

게시판, iBatis

# ORM 프레임워크인 iBATIS

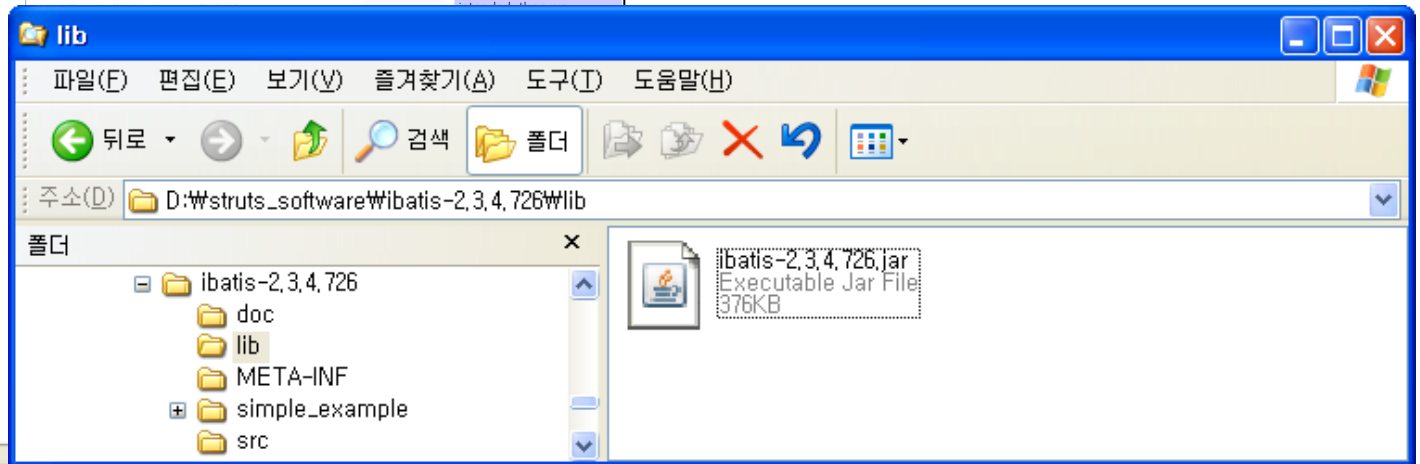
- ORM(Object Relational Mapping) 프레임워크는 데이터베이스와 객체와의 관계를 맵핑시켜 퍼시스턴스 로직처리를 도와주는 프레임워크이다.
- XML 파일에 맵핑 정보를 기술하여 데이터베이스의 테이블과 자바 객체를 맵핑하여 데이터베이스에 생성, 조회, 수정, 삭제(CRUD)작업을 도와주는 역할을 한다.
- ORM 프레임워크는 퍼시스턴스 층에 생산성을 높여 주기 위한 프레임워크로 JDBC를 사용할 때보다 약 60% 정도의 코드만으로 프로그램 작성이 가능해 진다.

## 작업 절차

- JDBC 드라이버 설치
- DBCP 사용을 위한 jar 파일 설치하기
- iBATIS 설치하기
- iBATIS 사용한 JDBC 프로젝트 작성하기
- 데이터베이스 연결 정보를 담은 프로퍼티 파일 작성하기
- SQLMaps 설정파일인 SqlMapConfig.xml 파일 작성하기

# iBATIS 다운받아 설치하기

- <http://ibatis.apache.org/>
- 현재는 mybatis로 변경 됨



# 데이터베이스 연결 정보를 담은 프로퍼티 파일 작성

- 액션 클래스에서 데이터베이스를 연결하고 iBATIS를 사용하도록 하기 위해서 몇 가지 파일들을 생성해서 설정을 해주어야 한다.
- 먼저 데이터베이스 연결 정보를 담은 dbconnect.properties을 작성해 보자.

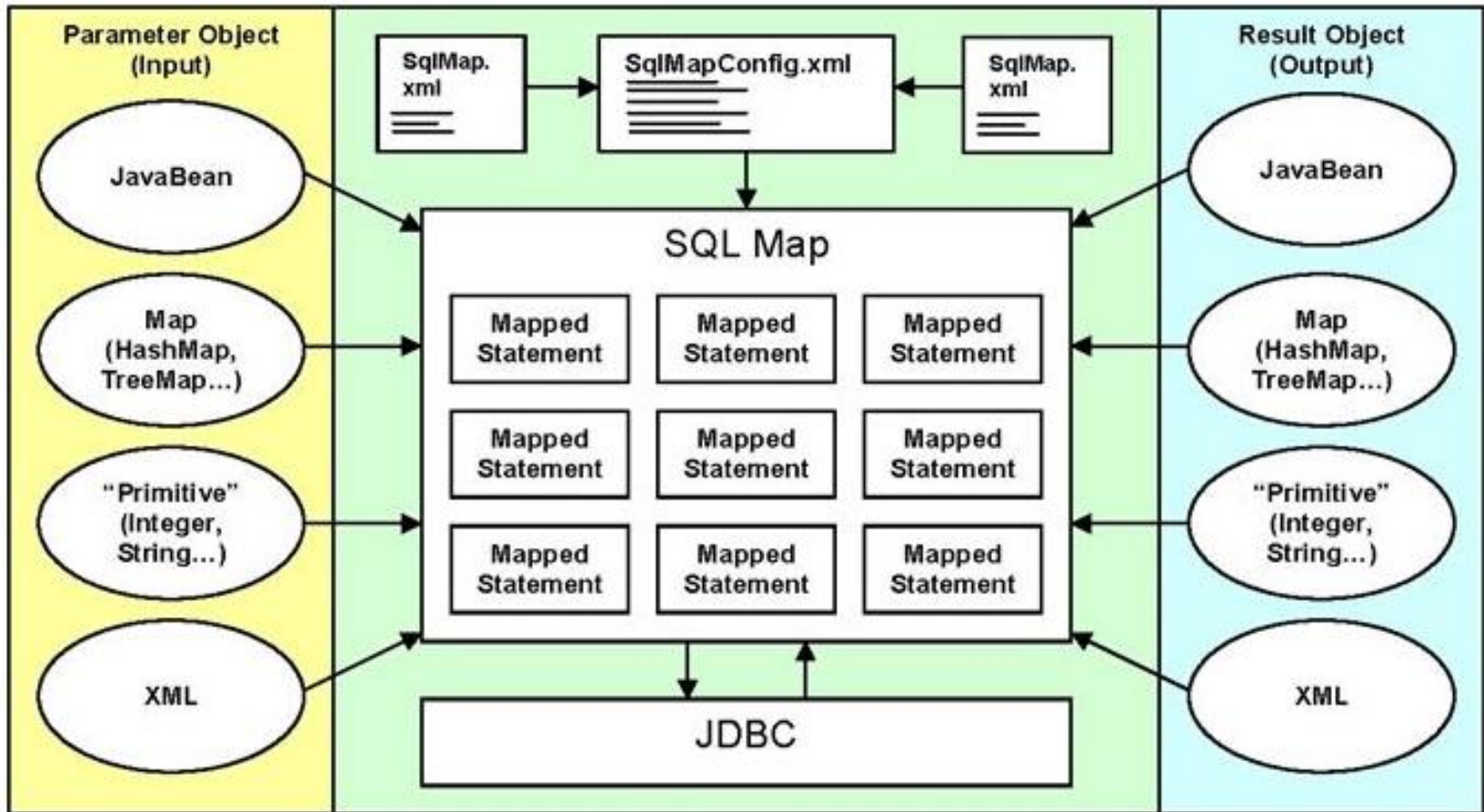
## mysql

```
driver=com.mysql.jdbc.Driver  
url=jdbc:mysql://localhost:3306/test  
username=root  
password=mysql
```

