컴의 ip 주소를 사용
```

```
            // 방법2 : MySQL
```

```
            // conn = DriverManager.getConnection("jdbc:mysql://@211.183.3.200:3306/test", "root",
```

```
"123");
```



```

        // 방법3 : oracle
        // conn = DriverManager.getConnection("jdbc:oracle:thin:@211.183.3.200:1521:orcl",
"scott", "tiger");
    } catch (Exception e) {
        System.out.println("DB 연결 실패:" + e);
        return;
    }

    try {
        // 참고 : java는 insert, update, delete할 때 autocommit이다. (Transaction이 내부적으로 진행됨)
        stmt = conn.createStatement();
        // 레코드 추가 - MariaDB 또는 MySql
        /*
String sqlins = "insert into sangdata values(9,'프린터',3,10000)";
        //oracle의 sequence를 사용할 경우
        //String sqlins = "insert into sangdata values(code_seq.nextval,'프린터',3,10000)";

        stmt.executeUpdate(sqlins); //select 이외의 sql문
        */

        // 레코드 수정
        /*
String sqlup = "update sangdata set sang='스캐너',su=7 where code=1";
        int result = stmt.executeUpdate(sqlup);
        if (result == 0)
            System.out.println("update 실패");
        */

        // 레코드 삭제
        /*
String sqldel = "delete from sangdata where code=9";
        int result2 = stmt.executeUpdate(sqldel);
        if (result2 == 0)
            System.out.println("delete 실패");
        */

        rs = stmt.executeQuery("select * from sangdata"); // 내 컴으로 테이블 자료 읽기
        //String sql = "select code as 코드, sang, su, dan + 1000 from sangdata order by dan
desc";

        //rs = stmt.executeQuery(sql);
        while (rs.next()) {
            String code = rs.getString("code");

```

```

        String sang = rs.getString("sang");
        String su = rs.getString("su");
        String dan = rs.getString("dan");
        //String code = rs.getString("코드");
        //String sang = rs.getString(2);
        //String su = rs.getString(3);
        //String dan = rs.getString(4);

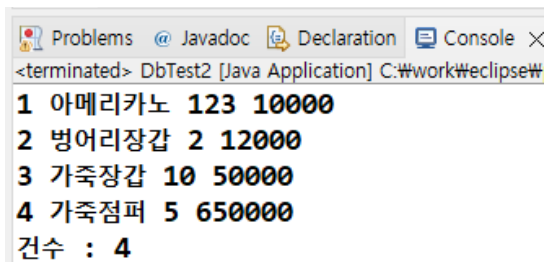
        System.out.println(code + " " + sang + " " + su + " " + dan);
    }

    String sql2 = "select count(*) from sangdata";    // 레코드 건수 읽기
    rs = stmt.executeQuery(sql2);
    rs.next();
    System.out.println("건수 : " + rs.getInt(1));
} catch (Exception e) {
    System.out.println("sql오류:" + e);
} finally {
    try {
        if (rs != null) rs.close();
        if (stmt != null) stmt.close();
        if (conn != null) conn.close();
    } catch (Exception e2) {
    }
}

}

public static void main(String[] args) {
    new DbTest1();
}
}

```



```

<terminated> DbTest2 [Java Application] C:\work\ eclipse\
1 아메리카노 123 10000
2 병어리장갑 2 12000
3 가죽장갑 10 50000
4 가죽점퍼 5 650000
건수 : 4

```

DbTest2.java : PreparedStatement 클래스 사용 – 선처리 방식(권장)

```
package pack;
import java.sql.*;

public class DbTest2 {
    Connection conn; // DB 연결
    PreparedStatement pstmt; // SQL문 실행, 동일한 sql문 반복될 경우 한번 만 작성 후 재활용.
    ResultSet rs; // select의 결과로 내 컴퓨터의 주기억장치에 저장된 레코드 집합에 접근용 클래스

    public DbTest2() {
        try {
            // Driver 로딩 -----
            // 방법1 : MariaDB용 Driver 파일을 이용
            Class.forName("org.mariadb.jdbc.Driver");

            // 방법2 : MySQL용 Driver 파일을 이용
            // Class.forName("com.mysql.jdbc.Driver");

            // 방법3 : Oracle용 Driver 파일을 이용
            // Class.forName("oracle.jdbc.driver.OracleDriver");
        } catch (Exception e) {
            System.out.println("드라이버 로딩 실패:" + e);
            return;
        }
        connect();
    }

    private void connect() {
        try {
            // DB 연결
            // 방법1 : MariaDB
            conn = DriverManager.getConnection("jdbc:mariadb://localhost:3306/test", "root", "123");
            //conn = DriverManager.getConnection("jdbc:mariadb://127.0.0.1:3306/test", "root",
"123"); //localhost와 동일
            //conn = DriverManager.getConnection("jdbc:mariadb://211.123.100.56:3306/test", "root",
"123"); //현재 컴의 ip 주소 사용

            // 방법2 : MySQL
            // conn = DriverManager.getConnection("jdbc:mysql://@211.183.3.200:3306/test", "root",
"123");

            // 방법2 : oracle
            // conn = DriverManager.getConnection("jdbc:oracle:thin:@211.183.3.200:1521:ordl",
```

```

"scott", "tiger");
    } catch (Exception e) {
        System.out.println("DB 연결 실패:" + e);
        return;
    }

    try {
        // 레코드 추가 - MariaDB 또는 MySql
        /*
        String sqlins = "insert into sangdata values(?,?,?,?)";
        //oracle의 sequence를 사용할 경우
        //String sqlins = "insert into sangdata values(code_seq.nextval,?,?,?)";

        pstmt = conn.prepareStatement(sqlins);
        pstmt.setInt(1, 9);
        pstmt.setString(2, "아메리카노");
        pstmt.setInt(3, 123);
        pstmt.setString(4, "5000");
        pstmt.executeUpdate(); //select 이외의 sql문
        */

        // 레코드 수정
        /*
        String sqlup = "update sangdata set sang=?,su=? where code=?";
        pstmt = conn.prepareStatement(sqlup);
        pstmt.setString(1, "아메리카노");
        pstmt.setInt(2, 123);
        pstmt.setInt(3, 2);    // 2번 레코드가 수정 대상
        int result = pstmt.executeUpdate();
        if (result == 0)
            System.out.println("update 실패");
        */

        // 레코드 삭제
        /*
        String sqldel = "delete from sangdata where code=?";
        pstmt = conn.prepareStatement(sqldel);
        pstmt.setInt(1, 9);
        int result2 = pstmt.executeUpdate();
        if (result2 == 0)
            System.out.println("delete 실패");
        */
    }
}

```

```

desc";

pstmt = conn.prepareStatement("select * from sangdata"); // 전체 자료 읽기
rs = pstmt.executeQuery();
//String sql = "select code as 코드, sang, su, dan + 1000 from sangdata order by dan

//rs = stmt.executeQuery(sql);
while (rs.next()) {
    String code = rs.getString("code");
    String sang = rs.getString("sang");
    String su = rs.getString("su");
    String dan = rs.getString("dan");
    //String code = rs.getString("코드");
    //String sang = rs.getString(2);
    //String su = rs.getString(3);
    //String dan = rs.getString(4);

    System.out.println(code + " " + sang + " " + su + " " + dan);
}

pstmt = conn.prepareStatement("select count(*) from sangdata");
rs = pstmt.executeQuery();
rs.next();
System.out.println("건수 : " + rs.getInt(1));
} catch (Exception e) {
    System.out.println("sql오류:" + e);
} finally {
    try {
        if (rs != null) rs.close();
        if (pstmt != null) pstmt.close();
        if (conn != null) conn.close();
    } catch (Exception e2) {
    }
}

}

public static void main(String[] args) {
    new DbTest2();
}
}

```


DbTest3.java : PreparedStatement 클래스 사용 – 연결 정보는 별도의 클래스로 작성 후 포함 관계로 호출

DbConnection.java : DB와의 연결 정보를 가진 별도의 클래스 작성

```
package pack;
```

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
```

```
public class DbConnection {
```

```
    Connection conn; // DB 연결
```

```
    public DbConnection() {
```

```
        try {
```

```
            // Driver 로딩 -----
```

```
            // 방법1 : MariaDB용 Driver 파일을 이용
```

```
            Class.forName("org.mariadb.jdbc.Driver");
```

```
            // 방법2 : MySQL용 Driver 파일을 이용
```

```
            // Class.forName("com.mysql.jdbc.Driver");
```

```
            // 방법3 : Oracle용 Driver 파일을 이용
```

```
            // Class.forName("oracle.jdbc.driver.OracleDriver");
```

```
        } catch (Exception e) {
```

```
            System.out.println("드라이버 로딩 실패:" + e);
```

```
            return;
```

```
        }
```

```
        connect();
```

```
    }
```

```
    private void connect() {
```

```
        try {
```

```
            // DB 연결
```

```
            // 방법1 : MariaDB
```

```
            conn = DriverManager.getConnection("jdbc:mariadb://localhost:3306/test", "root", "123");
```

```
            //conn = DriverManager.getConnection("jdbc:mariadb://127.0.0.1:3306/test", "root",
```

```
"123"); //localhost와 동일
```

```
            //conn = DriverManager.getConnection("jdbc:mariadb://211.123.100.56:3306/test", "root",
```

```
"123"); //현재 컴의 ip 주소 사용
```

```
            // 방법2 : MySQL
```

```
            // conn = DriverManager.getConnection("jdbc:mysql://@211.183.3.200:3306/test", "root",
```

```
"123");
```

```

        // 방법2 : oracle
        // conn = DriverManager.getConnection("jdbc:oracle:thin:@211.183.3.200:1521:orcl",
"scott", "tiger");
    } catch (Exception e) {
        System.out.println("DB 연결 실패:" + e);
        return;
    }
}
}
}