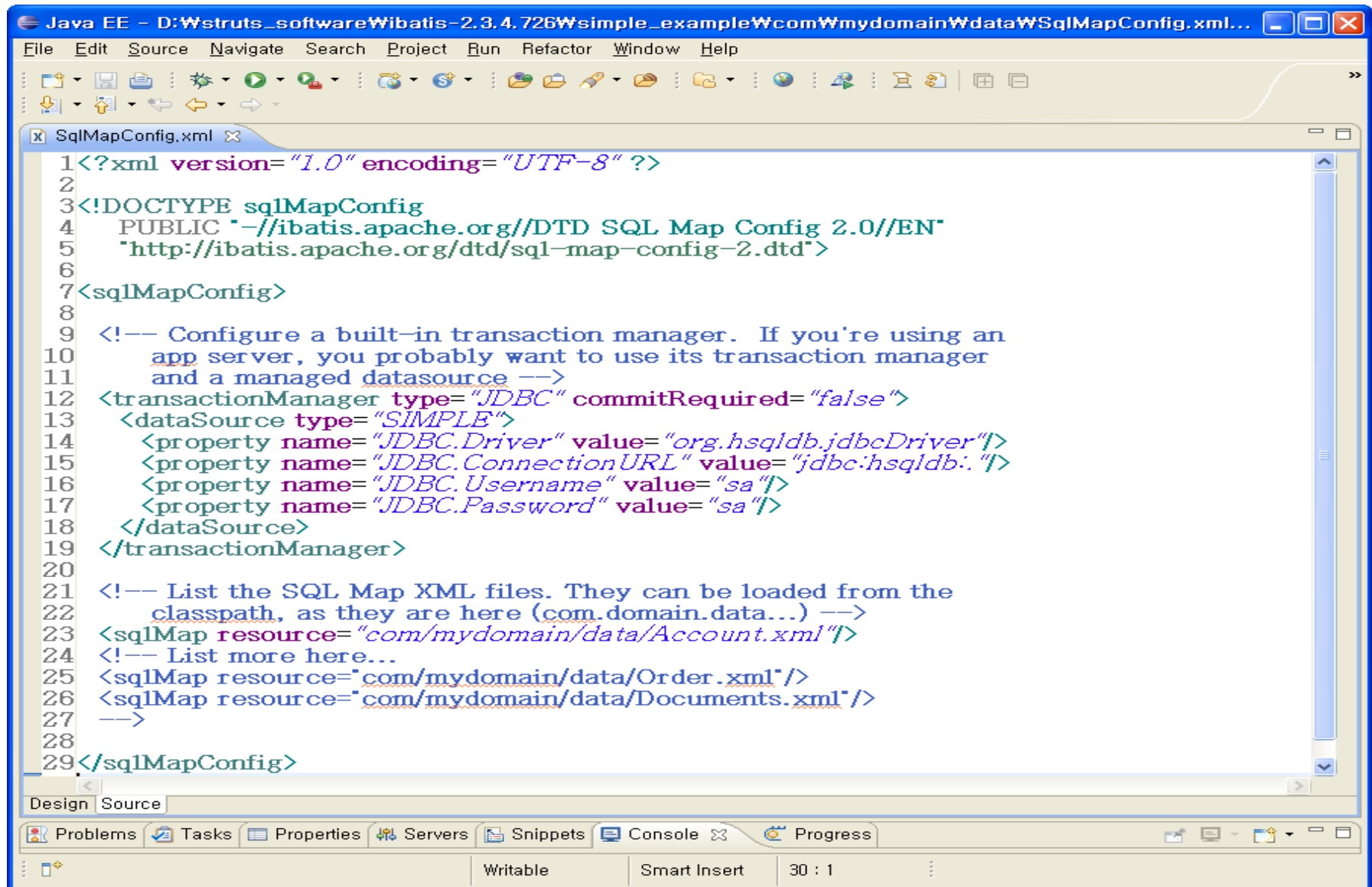
## Oracle

- driver=oracle.jdbc.driver.OracleDriver
- url=jdbc:oracle:thin:@localhost:1521:orcl
- username=scott
- password=tiger