```

DbTest3.java

```

package pack;
import java.sql.*;

public class DbTest3 {
    PreparedStatement pstmt;
    ResultSet rs;    // select의 결과에 접근
    DbConnection dbConnection;

    public DbTest3() {
        dbConnection = new DbConnection();    // 클래스의 포함 관계. 상속으로 처리할 수도 있다.
    }

    public void processDb() {
        try {
            pstmt = dbConnection.conn.prepareStatement("select * from sangdata");
            rs = pstmt.executeQuery();

            while (rs.next()) {
                String code = rs.getString("code");
                String sang = rs.getString("sang");
                String su = rs.getString("su");
                String dan = rs.getString("dan");

                System.out.println(code + " " + sang + " " + su + " " + dan);
            }

            pstmt = dbConnection.conn.prepareStatement("select count(*) from sangdata");
            rs = pstmt.executeQuery();
            rs.next();
            System.out.println("건수 : " + rs.getInt(1));
        } catch (Exception e) {

```

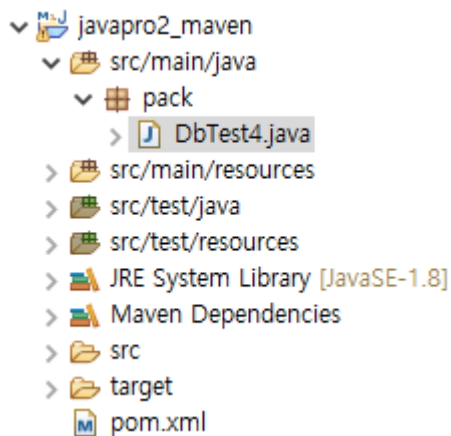
```

        System.out.println("sql오류:" + e);
    } finally {
        try {
            if (rs != null)      rs.close();
            if (pstmt != null) pstmt.close();
            if (dbConnection.conn != null) dbConnection.conn.close();
        } catch (Exception e2) {
        }
    }
}

public static void main(String[] args) {
    new DbTest3().processDb();
}
}

```

DbTest4.java : PreparedStatement 클래스 사용 - maven Project로 작성



jdbcProject실습방법.docx 파일을 참조하세요.

pom.xml

```
<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">
  <modelVersion>4.0.0</modelVersion>
  <groupId>aa</groupId>
  <artifactId>javapro2_maven</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <dependencies>
    <dependency>
      <groupId>org.mariadb.jdbc</groupId>
      <artifactId>mariadb-java-client</artifactId>
      <version>3.3.0</version>
    </dependency>
  </dependencies>
</project>
```

DbTest4.java

```
package pack;
import java.sql.*;

public class DbTest4 {
    Connection conn; // DB 연결
    PreparedStatement pstmt;
    ResultSet rs;    // select의 결과에 접근

    public DbTest4() {
```

```

try {
    // Driver 로딩 -----
    // 방법1 : MariaDB용 Driver 파일을 이용
    Class.forName("org.mariadb.jdbc.Driver");

    // 방법2 : MySQL용 Driver 파일을 이용
    // Class.forName("com.mysql.jdbc.Driver");

    // 방법3 : Oracle용 Driver 파일을 이용
    // Class.forName("oracle.jdbc.driver.OracleDriver");
} catch (Exception e) {
    System.out.println("드라이버 로딩 실패:" + e);
    return;
}

connect();
}

private void connect() {
    try {
        // DB 연결
        // 방법1 : MariaDB
        conn = DriverManager.getConnection("jdbc:mariadb://localhost:3306/test", "root", "123");
    } catch (Exception e) {
        System.out.println("DB 연결 실패:" + e);
        return;
    }

    try {
        // 레코드 추가 - MariaDB 또는 MySql
        /*
        String sqlins = "insert into sangdata values(?,?,?,?)";
        //oracle의 sequence를 사용할 경우
        //String sqlins = "insert into sangdata values(code_seq.nextval,?,?,?)";

        pstmt = conn.prepareStatement(sqlins);
        pstmt.setInt(1, 9);
        pstmt.setString(2, "아메리카노");
        pstmt.setInt(3, 123);
        pstmt.setString(4, "5000");
        pstmt.executeUpdate(); //select 이외의 sql문
        */
    }
}

```

```

// 레코드 수정
/*
String sqlup = "update sangdata set sang=?,su=? where code=?";
pstmt = conn.prepareStatement(sqlup);
pstmt.setString(1, "아메리카노");
pstmt.setInt(2, 123);
pstmt.setInt(3, 2);    // 2번 레코드가 수정 대상
int result = pstmt.executeUpdate();
if (result == 0)
    System.out.println("update 실패");
*/

// 레코드 삭제
/*
String sqldel = "delete from sangdata where code=?";
pstmt = conn.prepareStatement(sqldel);
pstmt.setInt(1, 9);
int result2 = pstmt.executeUpdate();
if (result2 == 0)
    System.out.println("delete 실패");
*/

pstmt = conn.prepareStatement("select * from sangdata");
rs = pstmt.executeQuery();

while (rs.next()) {
    String code = rs.getString("code");
    String sang = rs.getString("sang");
    String su = rs.getString("su");
    String dan = rs.getString("dan");

    System.out.println(code + " " + sang + " " + su + " " + dan);
}

pstmt = conn.prepareStatement("select count(*) from sangdata");
rs = pstmt.executeQuery();
rs.next();
System.out.println("건수 : " + rs.getInt(1));
} catch (Exception e) {
    System.out.println("sql오류:" + e);
} finally {
    try {

```

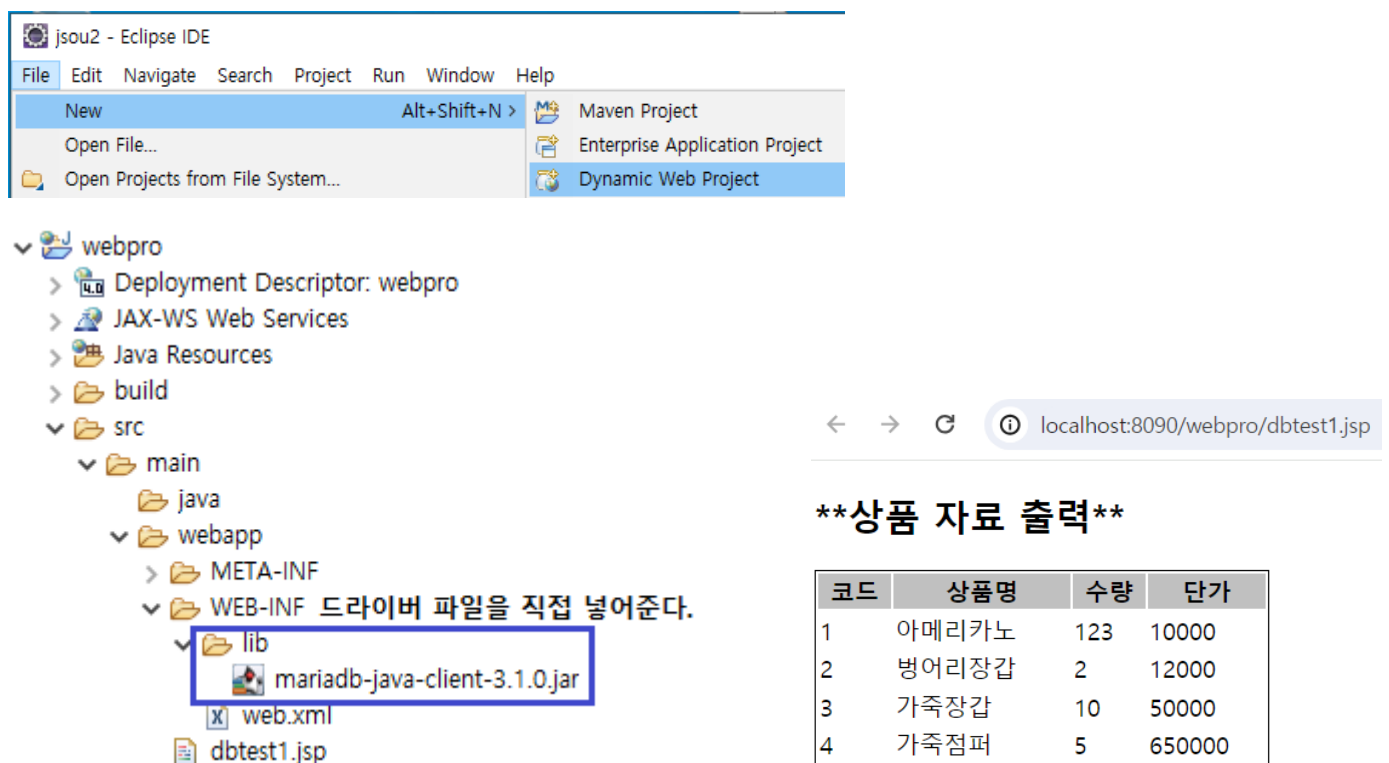
```

        if (rs != null)      rs.close();
        if (pstmt != null) pstmt.close();
        if (conn != null) conn.close();
    } catch (Exception e2) {
    }
}

public static void main(String[] args) {
    new DbTest4();
}
}

```

2) 웹 프로그래밍 : JSP



webpro

- Deployment Descriptor: webpro
- JAX-WS Web Services
- Java Resources
- build
- src
 - main
 - java
 - webapp
 - META-INF
 - WEB-INF 드라이버 파일을 직접 넣어준다.
 - lib
 - mariadb-java-client-3.1.0.jar
 - web.xml
 - dbtest1.jsp

localhost:8090/webpro/dbtest1.jsp

****상품 자료 출력****

코드	상품명	수량	단가
1	아메리카노	123	10000
2	병어리장갑	2	12000
3	가죽장갑	10	50000
4	가죽점퍼	5	650000

dbtest1.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" import="java.sql.*"%>

<%
Connection conn=null;
PreparedStatement pstmt=null;
ResultSet rs=null;
try{
    //MariaDB
    Class.forName("org.mariadb.jdbc.Driver");
    conn=DriverManager.getConnection("jdbc:mariadb://localhost:3306/test","root","123");

    //MySQL
    //Class.forName("com.mysql.jdbc.Driver");
    //conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/test","root","123");

    //Oracle
    //Class.forName("oracle.jdbc.driver.OracleDriver");
    //conn=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:java","scott","tiger");

    pstmt=conn.prepareStatement("select * from sangdata");
}catch(Exception ex){
```



```

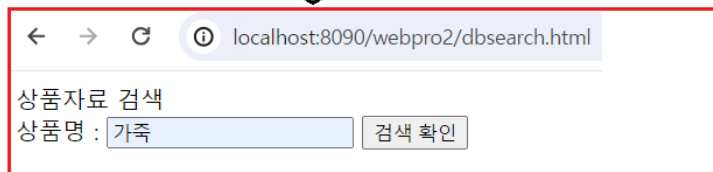
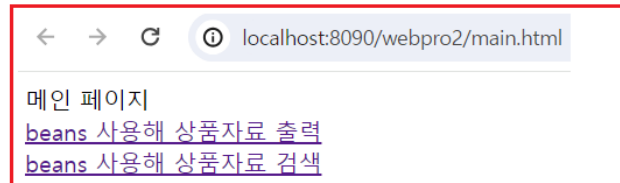
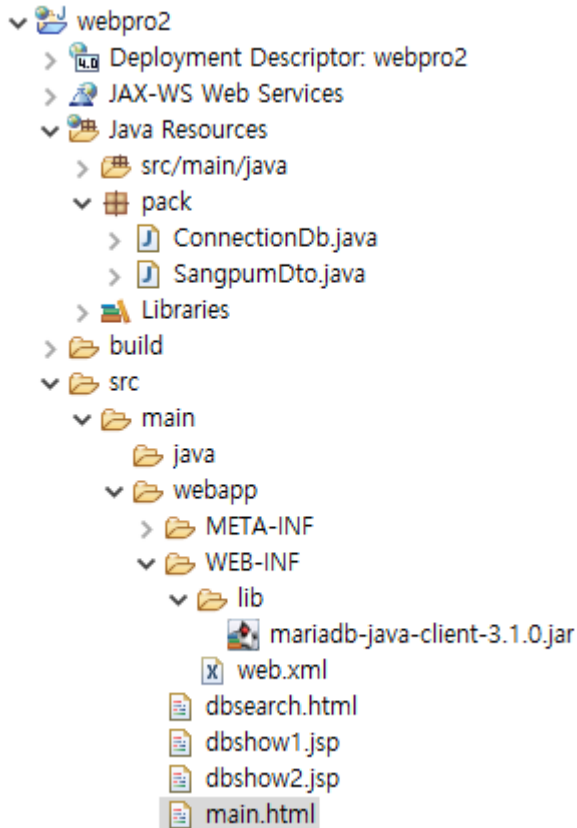
        System.out.println("DB연결 실패:" + ex);
        return;
    }
    try{
        rs=pstmt.executeQuery();
    %>

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h2>**상품 자료 출력**</h2>
    <table style="border: 1px solid black; width: 300px;">
        <tr style="background: silver;">
            <th>코드</th>
            <th>상품명</th>
            <th>수량</th>
            <th>단가</th>
        </tr>
        <%
while(rs.next()){
        %>
        <tr>
            <td><%=rs.getString("code")%></td>
            <td><%=rs.getString("sang")%></td>
            <td><%=rs.getString("su")%></td>
            <td><%=rs.getString("dan")%></td>
        </tr>
        <%
        }
        rs.close(); pstmt.close(); conn.close();
    %>
    }catch(Exception ex){
        System.out.println("SQL 오류:" + ex);
    }
    %>
    </table>
</body>
</html>

```

2) 웹 프로그래밍 : JSP 실습2

상품자료 전체를 출력하고, 상품 이름으로 검색하기



main.html

```
<!DOCTYPE html>
<html>
<body>
메인 페이지 <br>
<a href="dbshow1.jsp">beans 사용해 상품자료 출력</a> <br>
<a href="dbsearch.html">beans 사용해 상품자료 검색</a> <br>
</body>
</html>
```

SangpumDto.java

```
package pack;

public class SangpumDto {
    private String code, sang, su, dan;
    getter, setter 작성
}
```

ConnectionDb.java

```
package pack;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;

public class ConnectionDb {
    private Connection conn;
    private PreparedStatement pstmt;
    private ResultSet rs;

    public ConnectionDb() {
        try {
            // Driver 로딩 -----
            // 방법1 : MariaDB용 Driver 파일을 이용
            Class.forName("org.mariadb.jdbc.Driver");

            // 방법2 : MySQL용 Driver 파일을 이용
            // Class.forName("com.mysql.jdbc.Driver");
            // 방법3 : Oracle용 Driver 파일을 이용
            // Class.forName("oracle.jdbc.driver.OracleDriver");
        } catch (Exception e) {
            System.out.println("드라이버 로딩 실패:" + e);
            return;
        }
    }

    public ArrayList<SangpumDto> getDataAll() { // 전체 상품 읽기
        ArrayList<SangpumDto> list = new ArrayList<SangpumDto>();
        try {
            conn = DriverManager.getConnection("jdbc:mariadb://localhost:3306/test", "root", "123");
            String sql = "select * from sangdata";
            pstmt = conn.prepareStatement(sql);
            rs = pstmt.executeQuery();
            while(rs.next()) {
                SangpumDto dto = new SangpumDto();
                dto.setCode(rs.getString("code"));
                dto.setSang(rs.getString("sang"));
                dto.setSu(rs.getString("su"));
                dto.setDan(rs.getString("dan"));
            }
        }
    }
}
```

```

        list.add(dto);
    }
} catch (Exception e) {
    System.out.println("getDataAll error " + e);
} finally {
    try {
        if (rs != null)    rs.close();
        if (pstmt != null) pstmt.close();
        if (conn != null) conn.close();
    } catch (Exception e2) {
        // TODO: handle exception
    }
}
System.out.println(list.size());
return list;
}

```

// 검색용

public ArrayList<SangpumDto> getSearch(String sang){

```

    System.out.println(sang);

```

```

    ArrayList<SangpumDto> list = new ArrayList<SangpumDto>();

```

```

    try {

```

```

        conn = DriverManager.getConnection("jdbc:mariadb://localhost:3306/test", "root", "123");

```

```

        String sql = "select * from sangdata where sang like ?"; // like 연산자 사용 ex: sang

```

like '가죽%'

```

        //String sql = "select * from sangdata where su >= ?";    수량으로 검색하는 경우

```

```

        pstmt = conn.prepareStatement(sql);

```

```

        pstmt.setString(1, sang + "%"); //

```

```

        rs = pstmt.executeQuery();

```

```

        while(rs.next()) {

```

```

            SangpumDto dto = new SangpumDto();

```

```

            dto.setCode(rs.getString("code"));

```

```

            dto.setSang(rs.getString("sang"));

```

```

            dto.setSu(rs.getString("su"));

```

```

            dto.setDan(rs.getString("dan"));

```

```

            list.add(dto);

```

```

        }

```

```

    } catch (Exception e) {

```

```

        System.out.println("getSearch error " + e);

```

```

    } finally {

```

```

        try {

```

```

            if (rs != null)    rs.close();

```

```

        if (pstmt != null) pstmt.close();
        if (conn != null) conn.close();
    } catch (Exception e2) {
        // TODO: handle exception
    }
}
System.out.println(list.size());
return list;
}
}

```

dbshow1.jsp

```

<%@page import="pack.SangpumDto"%>
<%@page import="java.util.ArrayList"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<jsp:useBean id="connClass" class="pack.ConnectionDb" />

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h2>상품자료 : jsp : beans 사용</h2>
<table style="border: 1px solid black; width: 300px;">
    <tr style="background: silver;">
        <th>코드</th>
        <th>상품명</th>
        <th>수량</th>
        <th>단가</th>
    </tr>
    <%
        ArrayList<SangpumDto> list = connClass.getDataAll();
        //System.out.println(list.size());
        for (SangpumDto s : list) {
    %>
    <tr>
        <td><%=s.getCode()%></td>
        <td><%=s.getSang()%></td>

```

```

        <td><%=s.getSu()%></td>
        <td><%=s.getDan()%></td>
    </tr>
</table>
</body>
</html>

```

dbsearch.html

```

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<script type="text/javascript">
window.onload = function(){
    document.querySelector("#btnClick").onclick = func1;
}

function func1(){
    //alert("good");
    if(frm.sang.value === "") {
        alert("검색 상품명을 입력");
        return;
    }
    frm.submit();
}
</script>
</head>
<body>상품자료 검색<br>
<form action="dbshow2.jsp" name="frm" method="get">
    상품명 : <input type="text" name="sang">
    <input type="button" id="btnClick" value="검색 확인">
</form>
</body>
</html>

```

dbshow2.jsp