# SQLMaps 설정파일 : SqlMapConfig.xml

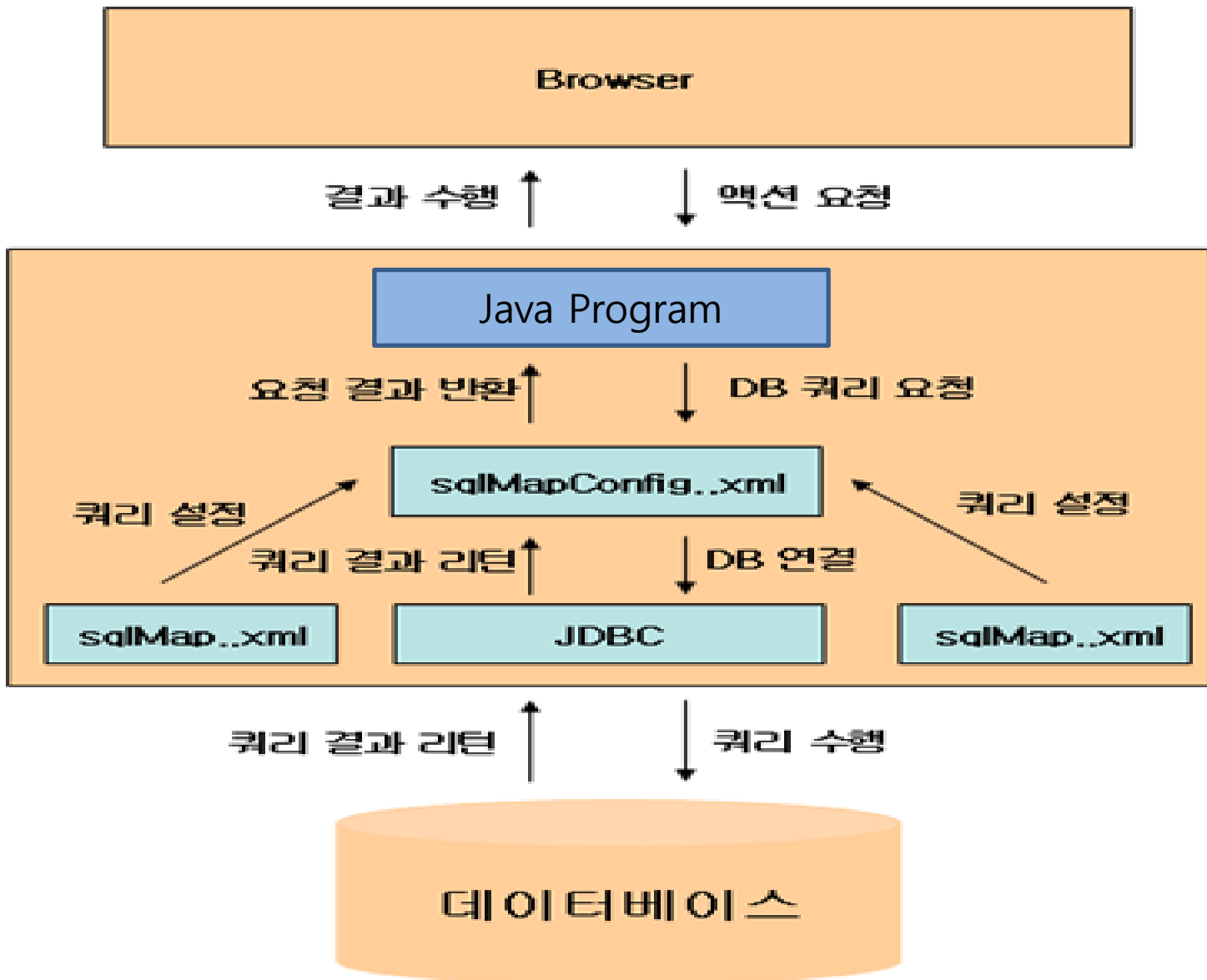


## 3.2 SQLMaps 설정파일 : SqlMapConfig.xml



```
1<?xml version="1.0" encoding="UTF-8" ?>
2
3<!DOCTYPE sqlMapConfig
4  PUBLIC "-//ibatis.apache.org//DTD SQL Map Config 2.0//EN"
5  "http://ibatis.apache.org/dtd/sql-map-config-2.dtd">
6
7<sqlMapConfig>
8
9  <!-- Configure a built-in transaction manager. If you're using an
10   app server, you probably want to use its transaction manager
11   and a managed datasource -->
12  <transactionManager type="JDBC" commitRequired="false">
13    <dataSource type="SIMPLE">
14      <property name="JDBC.Driver" value="org.hsqldb.jdbcDriver"/>
15      <property name="JDBC.ConnectionURL" value="jdbc:hsqldb:."/>
16      <property name="JDBC.Username" value="sa"/>
17      <property name="JDBC.Password" value="sa"/>
18    </dataSource>
19  </transactionManager>
20
21  <!-- List the SQL Map XML files. They can be loaded from the
22   classpath, as they are here (com.domain.data...) -->
23  <sqlMap resource="com/mydomain/data/Account.xml"/>
24  <!-- List more here...
25  <sqlMap resource="com/mydomain/data/Order.xml"/>
26  <sqlMap resource="com/mydomain/data/Documents.xml"/>
27  -->
28
29</sqlMapConfig>
```

# iBatis가 적용한 Spring 웹 애플리케이션의 흐름도



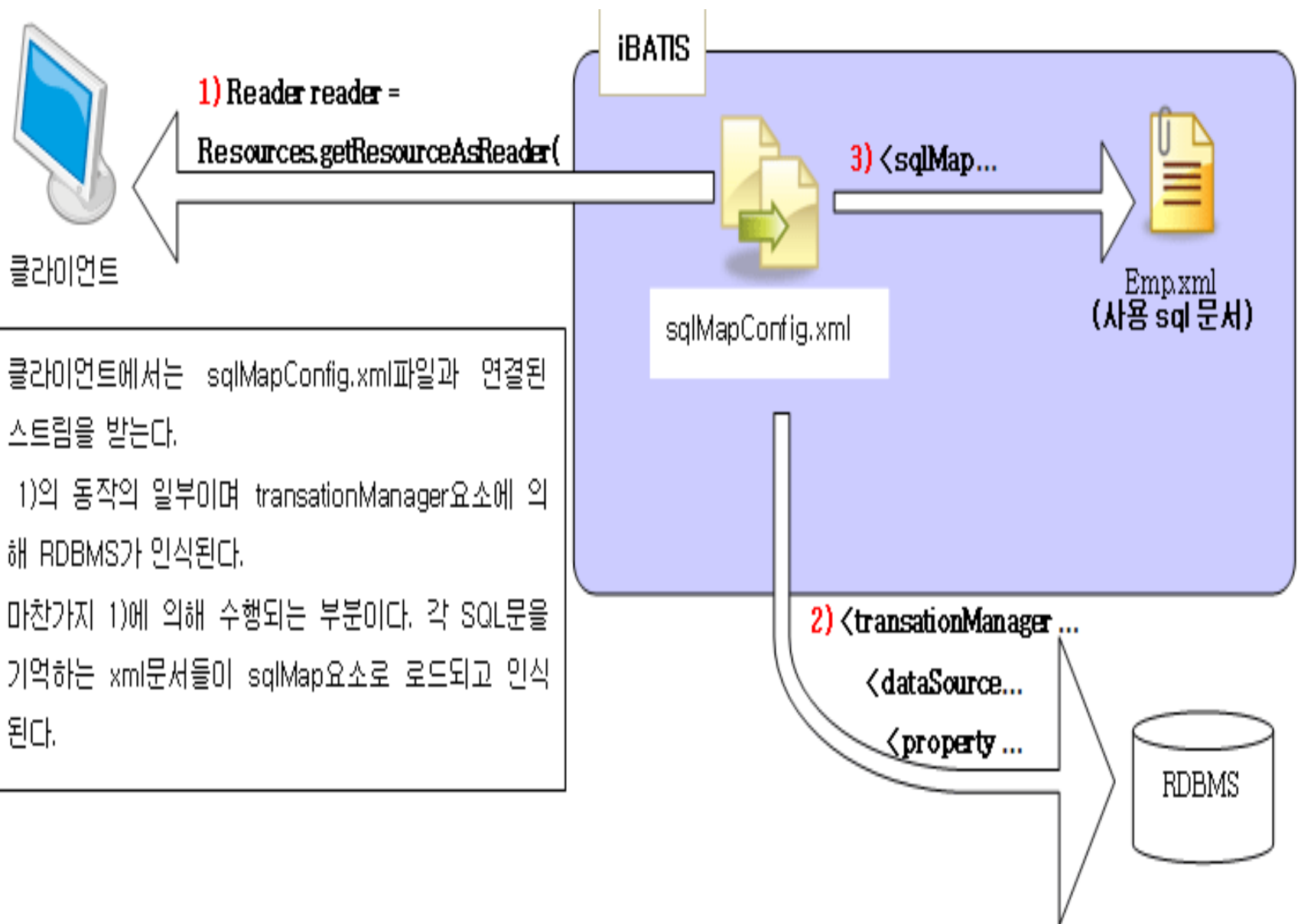
# SQL Map API

## iBATIS란

- ❖클래스를 테이블로 필드를 컬럼으로 직접 매핑하지 않고 SQL 구문과 파라미터와 결과를 클래스 클래스에 매핑
- ❖쿼리를 담은 내용의 XML과 그 쿼리를 실행시키는 자바 코드로 유연하게 클래스와 테이블을 매핑
- ❖클래스의 프로퍼티와 데이터베이스의 테이블간의 파라미터와 결과값을 매핑하는 책임을 맡고 있기에 SQL 매퍼라고 부름