```

<%@page import="pack.SangpumDto"%>
<%@page import="java.util.ArrayList"%>

```

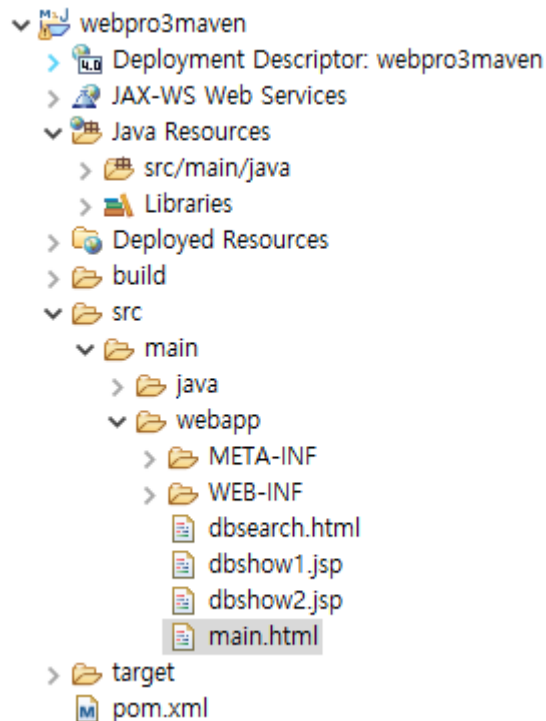
```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<jsp:useBean id="connClass" class="pack.ConnectionDb" />
<%
String sang = request.getParameter("sang");
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h2>상품자료 : 검색 결과</h2>
<table style="border: 1px solid black; width: 300px;">
    <tr style="background: silver;">
        <th>코드 </th>
        <th>상품명 </th>
        <th>수량 </th>
        <th>단가 </th>
    </tr>
    <%
ArrayList<SangpumDto> list = connClass.getSearch(sang);
//System.out.println("검색 후 :" + list.size());
for (SangpumDto s : list) {
%>
    <tr>
        <td><%=s.getCode()%> </td>
        <td><%=s.getSang()%> </td>
        <td><%=s.getSu()%> </td>
        <td><%=s.getDan()%> </td>
    </tr>
    <%
}
%>
</table>
</body>
</html>

```

3) 웹 프로그래밍 : JSP 실습2 - 1

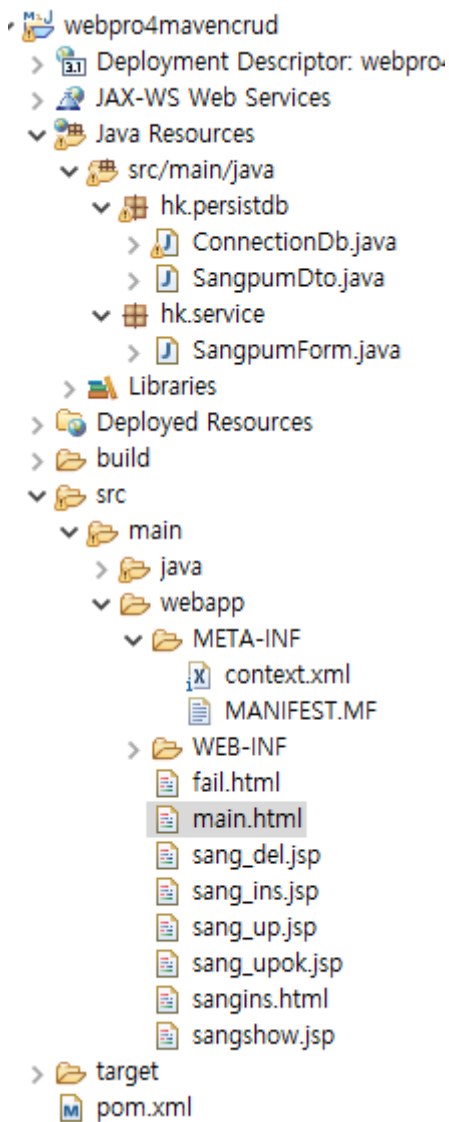
위에서 작업한 소스를 **maven project**로 작성한다면 pom.xml에 **dependency** 설정만 해주면 된다.
나머지는 위와 같다.



pom.xml

```
<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">
    <modelVersion>4.0.0</modelVersion>
    <groupId>webpro3maven</groupId>
    <artifactId>webpro3maven</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>war</packaging>
    <build>
        ...
    </build>
    <dependencies>
        <dependency>
            <groupId>org.mariadb.jdbc</groupId>
            <artifactId>mariadb-java-client</artifactId>
            <version>3.3.0</version>
        </dependency>
    </dependencies>
</project>
```


4) 웹 프로그래밍 : JSP – CRUD(select, insert, update, delete)



메인 페이지
[상품자료 보기](#)

↓

상품자료 : jsp - beans 사용

[상품 추가](#) [수정](#) [삭제](#)

코드	상품명	수량	단가
1	아메리카노	123	10000
2	병어리장갑	2	12000
3	가죽장갑	10	50000
4	가죽점퍼	5	650000

localhost:8090 내용:
수정할 코드 입력

* 상품 수정 *

코드 : 3

상품명 :

수량 :

단가 :

pom.xml

```
<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">
...
</build>
<dependencies>
    <dependency>
        <groupId>org.mariadb.jdbc</groupId>
        <artifactId>mariadb-java-client</artifactId>
        <version>3.3.0</version>
    </dependency>
</dependencies>
</project>
```

SangpumDto.java : DB와 관련된 작업에서 레코드 단위로 자료를 전달할 때 사용

```
package hk.persistdb;
```

```
public class SangpumDto {  
    private String code, sang, su, dan;  
  
    getter, setter  
}
```

SangpumForm.java : 유저를 통해 레코드 단위로 자료를 전달받을 때 사용. 내용이 다르지만 용도가 다름

```
package hk.persistdb;
```

```
public class SangpumForm {  
    private String code, sang, su, dan;  
  
    getter, setter  
}
```

META-INF/context.xml

```
<?xml version="1.0" encoding="UTF-8"?>  
<Context>  
    <Resource name="jdbc_maria" auth="Container" type="javax.sql.DataSource"  
        driverClassName="org.mariadb.jdbc.Driver" loginTimeout="10" maxWait="5000"  
        username="root" password="123" testOnBorrow="true"  
        maxActive="500" maxIdle="100"  
        url="jdbc:mariadb://localhost:3306/test" />  
</Context>
```

ConnectionDb.java

```
package hk.persistdb;
```

```
import java.sql.*;  
import java.util.ArrayList;  
import javax.naming.Context;  
import javax.naming.InitialContext;  
import javax.sql.DataSource;  
import hk.service.SangpumForm;
```

```
public class ConnectionDb {  
    private Connection conn;
```

```
private PreparedStatement pstmt;
private ResultSet rs;
```

```
//커넥션 풀 (DBCP) 커넥션을 미리 만들어 캐싱하고 필요시 마다 사용/반환할 수 있다
//DataSource 가 커넥션 풀을 관리한다
private DataSource dataSource;
```

```
public ConnectionDb() {
```

```
    try {
```

```
        // Driver 로딩 : 기본방법 1 -----
```

```
        // 방법 1 : MariaDB 용 Driver 파일을 이용
```

```
        Class.forName("org.mariadb.jdbc.Driver");
```

```
        // 방법 2 : MySQL 용 Driver 파일을 이용
```

```
        // Class.forName("com.mysql.jdbc.Driver");
```

```
        // 방법 3 : Oracle 용 Driver 파일을 이용
```

```
        // Class.forName("oracle.jdbc.driver.OracleDriver");
```

```
        // Driver 로딩 및 DB 연결방법 2 : Connection pooling 처리를 위해 JNDI 사용
```

```
        // 참조 페이지   https://cafe.daum.net/flowlife/HqLp/8
```

```
        Context ctx = new InitialContext();
```

```
        dataSource = (DataSource)ctx.lookup("java:comp/env/jdbc_maria");
```

```
        // java:comp/env/는 예약어임. 여기에 context.xml 의 Resource
```

```
name="jdbc_maria"부분을 적는다.
```

```
    } catch (Exception e) {
```

```
        System.out.println("드라이버 로딩 실패:" + e);
```

```
        return;
```

```
    }
```

```
}
```

```
public ArrayList<SangpumDto> getDataAll(){
```

```
    ArrayList<SangpumDto> list = new ArrayList<SangpumDto>();
```

```
    try {
```

```
        //db 연결 기본방법
```

```
        //conn = DriverManager.getConnection("jdbc:mariadb://localhost:3306/test", "root",
```

```
"123");
```

```
        //db 연결 DBCP 를 사용하는 경우
```

```
        conn = dataSource.getConnection();
```

```

String sql = "select * from sangdata";
pstmt = conn.prepareStatement(sql);
rs = pstmt.executeQuery();
while(rs.next()) {
    SangpumDto dto = new SangpumDto();
    dto.setCode(rs.getString("code"));
    dto.setSang(rs.getString("sang"));
    dto.setSu(rs.getString("su"));
    dto.setDan(rs.getString("dan"));
    list.add(dto);
}
} catch (Exception e) {
    System.out.println("getDataAll error " + e);
} finally {
    try {
        if (rs != null) rs.close();
        if (pstmt != null) pstmt.close();
        if (conn != null) conn.close();
    } catch (Exception e2) {
        // TODO: handle exception
    }
}
System.out.println(list.size());
return list;
}

```

// 추가용

public void insertData(String sang, String su, String dan){

System.out.println(sang);

try {

//conn = DriverManager.getConnection("jdbc:mariadb://localhost:3306/test", "root",

"123");

conn = dataSource.getConnection();

//새로운 상품 code 구하기 (현재 레코드의 가장 큰 code 를 읽어 code + 1 해준다.

String sql = "select max(code) from sangdata";

pstmt = conn.prepareStatement(sql);

rs = pstmt.executeQuery();

int maxCode = 0;

if(rs.next()) {

maxCode = rs.getInt(1);

```

    }
    System.out.println("maxCode : " + maxCode);

    // 새 상품 추가하기
    pstmt = conn.prepareStatement("insert into sangdata values(?,?,?)");
    pstmt.setInt(1, maxCode + 1); // 입력되는 값의 타입이 숫자이므로 setInt()
    pstmt.setString(2, sang); // 입력되는 값의 타입이 문자이므로 setString()
    pstmt.setString(3, su); // 입력되는 값의 타입이 문자이므로 setString()
    pstmt.setString(4, dan);
    pstmt.executeUpdate();
} catch (Exception e) {
    System.out.println("insertData error " + e);
} finally {
    try {
        if (rs != null) rs.close();
        if (pstmt != null) pstmt.close();
        if (conn != null) conn.close();
    } catch (Exception e2) {
        // TODO: handle exception
    }
}
}
}

```

```

public SangpumDto selectPart(String code){
    SangpumDto dto = null;
    try {
        //db 연결 DBCP 를 사용하는 경우
        conn = dataSource.getConnection();