# iBATIS의 동작원리



# 게시판

## Board Table

```
create table board1 (  
    id number(10) primary key,  
    title varchar2(50) not null,  
    writer varchar2(20),  
    content varchar2(500),  
    hit number(10),  
    regDate Date  
);  
insert into board1 values(1,'1','1','1',0,sysdate);
```

# web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
  <!-- 한글 입력 -->
  <filter>
    <filter-name>CharacterEncodingFilter</filter-name>
    <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
    <init-param>
      <param-name>encoding</param-name>
      <param-value>UTF-8</param-value>
    </init-param>
    <init-param>
      <param-name>forceEncoding</param-name>
      <param-value>true</param-value>
    </init-param>
  </filter>
  <filter-mapping>
    <filter-name>CharacterEncodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
```

**<!-- The definition of the Root Spring Container shared by all Servlets and Filters -->**

**<context-param>**

**<param-name>contextConfigLocation</param-name>**

**<param-value>/WEB-INF/spring/root-context.xml</param-value>**

**</context-param>**

**<!-- Creates the Spring Container shared by all Servlets and Filters -->**

**<listener>**

**<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>**

**</listener>**

**<!-- Processes application requests -->**

**<servlet>**

**<servlet-name>appServlet</servlet-name>**

**<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>**

**<init-param>**

**<param-name>contextConfigLocation</param-name>**

**<param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>**

**</init-param>**

**<load-on-startup>1</load-on-startup>**

**</servlet>**

**<servlet-mapping>**

**<servlet-name>appServlet</servlet-name>**

**<url-pattern>\*.do</url-pattern>**

**</servlet-mapping>**

**</web-app>**

# SqlMapConfig.xml 설정의 구성요소

## 1) <properties>

외부에 존재하는 프로퍼티 파일을 사용하기 위해서 사용

**jdbc.properties**

예

```
jdbc.driverClassName=oracle.jdbc.driver.OracleDriver  
jdbc.url=jdbc:oracle:thin:@127.0.0.1:1521:xe  
jdbc.username=scott  
jdbc.password=tiger  
jdbc.maxPoolSize=20
```

# SqlMapConfig.xml 설정의 구성요소

## 2) <settings>

SQLMaps에서 사용되는 다양한 옵션과 최적화를 위해 사용

## 3) <typeAlias>

패키지명을 포함한 클래스가 너무 길 때 각각의 SQLMaps맵핑 파일에서 사용하기 번거로우므로 별칭을 두어서 간단하게 사용

## 4) <transactionManager>

트랜잭션에 관련된 값을 셋팅하고 하위 데이터소스값을 지정하는 <dataSource>를 가진다.

transactionManager의 type속성 값은 JDBC, JTA, EXTERNAL 중에 하나를 사용할 수 있다.

## 5) <sqlMap>

SqlMapConfig.xml의 마지막 부분에서 <sqlMap>요소를 설정.

<sqlMap>요소는 SQLMaps맵핑 파일의 위치를 지정한다.

## ibatis.config/SqlMapConfig.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE sqlMapConfig
    PUBLIC "-//ibatis.apache.org//DTD SQL Map Config 2.0//EN"
    "http://ibatis.apache.org/dtd/sql-map-config-2.dtd">
<sqlMapConfig>
    <properties resource="ibatis/jdbc.properties"/>
    <transactionManager type="JDBC" commitRequired="false">
        <dataSource type="SIMPLE">
            <property name="JDBC.Driver" value="${jdbc.driverClassName}"/>
            <property name="JDBC.ConnectionURL" value="${jdbc.url}"/>
            <property name="JDBC.Username" value="${jdbc.username}"/>
            <property name="JDBC.Password" value="${jdbc.password}"/>
        </dataSource>
    </transactionManager>
    <sqlMap resource="model/Board.xml"/>
</sqlMapConfig>
```

# 자바빈즈 model/Board

```
package model;
import java.util.Date;
public class Board {
    private int id;          private String title;   private String writer;
    private String content;  private int hit;       private Date regDate;
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
    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 int getHit() { return hit; }
    public void setHit(int hit) { this.hit = hit; }
    public Date getRegDate() { return regDate; }
    public void setRegDate(Date regDate) { this.regDate = regDate; }
}
```



# SQL 구문 매핑하기 board.model/Board.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE sqlMap
  PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"
  "http://ibatis.apache.org/dtd/sql-map-2.dtd">
<sqlMap namespace="Board">
  <!-- Use type aliases to avoid typing the full classname every time. -->
  <typeAlias alias="Board" type="model.Board"/>
  <resultMap id="BoardResult" class="Board">
    <result property="id" column="id"/>
    <result property="title" column="title"/>
    <result property="writer" column="writer"/>
    <result property="content" column="content"/>
    <result property="hit" column="hit"/>
    <result property="regDate" column="regDate"/>
  </resultMap>
  <!-- Select with no parameters using the result map for Account class. -->
  <select id="listAll" resultMap="BoardResult">
    <![CDATA[ select * from board1 ]]>
  </select>
```

```
<insert id= "insert" parameterClass="Board">
  <selectKey type= "pre" keyProperty="id" resultClass="int">
    select nvl(max(id),0)+1 from board1
  </selectKey>
  insert into board1 values (
    #id#, #title#, #writer#, #content#,0,sysdate )
</insert>
<select id= "select" parameterClass="int" resultClass="Board">
  select * from board1  where id = #id#
</select>
<update id= "hitUpdate" parameterClass="int">
  update board1 set hit = hit + 1 where id = #id#
</update>
<update id= "update" parameterClass="Board">
  update board1 set title=#title#,writer=#writer#,
    content=#content#  where id=#id#
</update>
<delete id= "delete" parameterClass="int">
  delete from board1 where id=#id#
</delete>
</sqlMap>
```

# SQL Map API

## ❖ 조회

queryForObject() 메소드 : `sqlMapper.queryForObject("getTotalRecord")`

데이터베이스로부터 한 개의 레코드를 가져다가 자바 객체에 저장하기 위한 용도로 사용

queryForList() 메소드 : `sqlMapper.queryForList("getListAll");`

데이터베이스로부터 한 개 이상의 레코드를 가져와서 자바 객체의 List를 만드는데 사용

`sqlMapper.insert("insert", bean);`

`sqlMapper.update("hitupdateBeanByIdx", board_idx);`

`sqlMapper.update("updateByIdx", bean);`

`sqlMapper.delete("deleteByIdx", board_idx);`

## config/ SqlMapConfigManager

```
package ibatis;
import java.io.IOException;    import java.io.Reader;
import com.ibatis.common.resources.Resources;
import com.ibatis.sqlmap.client.SqlMapClient;
import com.ibatis.sqlmap.client.SqlMapClientBuilder;
public class SqlMapConfigManager {
    Reader reader;
    private SqlMapClient sqlMapper;
    private SqlMapConfigManager(){
        try {
            reader =
                Resources.getResourceAsReader("ibatis/SqlMapConfig.xml");
            sqlMapper=SqlMapClientBuilder.buildSqlMapClient(reader);
        } catch (IOException e) { System.out.println(e.getMessage()); }
    }
    public static SqlMapConfigManager getInstance() {
        return new SqlMapConfigManager();
    }
    public SqlMapClient getMapper() {
        return sqlMapper;
    }
}
```

# dao/BoardDao.java