        String sql = "select * from sangdata where code=?";
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, code);

        rs = pstmt.executeQuery();
        if(rs.next()) {
            dto = new SangpumDto();
            dto.setCode(rs.getString("code"));
            dto.setSang(rs.getString("sang"));
            dto.setSu(rs.getString("su"));
            dto.setDan(rs.getString("dan"));
        }
    } catch (Exception e) {

```

```

        System.out.println("selectPart error " + e);
    } finally {
        try {
            if (rs != null)    rs.close();
            if (pstmt != null) pstmt.close();
            if (conn != null) conn.close();
        } catch (Exception e2) {
            // TODO: handle exception
        }
    }
    return dto;
}

```

public boolean updateData(SangpumForm form) {

```

    boolean b = false;
    //System.out.println("상품명 : " + form.getSang());
    try {
        conn = dataSource.getConnection();
        String sql = "update sangdata set sang=?,su=?,dan=? where code=?";
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, form.getSang());
        pstmt.setString(2, form.getSu());
        pstmt.setString(3, form.getDan());
        pstmt.setString(4, form.getCode());
        if(pstmt.executeUpdate() > 0) b = true;
    } catch (Exception e) {
        System.out.println("updateData error " + e);
    } finally {
        try {
            if (rs != null)    rs.close();
            if (pstmt != null) pstmt.close();
            if (conn != null) conn.close();
        } catch (Exception e2) {
            // TODO: handle exception
        }
    }
    return b;
}

```

public boolean deleteData(String code) {

```

    boolean b = false;

```

```

        try {
            conn = dataSource.getConnection();
            String sql = "delete from sangdata where code=?";
            pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, code);
            if(pstmt.executeUpdate() > 0) b = true;
        } catch (Exception e) {
            System.out.println("deleteData error " + e);
        } finally {
            try {
                if (pstmt != null) pstmt.close();
                if (conn != null) conn.close();
            } catch (Exception e2) {
                // TODO: handle exception
            }
        }
        return b;
    }
}

```

main.html

```

<!DOCTYPE html>
<html>
<body>
메인 페이지 <br><a href="sangshow.jsp">상품자료 읽기</a><br>
</body>
</html>

```

sangshow.jsp

```

<%@page import="hk.persistdb.SangpumDto"%>
<%@page import="java.util.ArrayList"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<jsp:useBean id="connClass" class="hk.persistdb.ConnectionDb" scope="page" />

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<script type="text/javascript">

```

```
function funcDel(){
    //alert("b");
    let code = prompt("수정할 코드 입력");
    if(code != "" && code != null) {
        if(confirm("정말 삭제할까요?") == true){
            location.href="sang_del.jsp?code=" + code;
        }
    }
}
```



```
</body>
</html>
```

showins.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>상품 추가</title>
<script type="text/javascript">
window.onload = function(){
    document.querySelector("#btnClick").onclick = func1;
}

function func1(){
    //alert("good");
    if(frm.sang.value === "") {    //입력자료 오류검사
        frm.sang.placeholder = "상품명을 입력";
        return;
    }

    let regExp = /^wd/;    //정규 표현식으로 digit만 허용하는 입력자료 오류검사
    if(!frm.su.value.match(regExp)){
        frm.su.value = "";
        frm.su.placeholder = "수량은 숫자만 허용";
        return;
    }

    frm.submit();
}
</script>
</head>
<body>상품 추가<br>
<form action="sang_ins.jsp" name="frm" method="post">
품명 : <input type="text" name="sang"> <br/>
수량 : <input type="text" name="su"> <br/>
단가 : <input type="number" name="dan"> <br/>
<input type="button" id="btnClick" value="상품 등록">
</form>
</body>
</html>
```

sang_ins.jsp

```
<%@page import="hk.persistdb.SangpumDto"%>
<%@page import="java.util.ArrayList"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>

<jsp:useBean id="processDb" class="hk.persistdb.ConnectionDb" scope="page" />

<%
request.setCharacterEncoding("utf-8");
String sang = request.getParameter("sang");
String su = request.getParameter("su");
String dan = request.getParameter("dan");
System.out.println(sang + " " + su + " " + dan);

// 비어있는 값이나 null인 값을 체크해 만족하지 못한 경우 입력화면으로 이동하기
if(sang.equals("") || sang == null ||
    su.equals("") || su == null ||
    dan.equals("") || dan == null){
    response.sendRedirect("sangins.html");
    return;
}

processDb.insertData(sang, su, dan);
//새 상품 추가 후 상품자료 보기를 해야한다.
response.sendRedirect("sangshow.jsp");
%>
```

sang_up.jsp

```
<%@page import="hk.persistdb.SangpumDto"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>

<jsp:useBean id="processDb" class="hk.persistdb.ConnectionDb" />
<% String code = request.getParameter("code"); %>

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
```

```

<title>Insert title here</title>
<script type="text/javascript">
window.onload = function(){
    document.querySelector("#btnOk").onclick=function(){
        //입력자료 검사 생략
        frm.submit();
    };
}
</script>
</head>
<body>
<%
SangpumDto dto = processDb.selectPart(code);
if(dto == null){
%>
    <script>
    alert("등록된 코드가 아닙니다.\n수정 불가!");
    location.href="sangshow.jsp";
    </script>
<%
    return;
}
%>

* 상품 수정 *<br/>
코드 : <%=dto.getCode() %> <br/>
<b><form action="sang_upok.jsp" name="frm" method="post">
    <input type="hidden" name="code" value="<%=dto.getCode() %>">
    품명 : <input type="text" name="sang" value="<%=dto.getSang() %>"> <br/>
    수량 : <input type="text" name="su" value="<%=dto.getSu() %>"> <br/>
    단가 : <input type="number" name="dan" value="<%=dto.getDan() %>"> <br/>
    <input type="button" id="btnOk" value="상품 수정">
    <input type="button" value="수정 취소" onclick="javascript:location.href='sangshow.jsp'">
</form>
</body>
</html>

```

sang_upok.jsp

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%

```

```
//String code = request.getParameter("code");
// 이런 식으로 받을 수 있으나 전송 종류가 많은 경우에는 아래처럼 폼빈을 사용한다.
%>
<% request.setCharacterEncoding("utf-8"); // post 방식일 때 한글깨짐 방지 %>

<jsp:useBean id="bean" class="hk.service.SangpumForm" />
<jsp:setProperty property="*" name="bean" />
<jsp:useBean id="processDb" class="hk.persistdb.ConnectionDb" />

<%
if(processDb.updateData(bean))
    response.sendRedirect("sangshow.jsp"); // 수정 후 전체자료 읽기
else
    response.sendRedirect("fail.html");
%>
```

fail.html

```
<!DOCTYPE html>
<html>
<body>
작업 실패!<br/> <a href="sangshow.jsp">상품 자료</a> 보기
</body>
</html>
```

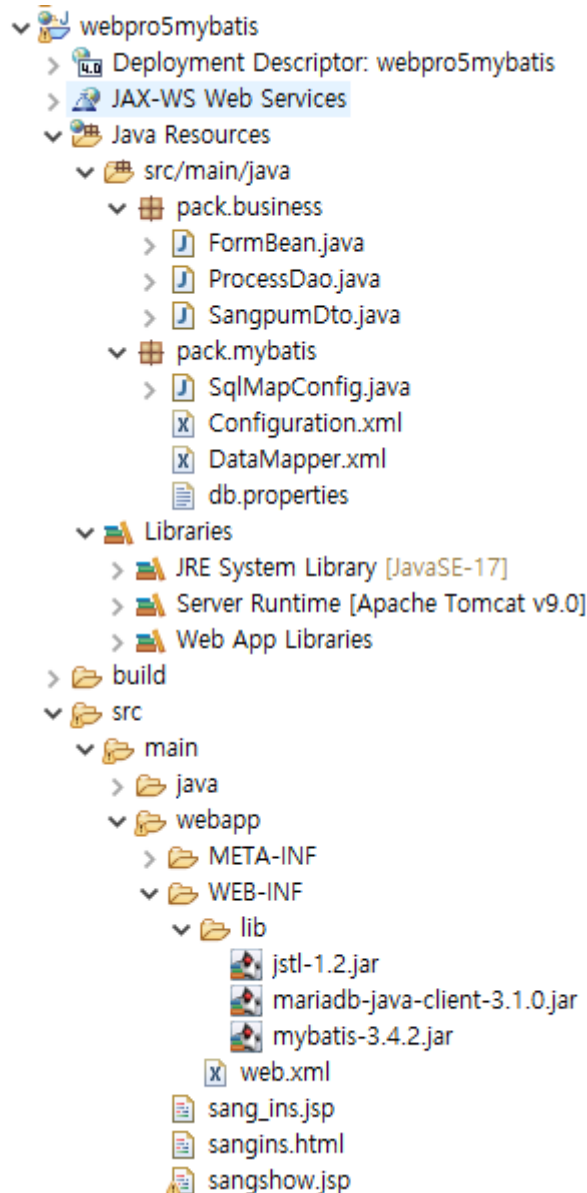
sang_del.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<% String code = request.getParameter("code");
%>

<jsp:useBean id="processDb" class="hk.persistdb.ConnectionDb" />

<%
if(processDb.deleteData(code))
    response.sendRedirect("sangshow.jsp"); // 삭제 후 전체자료 읽기
else
    response.sendRedirect("fail.html");
%>
```

5) 웹 프로그래밍 : JSP – Mybatis framework 사용



SangpumDto.java

```
package pack.business;

public class SangpumDto {
    private String code,sang,su,dan;
    getter, setter
}
```

FormBean.java

```
package pack.business;

public class FormBean {
    private String code,sang,su,dan;
    getter, setter
}
```

db.properties

```
driver=org.mariadb.jdbc.Driver
url=jdbc:mariadb://127.0.0.1:3306/test
username=root
password=123
```

Mapper.xml

```
<?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="dev">
  <select id="selectDataAll" resultType="dto">
    select * from sangdata order by code asc
  </select>

  <select id="selectDataById" parameterType="string" resultType="dto">
    select code,sang,su,dan from sangdata where code = #{code}
  </select>

  <insert id="insertData" parameterType="formbean">
    insert into sangdata(code,sang,su,dan) values(#{code},#{sang},#{su},#{dan})
  </insert>
</mapper>
```

Configuration.xml

```
<?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>
  <properties resource="pack/mybatis/db.properties" />
  <typeAliases>
    <typeAlias type="pack.business.SangpumDto" alias="dto"/>
    <typeAlias type="pack.business.FormBean" alias="formbean"/>
  </typeAliases>
  <environments default="dev">
    <environment id="dev">
      <transactionManager type="JDBC" />
      <dataSource type="POOLED">
        <property name="driver" value="${driver}" />

```

```

        <property name="url" value="${url}" />
        <property name="username" value="${username}" />
        <property name="password" value="${password}" />
    </dataSource>
</environment>
</environments>
<mappers>
    <mapper resource="pack/mybatis/DataMapper.xml" />
</mappers>
</configuration>

```

SqlMapConfig.java

```

package pack.mybatis;
import java.io.Reader;
import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;

public class SqlMapConfig {
    public static SqlSessionFactory sqlSession; // DB의 SQL명령을 실행시킬 때 필요한 메소드를 갖고 있다.

    static {
        String resource = "pack/mybatis/Configuration.xml";
        try {
            Reader reader = Resources.getResourceAsReader(resource);
            sqlSession = new SqlSessionFactoryBuilder().build(reader);
            reader.close();
        } catch (Exception e) {
            System.out.println("SqlMapConfig 오류 : " + e);
        }
    }

    public static SqlSessionFactory getSqlSession() {
        return sqlSession;
    }
}

```

ProcessDao.java

```

package pack.business;
import java.sql.SQLException;

```

```

import java.util.List;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import pack.mybatis.SqlMapConfig;

public class ProcessDao {
    private SqlSessionFactory sessionFactory = SqlMapConfig.getSqlSession();

    public List<SangpumDto> selectDataAll() throws SQLException{
        SqlSession sqlSession = sessionFactory.openSession();
        List<SangpumDto> list = sqlSession.selectList("selectDataAll");
        sqlSession.close();
        return list;
    }

    public SangpumDto selectDataPart(String arg){
        SqlSession sqlSession = sessionFactory.openSession();
        SangpumDto dto = null;
        try {
            dto = sqlSession.selectOne("selectDataByld", arg);
        } catch (Exception e) {
            System.out.println("selectDataPart error : " + e);
        } finally {
            if(sqlSession != null) sqlSession.close();
        }
        return dto;
    }

    public void insertData(FormBean formBean) throws SQLException{
        SqlSession sqlSession = sessionFactory.openSession();

        try {
            sqlSession.insert("insertData", formBean);
            sqlSession.commit();
        } catch (Exception e) {
            System.out.println("insertData error : " + e);
            sqlSession.rollback();
        } finally {
            if(sqlSession != null) sqlSession.close();
        }
    }
}

```


sangshow.jsp

```
<%@page import="pack.business.SangpumDto"%>
<%@page import="java.util.ArrayList"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<jsp:useBean id="processDao" class="pack.business.ProcessDao" />

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h2>상품자료 : jsp - MyBatis 사용</h2>
<a href="sangins.html">상품 추가</a>
<br/>
<table border="1">
    <tr style="background: silver;">
        <th>코드</th>
        <th>상품명</th>
        <th>수량</th>
        <th>단가</th>
    </tr>
    <%
        ArrayList<SangpumDto> list = (ArrayList)processDao.selectDataAll();

        for (SangpumDto s : list) {
    %>
    <tr>
        <td><%=s.getCode()%></td>
        <td><%=s.getSang()%></td>
        <td><%=s.getSu()%></td>
        <td><%=s.getDan()%></td>
    </tr>
    <%
        }
    %>
</table>
<hr>JSTL 사용<br>
<table border="1">
```

```

<tr style="background: silver;">
    <th>코드 </th>
    <th>상품명 </th>
    <th>수량 </th>
    <th>단가 </th>
</tr>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%
ArrayList<SangpumDto> list2 = (ArrayList)processDao.selectAll();
%>
<c:forEach var="s" items="<%=list2 %>">
<tr>
    <td>${s.code}</td>
    <td>${s.sang}</td>
    <td>${s.su}</td>
    <td>${s.dan}</td>
</tr>
</c:forEach>
</table>
</body>
</html>

```

sangins.html

```

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>상품 추가</title>
<script type="text/javascript">
window.onload = function(){
    document.querySelector("#btnClick").onclick = func1;
}

function func1(){
    //alert("good");
    if(frm.sang.value === "") {    //입력자료 오류검사
        frm.sang.placeholder = "상품명을 입력";
        return;
    }

    let regExp = /^wd/;    //정규 표현식으로 digit만 허용하는 입력자료 오류검사
    if(!frm.su.value.match(regExp)){

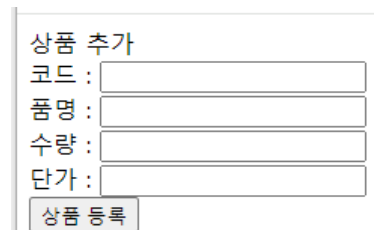
```

```

        frm.su.value = "";
        frm.su.placeholder = "수량은 숫자만 허용";
        return;
    }

    frm.submit();
}
</script>
</head>
<body>
상품 추가<br>
<form action="sang_ins.jsp" name="frm" method="post">
<!-- 사실 code는 프로그래밍 처리해야 하나 생략 -->
코드 : <input type="text" name="code"> <br/>
품명 : <input type="text" name="sang"> <br/>
수량 : <input type="text" name="su"> <br/>
단가 : <input type="number" name="dan"> <br/>
<input type="button" id="btnClick" value="상품 등록">
</form>
</body>
</html>

```



sang_ins.jsp

```

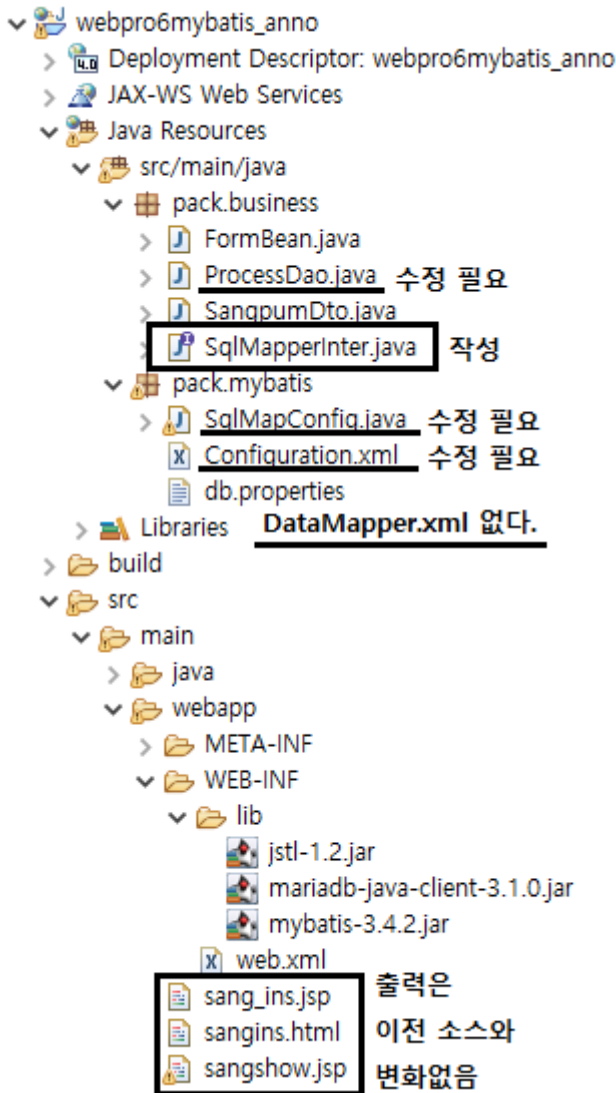
<%@page import="java.util.ArrayList"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>

<% request.setCharacterEncoding("utf-8"); %>
<jsp:useBean id="bean" class="pack.business.FormBean" />
<jsp:setProperty property="*" name="bean" />
<jsp:useBean id="processDao" class="pack.business.ProcessDao" />

<%
processDao.insertData(bean);
//새 상품 추가 후 상품자료 보기를 해야한다.
response.sendRedirect("sangshow.jsp");
%>

```

6) 웹 프로그래밍 : JSP – Mybatis Annotation 사용



SangpumDto, FormBean 파일은 이전과 동일
db.properties 파일 이전과 동일

Configuration.xml

```
<?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>
```

```
    <properties resource="pack/mybatis/db.properties" />
```

```
    <environments default="dev">
```

```
        <environment id="dev">
```

```
            <transactionManager type="JDBC" />
```

```

        <dataSource type="POOLED">
            <property name="driver" value="${driver}" />
            <property name="url" value="${url}" />
            <property name="username" value="${username}" />
            <property name="password" value="${password}" />
        </dataSource>
    </environment>
</environments>
</configuration>

```

SqlMapperInter.java

```

package pack.business;
import java.util.List;
import org.apache.ibatis.annotations.Insert;
import org.apache.ibatis.annotations.Select;

public interface SqlMapperInter {
    @Select("select * from sangdata")
    public List<SangpumDto> selectDataAll();

    @Select("select * from sangdata where code=#{code}")
    public SangpumDto selectDataPart(String code);

    @Insert("insert into sangdata values(#{code},#{sang},#{su},#{dan})")
    public void insertData(FormBean bean);
}

```

SqlMapConfig.java

```

package pack.mybatis;
import java.io.Reader;
import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
import pack.business.SqlMapperInter;

public class SqlMapConfig {
    public static SqlSessionFactory sqlSession;

    static {
        String resource = "pack/mybatis/Configuration.xml";
        try {

```

```

Reader reader = Resources.getResourceAsReader(resource);
sqlSession = new SqlSessionFactoryBuilder().build(reader);
reader.close();