```
package dao;
import java.util.List;
import model.Board;
public interface BoardDao {
    List<Board> list();
    int insert(Board board);
    Board select(int id);
    void hitUpdate(int id);
    int update(Board board);
    int delete(int id);
}
```

## dao/BoardDaoImpl

```
package dao;
import ibatis.SqlMapConfigManager;
import java.sql.SQLException;
import java.util.List;
import org.springframework.stereotype.Repository;
import com.ibatis.sqlmap.client.SqlMapClient;
import model.Board;
@Repository
public class BoardDaoImpl implements BoardDao {
    private SqlMapClient getMapper() {
        SqlMapConfigManager manager = SqlMapConfigManager.getInstance();
        SqlMapClient mapper = manager.getMapper();
        return mapper;
    }
    public List<Board> list() {
        List<Board> list = null;
        SqlMapClient mapper = getMapper();
        try {
            list = mapper.queryForList("listAll");
        } catch (SQLException e) { System.out.println("놀구 있네 : "+e.getMessage()); }
        return list;
    }
}
```

```
public int insert(Board board) {  
    int result = 0;  
    SqlMapClient mapper = getMapper();  
    try {  
        mapper.insert("insert",board);  
        result = 1;  
    } catch (SQLException e) { System.out.println("자고 있네! : "+e.getMessage()); }  
    return result;  
}  
  
public Board select(int id) {  
    Board board = null;  
    SqlMapClient mapper = getMapper();  
    try {  
        board = (Board)mapper.queryForObject("select",id);  
    } catch (SQLException e) { System.out.println("놀고 있네! : "+e.getMessage()); }  
    return board;  
}  
  
public void hitUpdate(int id) {  
    SqlMapClient mapper = getMapper();  
    try {  
        mapper.update("hitUpdate",id);  
    } catch (SQLException e) { System.out.println("대박 : "+e.getMessage()); }  
}
```

```
public int update(Board board) {  
    int result = 0;  
    SqlMapClient mapper = getMapper();  
    try {  
        result = mapper.update("update",board);  
    } catch (SQLException e) {  
        System.out.println("쪽박 : "+e.getMessage());  
    }  
    return result;  
}  
  
public int delete(int id) {  
    int result = 0;  
    SqlMapClient mapper = getMapper();  
    try {  
        result = mapper.delete("delete",id);  
    } catch (SQLException e) {  
        System.out.println("쪽박 : "+e.getMessage());  
    }  
    return result;  
}  
}
```



# service/BoardService.java

```
package service;
import java.util.List;
import model.Board;
public interface BoardService {
    List<Board> list();
    int insert(Board board);
    Board select(int id);
    void hitUpdate(int id);
    int update(Board board);
    int delete(int id);
}
```

## service/BoardServiceImpl.java

```
package service;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import dao.BoardDao;
import model.Board;
@Service
public class BoardServiceImpl implements BoardService{
    @Autowired
    private BoardDao bd;
    public List<Board> list() {    return bd.list();    }
    public int insert(Board board) {    return bd.insert(board);    }
    public Board select(int id) {    return bd.select(id);    }
    public void hitUpdate(int id) {    bd.hitUpdate(id);    }
        public int update(Board board) {    return bd.update(board);    }
    public int delete(int id) {    return bd.delete(id);    }
}
```

## controller/BoardController

```
package controller;  
import java.util.List;  
import model.Board;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestMethod;
```

```
import service.BoardService;
```

```
@Controller
```

```
public class BoardController {  
    @Autowired  
    private BoardService bs;  
    @RequestMapping(value="list")  
    public String list(Model model) {  
        List<Board> list = bs.list();  
        model.addAttribute("list",list);  
        return "list";  
    }  
    @RequestMapping(value="writeForm")  
    public String writeForm() {  
        return "writeForm";  
    }  
}
```

```
@RequestMapping(value="write",method=RequestMethod.POST)  
public String write(Board board, Model model) {  
    int result = bs.insert(board);  
    if (result > 0) {return "redirect:list.do";  
    } else {return "writeForm";}  
}  
  
@RequestMapping(value="detail")  
public String detail(int id, Model model) {  
    Board board = bs.select(id);  
    bs.hitUpdate(id);  
    model.addAttribute("board", board);  
    return "detail";  
}  
  
@RequestMapping(value="updateForm")  
public String updateForm(int id, Model model) {  
    Board board = bs.select(id);  
    model.addAttribute("board", board);  
    return "updateForm";  
}  
  
@RequestMapping(value="update")  
public String update(Board board, Model model) {  
    int result = bs.update(board);  
    if (result > 0) {return "redirect:list.do";  
    } else { return "redirect:updateForm.do?id="+board.getId(); }  
}
```

```
@RequestMapping(value="delete")
public String delete(int id, Model model) {
    int result = bs.delete(id);
    if (result > 0) {return "redirect:list.do";
    } else {
        model.addAttribute("msg", "게임하니까 안되지");
        return "forward:detail.do?id="+id;
    }
}
}
```

# WEB-INF/servlet-context.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/mvc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:beans="http://www.springframework.org/schema/beans"
xmlns:context="http://www.springframework.org/schema/context"
xsi:schemaLocation="http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc.xsd
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.xsd">
<!-- DispatcherServlet Context: defines this servlet's request-processing infrastructure -->
<!-- Enables the Spring MVC @Controller programming model -->
<annotation-driven />
<!-- Handles HTTP GET requests for /resources/** by efficiently serving up static resources in
the ${webappRoot}/resources directory -->
<resources mapping="/resources/**" location="/resources/" />
<!-- Resolves views selected for rendering by @Controllers to .jsp resources in the /WEB-
INF/views directory -->
<beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
<beans:property name="prefix" value="/WEB-INF/views/" />
<beans:property name="suffix" value=".jsp" />
</beans:bean>
<context:component-scan base-package="dao,service,controller" />
</beans:beans>
```

# header.jsp

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  
<link href="css/board.css" rel="stylesheet" type="text/css">
```

## views/list.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ include file="header.jsp" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html> <head> <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title> </head>
<body>
    <h2>게시판</h2>
    <a href="writeForm.do">글쓰기</a>
    <table>
        <tr> <th>번호</th> <th>제목</th> <th>작성자</th>
        <th>조회수</th> <th>작성일</th> </tr>
        <c:forEach items="${list}" var="board">
            <tr> <td>${board.id}</td>
                <td><a href="detail.do?id=${board.id}">${board.title}</a></td>
                <td>${board.writer}</td> <td>${board.hit}</td>
                <td><fmt:formatDate value="${board.regDate}"
                    pattern="yyyy/MM/dd"/></td> </tr>
        </c:forEach>
    </table>
</body>
</html>
```