```

//MyBatis Annotation 사용 시 추가할 코드

```

Class[] mappers = {SqlMapperInter.class};
for(Class m:mappers) {
    //Mapper 등록
    sqlSession.getConfiguration().addMapper(m);
}

```

```

} catch (Exception e) {
    System.out.println("SqlMapConfig 오류 : " + e);
}

```

```

}

```

```

public static SqlSessionFactory getSqlSession() {
    return sqlSession;
}

```

```

}

```

ProcessDao.java

```

package pack.business;

```

```

import java.sql.SQLException;

```

```

import java.util.List;

```

```

import org.apache.ibatis.session.SqlSession;

```

```

import org.apache.ibatis.session.SqlSessionFactory;

```

```

import pack.mybatis.SqlMapConfig;

```

```

public class ProcessDao {

```

```

    private SqlSessionFactory sessionFactory = SqlMapConfig.getSqlSession();

```

```

    public List<SangpumDto> selectDataAll(){

```

```

        SqlSession sqlSession = sessionFactory.openSession();

```

```

        List<SangpumDto> list = null;

```

```

        try {

```

```

            SqlMapperInter inter = (SqlMapperInter)sqlSession.getMapper(SqlMapperInter.class);

```

```

            list = inter.selectDataAll();

```

```

        } catch (Exception e) {

```

```

            System.out.println("selectDataAll err : " + e);

```

```

        } finally {

```

```

            if(sqlSession != null) sqlSession.close();

```

```

    }

    return list;
}

public SangpumDto selectDataPart(String arg) throws SQLException{
    SqlSession sqlSession = sessionFactory.openSession();
    SangpumDto dto = null;
    try {
        SqlMapperInter inter = (SqlMapperInter)sqlSession.getMapper(SqlMapperInter.class);
        dto = inter.selectDataPart(arg);
    } catch (Exception e) {
        System.out.println("selectDataPart err : " + e);
    } finally {
        if(sqlSession != null) sqlSession.close();
    }

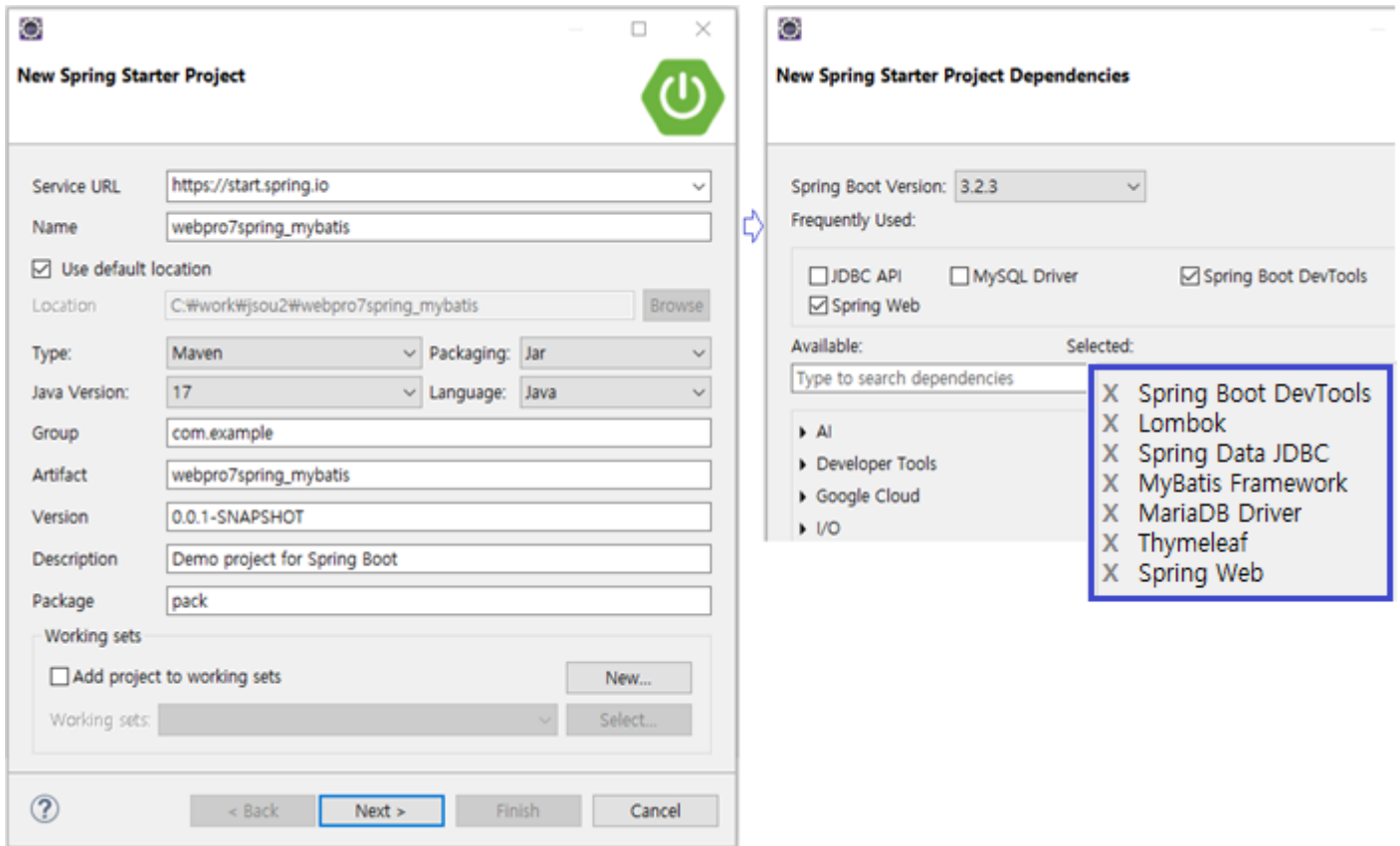
    return dto;
}

public void insertData(FormBean bean) throws SQLException{
    SqlSession sqlSession = sessionFactory.openSession();

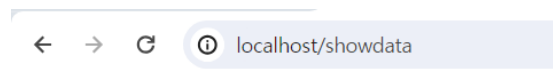
    try {
        SqlMapperInter inter = (SqlMapperInter)sqlSession.getMapper(SqlMapperInter.class);
        inter.insertData(bean);
        sqlSession.commit();
    } catch (Exception e) {
        System.out.println("insertData err : " + e);
        sqlSession.rollback();
    } finally {
        if(sqlSession != null) sqlSession.close();
    }
}
}

```

7) 웹 프로그래밍 : Spring Boot + MyBatis Annotation 사용



- ▼ webpro7spring_mybatis [boot] [devtools]
 - ▼ src/main/java
 - ▼ pack
 - > Webpro7springMybatisApplication.java
 - ▼ pack.controller
 - > FormBean.java
 - > ListController.java
 - ▼ pack.model
 - > DataDao.java
 - > DataMappingInterface.java
 - > SangpumDto.java
 - ▼ src/main/resources
 - ▼ static
 - > index.html
 - ▼ templates
 - > error.html
 - > insert.html
 - > show.html
 - > application.properties
 - > src/test/java
 - > JRE System Library [JavaSE-17]
 - > Maven Dependencies
 - > target/generated-sources/annotations
 - > target/generated-test-sources/test-annotations
 - > src
 - > target
 - > HELP.md
 - > mvnw
 - > mvnw.cmd
 - > pom.xml



직원 목록(Spring + MyBatis + Thymeleaf)

추가

코드	상품명	수량	단가
1	아메리카노	123	10,000
2	병어리장갑	2	12,000
3	가죽장갑	10	50,000
4	가죽점퍼	5	650,000
5	지우개	11	1,700
6	우산	5	25,000
건수 : 6			
상품명 :			검색

application.properties

```
server.port=80
spring.thymeleaf.cache=false
```

```
#mariadb server connect
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
spring.datasource.url=jdbc:mariadb://127.0.0.1:3306/test
spring.datasource.username=root
spring.datasource.password=123
```

SangpumDto.java

```
package pack.model;
import lombok.Data;

@Data
public class SangpumDto {
    private String code, sang, su, dan;
}
```

FormBean.java

```
package pack.controller;
import lombok.Data;

@Data
public class FormBean {
    private String searchValue;

    private String code, sang, su, dan;
}
```

DataMappingInterface.java

```
package pack.model;
import java.util.List;
import org.apache.ibatis.annotations.Insert;
import org.apache.ibatis.annotations.Mapper;
import org.apache.ibatis.annotations.Select;
import pack.controller.FormBean;

@Mapper
public interface DataMappingInterface {
```

```

// 추상메소드의 이름은 mapper.xml의 id와 일치시키자
// 추상메소드의 이름은 mapper.xml의 id명과 일치시켜 준다.
    @Select("select * from sangdata")
    List<SangpumDto> selectAll(); // 전체 자료 출력

    @Select("select code,sang,su,dan from sangdata where sang like concat('%', #{searchValue}, '%')")
    List<SangpumDto> selectSearch(FormBean bean); // 상품명 검색 결과 : like 연산자 사용

    @Insert("insert into sangdata values(#{code},#{sang},#{su},#{dan})")
    int insertData(FormBean bean);
}

```

DataDao.java

```

package pack.model;
import java.util.List;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;
import pack.controller.FormBean;

@Repository
public class DataDao {
    // private Logger logger = LoggerFactory.getLogger(DataDao.class);
    private Logger logger = LoggerFactory.getLogger(this.getClass());

    @Autowired
    private DataMappingInterface dataInterface; // 자동으로 pooling 처리됨. HikariPool

    public List<SangpumDto> getDataAll() {
        List<SangpumDto> list = dataInterface.selectAll();
        logger.info("datas : " + list.size() + "개");
        return list;
    }

    public List<SangpumDto> getDataSearch(FormBean bean) {
        System.out.println(bean.getSearchValue());
        List<SangpumDto> slist = dataInterface.selectSearch(bean);
        logger.info("datas : " + slist.size() + "개");
        return slist;
    }
}