## board/detail.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ include file="header.jsp" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html> <head> <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title> </head> <body> <h2>상세 조회</h2>
    <c:if test="${msg!=null}">${msg}</c:if>
    <table>
        <tr> <th>제목</th> <td>${board.title}</td> </tr>
        <tr> <th>작성자</th> <td>${board.writer}</td> </tr>
        <tr> <th>내용</th> <td><pre>${board.content}</pre> </td> </tr>
        <tr> <th>조회수</th> <td>${board.hit}</td> </tr>
        <tr> <th>작성일</th> <td>
            <fmt:formatDate value="${board.regDate}" pattern="yyyy/MM/dd"/> </td> </tr>
    </table>
    <input type="button" value="수정"
        onclick="location.href='updateForm.do?id=${board.id}'">
    <input type="button" value="삭제"
        onclick="location.href='delete.do?id=${board.id}'">
    <input type="button" value="목록" onclick="location.href='list.do'">
</body>
</html>
```

## views/update.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ include file="header.jsp" %>
<!DOCTYPE html>
<html> <head> <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title> </head> <body>
    <h2>게시글 수정</h2>
    <form action="update.do" method="post">
        <input type="hidden" name="id" value="${board.id}">
        <table>
            <tr> <th>제목</th> <td> <input type="text" name="title"
                required="required" value="${board.title}"> </td> </tr>
            <tr> <th>작성자</th> <td> <input type="text" name="writer"
                required="required" value="${board.writer}"> </td> </tr>
            <tr> <th>내용</th> <td> <textarea rows="10" cols="50"
                name="content" required="required"> ${board.content} </textarea> </td> </tr>
            <tr> <td colspan="2"> <input type="submit" value="확인">
                <input type="reset" value="취소"> </td> </tr>
        </table>
    </form>
</body>
</html>
```

## views/write.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ include file="header.jsp" %>
<!DOCTYPE html>
<html> <head> <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title> </head> <body>
    <h2>게시판 작성</h2>
    <form action="write.do" method="post">
        <table>
            <tr> <th>제목</th> <td> <input type="text" name="title"
                required="required"> </td> </tr>
            <tr> <th>작성자</th> <td> <input type="text" name="writer"
                required="required"> </td> </tr>
            <tr> <th>내용</th> <td> <textarea rows="10" cols="50"
                name="content" required="required"> </textarea> </td> </tr>
            <tr> <td colspan="2"> <input type="submit" value="확인">
                <input type="reset" value="취소"> </td> </tr>
        </table>
    </form>
</body>
</html>
```

# 문제

게시판에 글쓰기 좌측에 아래와 같이 제목으로 조회하는 내용을 추가하여 프로젝트를 수정하시오

## 게시판 리스트

번호	제목	내용
1	<a href="#">참</a>	알수없네
2	<a href="#">이건 뭐</a>	왜
3	<a href="#">헐</a>	왜되지
4	<a href="#">한글</a>	되나
5	<a href="#">대박</a>	헐
6	<a href="#">대박</a>	대박사건
7	<a href="#">3</a>	3
8	<a href="#">1</a>	1

새글쓰기

제목 :

검색

# 간단한 게시판 작성(board)

## 1. 테이블 생성

테이블명 : board2

Name	Null?	Type
-----		
SEQ	NOT NULL	NUMBER(11)
SUBJECT		VARCHAR2(128)
CONTENT		VARCHAR2(512)

## 2. 시퀀스

```
create sequence board2_seq  
  start with 1  
  increment by 1  
  nocache  
  nocycle;
```

# pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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
    http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.test</groupId>
  <artifactId>board</artifactId>
  <name>abc</name>
  <packaging>war</packaging>
  <version>1.0.0-BUILD-SNAPSHOT</version>
  <properties>
    <java-version>1.8</java-version>
    <org.springframework-version>4.1.1.RELEASE</org.springframework-version>
    <org.aspectj-version>1.6.9</org.aspectj-version>
    <org.slf4j-version>1.5.10</org.slf4j-version>
  </properties>
  <repositories>
    <repository> // 오라클 사용할때
      <id>codeIds</id>
      <url>https://code.lds.org/nexus/content/groups/main-repo</url>
    </repository>
  </repositories>
```

```
<dependencies>
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-orm</artifactId>
<version>${org.springframework-version}</version>
</dependency>
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-test</artifactId>
<version>${org.springframework-version}</version>
</dependency>
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-jdbc</artifactId>
<version>${org.springframework-version}</version>
</dependency>
<dependency>
<groupId>org.hibernate</groupId>
<artifactId>hibernate-validator</artifactId>
<version>5.1.3.Final</version>
</dependency>
```

```
<!-- oracle -->
<dependency>
  <groupId>com.oracle</groupId>    <artifactId>ojdbc6</artifactId>
  <version>11.2.0.3</version>
  <scope>compile</scope>
</dependency>
<!-- dbcp jdbc connection pool -->
<dependency>
  <groupId>commons-dbcp</groupId>
  <artifactId>commons-dbcp</artifactId> <version>1.2.2</version>
</dependency>
<dependency>
  <groupId>c3p0</groupId> <artifactId>c3p0</artifactId>
  <version>0.9.1.2</version>
</dependency>
<!-- mybatis -->
<dependency>
  <groupId>org.mybatis</groupId> <artifactId>mybatis</artifactId>
  <version>3.2.3</version>
</dependency>
<dependency>
  <groupId>org.mybatis</groupId> <artifactId>mybatis-spring</artifactId>
  <version>1.2.1</version>
</dependency>
```



```
<!-- ibatis -->
<dependency>
<groupId>org.apache.ibatis</groupId>
<artifactId>ibatis-sqlmap</artifactId>
<version>2.3.4.726</version>
</dependency>
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-ibatis</artifactId>
<version>2.0.8</version>
</dependency>
<!-- cglib ibatis를 위한 프록시 라이브러리 -->
<dependency>
<groupId>cglib</groupId>
<artifactId>cglib</artifactId>
<version>2.2</version>
</dependency>
```

```
<!-- Spring -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-context</artifactId>
<version>${org.springframework-version}</version>
<exclusions>
<!-- Exclude Commons Logging in favor of SLF4j -->
<exclusion>
<groupId>commons-logging</groupId>
<artifactId>commons-logging</artifactId>
</exclusion>
</exclusions>
</dependency>
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-webmvc</artifactId>
<version>${org.springframework-version}</version>
</dependency>
<!-- AspectJ -->
<dependency>
<groupId>org.aspectj</groupId>
<artifactId>aspectjrt</artifactId>
<version>${org.aspectj-version}</version>
</dependency>
```

```
<dependency>
<groupId>org.aspectj</groupId>
<artifactId>aspectjweaver</artifactId>
<version>1.6.8</version>
</dependency>
<!-- Logging -->
<dependency>
<groupId>org.slf4j</groupId>
<artifactId>slf4j-api</artifactId>
<version>${org.slf4j-version}</version>
</dependency>
<dependency>
<groupId>org.slf4j</groupId>
<artifactId>jcl-over-slf4j</artifactId>
<version>${org.slf4j-version}</version>
<scope>runtime</scope>
</dependency>
<dependency>
<groupId>org.slf4j</groupId>
<artifactId>slf4j-log4j12</artifactId>
<version>${org.slf4j-version}</version>
<scope>runtime</scope>
</dependency>
```

```
<dependency>
<groupId>log4j</groupId>
<artifactId>log4j</artifactId>
<version>1.2.15</version>
<exclusions>
<exclusion>
<groupId>javax.mail</groupId>
<artifactId>mail</artifactId>
</exclusion>
<exclusion>
<groupId>javax.jms</groupId>
<artifactId>jms</artifactId>
</exclusion>
<exclusion>
<groupId>com.sun.jdmk</groupId>
<artifactId>jmxtools</artifactId>
</exclusion>
<exclusion>
<groupId>com.sun.jmx</groupId>
<artifactId>jmxri</artifactId>
</exclusion>
</exclusions>
<scope>runtime</scope>
</dependency>
```

```
<!-- @Inject -->
<dependency>
<groupId>javax.inject</groupId>
<artifactId>javax.inject</artifactId>
<version>1</version>
</dependency>
<!-- Servlet -->
<dependency>
<groupId>javax.servlet</groupId>
<artifactId>servlet-api</artifactId>
<version>2.5</version>
<scope>provided</scope>
</dependency>
<dependency>
<groupId>javax.servlet.jsp</groupId>
<artifactId>jsp-api</artifactId>
<version>2.1</version>
<scope>provided</scope>
</dependency>
<dependency>
<groupId>javax.servlet</groupId>
<artifactId>jstl</artifactId>
<version>1.2</version>
</dependency>
```

```
<dependency>
<groupId>org.apache.tiles</groupId>
<artifactId>tiles-jsp</artifactId>
<version>2.2.2</version>
</dependency>
<!-- Test -->
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>4.7</version>
<scope>test</scope>
</dependency>
<dependency>
<groupId>commons-fileupload</groupId>
<artifactId>commons-fileupload</artifactId>
<version>1.2</version>
</dependency>
```

```
<!-- Spring Security -->
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-core</artifactId>
    <version>3.2.4.RELEASE</version>
</dependency>
```

```
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-web</artifactId>
  <version>3.2.4.RELEASE</version>
</dependency>
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-config</artifactId>
  <version>3.2.4.RELEASE</version>
</dependency>
<dependency>
<groupId>org.springframework.security</groupId>
<artifactId>spring-security-taglibs</artifactId>
<version>3.1.4.RELEASE</version>
</dependency>
<dependency>
<groupId>commons-io</groupId>
<artifactId>commons-io</artifactId>
<version>1.3.2</version>
</dependency>
<dependency>
<groupId>org.scala-lang</groupId>
<artifactId>scala-library</artifactId>
<version>2.10.4</version>
</dependency>
</dependencies>
```