```

```

// 추가
public boolean insert(FormBean bean) {
    // 번호 중복 및 자동 증가 작업은 생략~~~
    System.out.println(bean.getCode());
    int re = dataInterface.insertData(bean);
    System.out.println("re : " + re);
    if(re > 0)
        return true;
    else
        return false;
}
}

```

ListController.java

```

package pack.controller;
import java.util.ArrayList;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import pack.model.DataDao;
import pack.model.SangpumDto;

@Controller
public class ListController {
    @Autowired
    private DataDao dataDao;

    @GetMapping("showdata")
    public String showProcess(Model model) {
        ArrayList<SangpumDto> list = (ArrayList<SangpumDto>) dataDao.getDataAll();
        model.addAttribute("datas", list);
        return "show";
    }

    @GetMapping("search")
    public String searchProcess(FormBean bean, Model model) {
        ArrayList<SangpumDto> slist = (ArrayList<SangpumDto>) dataDao.getDataSearch(bean);
        model.addAttribute("datas", slist);
        return "show";
    }
}

```

```

@GetMapping("insert")
public String insert() {
    return "insert";
}

@PostMapping("insert")
public String insertProcess(FormBean bean) {
    boolean b = dataDao.insert(bean);
    System.out.println("bbbbbb:" + b);
    if(b)
        // return "list" 하면 추가 결과를 바로 볼 수 없다. 그냥 list.html이 호출됨
        // 클라이언트를 통해 호출되어야 하므로 아래처럼 적어 주자.
        return "redirect:http://localhost/showdata";
    else
        return "redirect:http://localhost/error.html";
}
}

```

index.html

```

<!DOCTYPE html>
<html>
<body>
메인 페이지 <br/> <a href="showdata">db 연동 - Spring + MyBatis</a>
</body>
</html>

```

show.html

```

<!DOCTYPE html>
<html xmlns:th="https://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body> <h3>직원 목록(Spring + MyBatis + Thymeleaf)</h3>
<a th:href="@{/insert}">추가</a>
<table border="1">
    <tr> <th>코드</th> <th>상품명</th> <th>수량</th> <th>단가</th> </tr>
    <th:block th:if="${datas.size > 0}">

```

```

<tr th:each="data, numStatus:${datas}">
    <!--
    <td th:text="${numStatus.index + 1}">현재 인덱스 : 0부터 출발</td>
    <td th:text="${numStatus.count}">현재 카운트 : 1부터 출발</td>
    -->
    <td th:text="${data.code}">상품 번호 출력</td>
    <td>[[${data.sang}]]</td>
    <td>[[${data.su}]]</td>
    <td>[[${#numbers.formatInteger(data.dan, 3, 'COMMA')}]</td>
    <!-- number format 참조 : https://gigas-blog.tistory.com/127 -->
</tr>
</th:block>
<tr><td colspan="5">건수 : [[${datas.size}]]</td></tr>
<tr>
    <!-- 검색 작업 추가 -->
    <td colspan="5">
        <form action="/search" method="get">
            상품명 : <input type="text" name="searchValue">
            <input type="submit" value="검 색">
        </form>
    </td>
</tr>
</table>
</body>
</html>

```

insert.html

```

<!DOCTYPE html>
<html>
<body>** 자료 입력 **<br>
<form action="/insert" method="post">
    코드: <input type="text" name="code"><br>
    품명: <input type="text" name="sang"><br>
    수량: <input type="text" name="su"><br>
    단가: <input type="text" name="dan"><br>
    <input type="submit" value="등록">
</form>
</body>
</html>

```

error.html

```

<body>에러 발생!<hr><a href="/showdata">회원목록</a></body>

```

8) 웹 프로그래밍 : Spring Boot + JPA 사용

New Spring Starter Project

Service URL:

Name:

☒ Use default location

Location:

Type: **Gradle - Groovy** Packaging: **Jar**

Java Version: **17** Language: **Java**

Group:

Artifact:

Version:

Description:

Package:

Working sets

☐ Add project to working sets

Working sets:

Spring Boot Version: **3.2.3**

Frequently Used:

☐ JDBC API ☒ Lombok

☐ MyBatis Framework ☐ MySQL Driver

☒ Spring Data JDBC ☒ Spring Web

Available:

Selected:

- X Spring Boot DevTools
- X Lombok
- X Spring Data JPA
- X Spring Data JDBC
- X MariaDB Driver
- X Thymeleaf
- X Spring Web

- webpro8spring_jpa [boot] [devtools]
 - src/main/java
 - pack
 - Webpro8springJpaApplication.java
 - pack.controller
 - FormBean.java
 - ListController.java
 - pack.model
 - DataDao.java
 - SangpumDto.java
 - SangpumRepository.java
 - src/main/resources
 - templates
 - show.html
 - static
 - index.html
 - application.properties
 - src/test/java
 - JRE System Library [JavaSE-17]
 - Project and External Dependencies
 - bin
 - gradle
 - src
 - build.gradle
 - gradlew
 - gradlew.bat
 - HELP.md
 - settings.gradle

← → ↺ ⓘ localhost/showdata

직원 목록(Spring + JPA + Thymeleaf)

코드	상품명	수량	단가
1	아메리카노	123	10,000
2	빙어리장갑	2	12,000
3	가족장갑	10	50,000
4	가족점퍼	5	650,000
5	지우개	11	1,700
6	우산	5	25,000

건수 : 6

상품명 :

build.gradle

```
dependencies {  
    implementation 'org.springframework.boot:spring-boot-starter-data-jdbc'  
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'  
    implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'  
    implementation 'org.springframework.boot:spring-boot-starter-web'  
    compileOnly 'org.projectlombok:lombok'  
    developmentOnly 'org.springframework.boot:spring-boot-devtools'  
    runtimeOnly 'org.mariadb.jdbc:mariadb-java-client'  
    annotationProcessor 'org.projectlombok:lombok'  
    testImplementation 'org.springframework.boot:spring-boot-starter-test'  
}
```

application.properties

```
server.port=80  
spring.thymeleaf.cache=false
```

```
#mariadb server connect
```

```
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver  
spring.datasource.url=jdbc:mariadb://127.0.0.1:3306/test  
spring.datasource.username=root  
spring.datasource.password=123
```

```
# jpa
```

```
spring.jpa.properties.hibernate.show_sql=true  
spring.jpa.properties.hibernate.format_sql=true  
spring.jpa.properties.hibernate.use_sql_comments=true  
logging.level.org.hibernate.type.descriptor.sql=trace  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MariaDBDialect
```

SangpumDto.java

```
package pack.model;  
  
import jakarta.persistence.Column;  
import jakarta.persistence.Entity;  
import jakarta.persistence.Id;  
import jakarta.persistence.Table;  
import lombok.Data;
```

```
@Entity
```

```
@Table(name="sangdata")
```

```

@Data
public class SangpumDto {
    @Id
    @Column(name="code")
    private int code;

    @Column(nullable = false)
    private String sang;

    private int su;
    private int dan;
}

```

FormBean.java

```

package pack.controller;
import lombok.Data;

@Data
public class FormBean {
    private String code,sang,su,dan;
    private String searchValue;
}

```

SangpumRepository.java

```

package pack.model;
import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

public interface SangpumRepository extends JpaRepository<SangpumDto, Integer>{
    // 메소드 사용
    // List<SangpumDto> findBySangContaining(String svalue);

    // JPQL 사용
    @Query(value="select s from SangpumDto s where s.sang like %:svalue%")
    List<SangpumDto> searchLike(@Param("svalue") String svalue);
}

```

DataDao.java

```

package pack.model;
import java.util.List;

```



```

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

@Repository
public class DataDao {

    @Autowired
    private SangpumRepository repository;

    public List<SangpumDto> getDataAll(){
        List<SangpumDto> list = repository.findAll();
        return list;
    }

    public List<SangpumDto> getSearchData(String svalue){
        // 메소드 사용
        //List<SangpumDto> list = repository.findBySangContaining(svalue);

        // JPQL 사용
        List<SangpumDto> list = repository.searchLike(svalue);

        return list;
    }
}

```

ListController.java

```

package pack.controller;
import java.util.ArrayList;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import pack.model.DataDao;
import pack.model.SangpumDto;

@Controller
public class ListController {
    @Autowired
    private DataDao dataDao;

    @GetMapping("showdata")

```

```

    public String listProcess(Model model) {
        ArrayList<SangpumDto> slist = (ArrayList<SangpumDto>)dataDao.getDataAll();
        model.addAttribute("datas", slist);
        return "show";
    }

    @GetMapping("search")
    public String searchProcess(FormBean bean, Model model) {
        ArrayList<SangpumDto> slist =
        (ArrayList<SangpumDto>)dataDao.getSearchData(bean.getSearchValue());
        model.addAttribute("datas", slist);
        return "show";
    }
}

```

index.html

```

<!DOCTYPE html>
<html>
<body>
메인 페이지 <br/> <a href="showdata">db 연동 - Spring + JPA</a>
</body>
</html>

```

show.html

```

<!DOCTYPE html>
<html xmlns:th="https://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body> <h3>직원 목록(Spring + JPA + Thymeleaf)</h3>
<table border="1">
    <tr> <th>코드</th> <th>상품명</th> <th>수량</th> <th>단가</th> </tr>
    <th:block th:if="${datas.size > 0}">
        <tr th:each="data, numStatus:${datas}">
            <td th:text="${data.code}">상품 번호 출력</td>
            <td>[[${data.sang}]]</td>
            <td>[[${data.su}]]</td>
            <td>[[${#numbers.formatInteger(data.dan, 3, 'COMMA')}]</td>
        </tr>
    </th:block>

```

```

<tr>
  <td colspan="5">건수 : [{${datas.size}]}</td>
</tr>
<tr>  <!-- 검색 작업 추가 -->
  <td colspan="5">
    <form action="/search" method="get">
      상품명 : <input type="text" name="searchValue">
      <input type="submit" value="검 색">
    </form>
  </td>
</tr>
</table>
</body>
</html>

```

작성자 소속: 에이콘 아카데미 이름:박영권