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <configuration>
        <source>${java-version}</source> <target>${java-version}</target>
      </configuration>
    </plugin>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-war-plugin</artifactId>
      <configuration> <warName>abc</warName> </configuration>
    </plugin>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-dependency-plugin</artifactId>
      <executions>
        <execution>
          <id>install</id> <phase>install</phase>
          <goals> <goal>sources</goal> </goals>
        </execution>
      </executions>
    </plugin>
    <plugin>
```



```
<groupId>org.apache.maven.plugins</groupId>  
<artifactId>maven-resources-plugin</artifactId>  
<version>2.5</version>  
<configuration>  
<encoding>UTF-8</encoding>  
</configuration>  
</plugin>  
</plugins>  
</build>  
</project>
```

# 웹 프로그램

## webapp/index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Insert title here</title>
    </head>
    <body>
        <%
            response.sendRedirect("list.do");
        %>
    </body>
</html>
```

## SQL Map XML 파일 ( <http://www.ibatis.com/dtd/sql-mapconfig-2.dtd>)

위 예제에서 우리는 SQL Maps의 가장 간단한 형태를 보았다. SQL Map문서 구조내에 사용가능한 다른 옵션이 있다. 좀더 많은 기능을 가지는 mapped statement의 예제이다.

```
<sqlMap namespace="Product">
<typeAlias alias="product" type="com.ibatis.example.Product" />
<parameterMap id="productParam" class="product">

<parameter property="id"/>
</parameterMap>

<resultMap id="productResult" class="product">
  <result property="id" column="PRD_ID"/>
  <result property="description" column="PRD_DESCRIPTION"/>
</resultMap>

<select id="getProduct" parameterMap="productParam"
  resultMap="productResult" cacheModel="product-cache">
  select * from PRODUCT where PRD_ID = ?
</select>
</sqlMap>
```

Statement Element	Attributes	Child Elements	Methods
<statement>	id parameterClass resultClass parameterMap resultMap cacheModel xmlResultName	All dynamic elements	insert update delete All query methods
<insert>	id parameterClass parameterMap	All dynamic elements <selectKey>	insert update delete
<update>	id parameterClass parameterMap	All dynamic elements	insert update delete
<delete>	id parameterClass parameterMap	All dynamic elements	insert update delete
<select>	id parameterClass resultClass parameterMap resultMap cacheModel	All dynamic elements	All query methods
<procedure>	id parameterClass resultClass parameterMap resultMap xmlResultName	All dynamic elements	insert update delete All query methods

# The SQL

1. SQL 은 map의 가장 중요한 부분을 차지한다. 데이터베이스와 JDBC드라이버에 적합한 어떤 SQL이 될수 있다.
2. 가능한 어떤 기능을 사용할수 있고, 드라이버가 지원하는 한 다중 statement에 전달할수도 있다.
3. 문서에서 SQL과 XML을 혼합하기 때문에 특수문자의 충돌이 잠재적으로 존재한다. 대부분의 공통적인 것은 greater-than 과 less-than 문자들이다(<>). 이것들은 SQL문에서 공통적으로 요구되고 XML에서는 예약어이다.
4. 특수 XML문자를 처리하기 위한 간단한 해결법이 있다. 표준적인 XML CDATA 섹션을 사용함으로써 특수문자의 어떤것도 파싱되지 않고 문제는 해결된다.

```
<statement id="getPersonsByAge" parameterClass="int"
resultClass="examples.domain.Person">
<![CDATA[
SELECT *
FROM PERSON
WHERE AGE > #value#
]]>
</statement>
```

# 자동 생성 키

1. 많은 관계형 데이터베이스 시스템은 기본키(primary key) 필드의 자동생성을 지원한다. 이 RDBMS의 기능은 종종 특정업체에 종속된다. SQL Map은 <insert> 요소의 <selectKey>를 통해 자동생성키를 지원한다. 선행생성키(pre-generated - 이를 테면 오라클)와 후생성키(post-generated - 이를 테면 MS-SQL 서버) 모두 지원한다. 여기에 그 예제가 있다.

**<!--Oracle SEQUENCE Example -->**

```
<insert id="insertProduct-ORACLE" parameterClass="com.domain.Product">  
  <selectKey resultClass="int" keyProperty="id" >  
    SELECT STOCKIDSEQUENCE.NEXTVAL AS ID FROM DUAL  
  </selectKey>  
  insert into PRODUCT (PRD_ID,PRD_DESCRIPTION)  
    values (#id#,#description#)  
</insert>
```

**<!-- Microsoft SQL Server IDENTITY Column Example -->**

```
<insert id="insertProduct-MS-SQL" parameterClass="com.domain.Product">  
insert into PRODUCT (PRD_DESCRIPTION) values (#description#)  
<selectKey resultClass="int" keyProperty="id" >  
SELECT @@IDENTITY AS ID  
</selectKey>  
</insert>
```

# 저장 프로시저

1. 저장 프로시저는 <procedure> statement요소를 통해 지원된다. 저장 프로시저를 출력물 파라미터와 함께 어떻게 사용하는지 다음 예제에서 보여준다.

```
<parameterMap id="swapParameters" class="map" >
    <parameter property="email1" jdbcType="VARCHAR"
        javaType="java.lang.String" mode="INOUT"/>
    <parameter property="email2" jdbcType="VARCHAR"
        javaType="java.lang.String" mode="INOUT"/>
</parameterMap>

<procedure id="swapEmailAddresses" parameterMap="swapParameters" >
    {call swap_email_address (?, ?)}
</procedure>
```

위처럼 프로시저를 호출하는 것은 파라미터 객체(map)내에서 두개의 칼럼사이에 두개의 이메일주소를 교체하는것이다. 파라미터 객체는 파라미터 맵핑의 mode속성값이 "INOUT"또는 "OUT"일 경우에만 변경된다. 다른 경우라면 변경되지 않고 남는다. 명백한 불변의 파라미터 객체(이를 테면 String)는 변경할수 없다.

# parameterClass

1. parameterClass 속성값은 자바클래스의 전체경로를 포함(예를 들면 패키지를 포함한)한 이름이다. parameterClass 속성은 옵션이지만 사용이 굉장히 추천되는 것이다. 2. 이것은 프레임워크 성능을 향상시키는 만큼 statement에 전달하는 파라미터를 제한하는데 사용된다. 만약 당신이 parameterMap을 사용한다면 parameterClass속성을 사용할 필요가 없다.
2. "examples.domain.Product" 타입의 객체를 허락하길 원한다면 당신은 다음처럼 할 수 있을 것이다.

```
<statement id="statementName"  
    parameterClass="examples.domain.Product">  
    insert into PRODUCT values (#id#, #description#, #price#)  
</statement>
```

1. 비록 이전버전과의 호환성을 위한 옵션이지만 이것은 언제나 파라미터 클래스를 제공하는 것은 매우 추천되는 사항이다(물론 요구되는 파라미터가 없더라도). 프레임워크가 먼저 타입을 안다면 스스로 최적화능력을 가지기 때문에 클래스를 제공함으로써 좀더 나은 성능을 달성할 수 있다.
2. 명시된 parameterClass 없이 선호하는 속성(get/set메소드)을 가지는 자바빈즈는 파라미터를 받을 것이고 어느 위치에서 매우 유용하다



# parameterMap

1. parameterMap 속성값은 명시된(밑의 경우처럼) parameterMap요소의 이름이다.
2. parameterMap속성은 parameterClass 속성과 인라인 파라미터의 이익이 되도록 사용된다.
3. parameterMap의 생각은 JDBC PreparedStatement의 값 토큰과 매치되는 정렬된 파라미터 목록을 명시한다.

예를들면:

```
<parameterMap id="insert-product-param" class="com.domain.Product">
    <parameter property="id"/>
    <parameter property="description"/>
</parameterMap>
<statement id="insertProduct" parameterMap="insert-product-param">
    insert into PRODUCT (PRD_ID, PRD_DESCRIPTION) values (?,?);
</statement>
```

위의 예제에서, 파라미터 map은 SQL문에서 값토큰("?")에 매치되고 정렬되는 두개의 파라미터를 서술한다. 그래서 첫번째 "?"는 "id" 속성값에 대체되고 두번째는 "description" 속성값에 대체된다.

# 인라인 파라미터

1. 인라인 파라미터는 맵핑된 statement내부에서 사용될수 있다.

예를 들면 :

```
<statement id="insertProduct" >  
    insert into PRODUCT (PRD_ID, PRD_DESCRIPTION)  
        values (#id#, #description#);  
</statement>
```

위 예제에서 인라인 파라미터는 #id# 와 #description# 이다. 각각은 statement파라미터를 대체하는 자바빈즈 속성을 표현한다.

Product클래스는 포함된 프라퍼티 토큰이 위치해 있는 statement내에 위치하는 값을 위해 읽게 되는 id 와 description 프라퍼티를 가진다. id=5 와 description="dog" 를 가지는 Product를 넘겨받은 statement는 다음처럼 수행된다.

```
insert into PRODUCT (PRD_ID, PRD_DESCRIPTION)  
values (5, 'dog');
```

# resultClass

resultClass속성값은 자바클래스의 전체경로를 포함(예를 들면 패키지를 포함한)한 이름이다. resultClass속성은 우리에게 ResultSetMetaData에 기반한 JDBC ResultSet에 자동매핑되는 클래스를 명시하도록 한다.

자바빈즈의 프라퍼티와 ResultSet의 칼럼이 매치될때마다 프라퍼티는 칼럼값과 함께 생성된다. 이것은 매우 짧고 달콤하게 매핑된 statement를 쿼리한다.

```
<statement id="getPerson" parameterClass="int" resultClass="examples.domain.Person">
SELECT PER_ID as id, PER_FIRST_NAME as firstName, PER_LAST_NAME as lastName,
PER_BIRTH_DATE as birthDate, PER_WEIGHT_KG as weightInKilograms,
PER_HEIGHT_M as heightInMeters FROM PERSON WHERE PER_ID = #value#
</statement>
```

위의 예제에서 Person클래스는 id, firstName, lastName, birthDate, weightInKilograms, heightInMeters를 포함하는 프라퍼티를 가진다. 칼럼별칭과 함께 대응되는 각각은 SQL select문에 의해 서술된다. 칼럼별칭은 데이터베이스 칼럼이름이 매치되지 않을때만 요구된다. 일반적으로는 요구되지 않는다.

실행되었을때 Person객체는 프라퍼티이름과 칼럼명에 기반해서 초기화되기 위해 매핑되는 result set으로부터 초기화되고 결과를 반환한다.

resultClass으로 자동매핑하는데는 몇 가지 제한점이 있다. 출력칼럼의 타입을 명시하는 방법은 없다. 관련된 데이터를 자동적으로 로드하는방법이 없고 ResultSetMetaData에 접근하는데 필요한 접근법내에서 하찮은 성능결과가 있다. 이 제한점 모드는 명시적인 resultMap를 사용함으로써 극복할수 있다.

# resultMap

1. resultMap프라퍼티는 좀더 공통적으로 사용되고 이해하기 위해 가장 중요한 속성중에 하나이다. 이 resultMap속성값은 명시된 resultMap요소의 이름이다.
2. resultMap속성을 사용하는 것은 result set으로부터 데이터와 칼럼에 매핑되는 프라퍼티를 어떻게 꺼내는지 제어하도록 한다. resultClass속성을 사용하는 자동매핑 접근법과는 달리 resultMap는 칼럼타입을 명시하고 null값을 대체 그리고 복합 프라퍼티매핑(다른 자바빈즈, Collections 그리고 원시타입래퍼)을 허락한다.

```
<resultMap id="get-product-result" class="com.ibatis.example.Product">
  <result property="id" column="PRD_ID"/>
  <result property="description" column="PRD_DESCRIPTION"/>
</resultMap>
<statement id="getProduct" resultMap="get-product-result">
select * from PRODUCT
</statement>
```

위 예제에서 SQL쿼리로부터 ResultSet은 resultMap정의를 사용해서 Product인스턴스에 매핑할것이다. resultMap은 "id"프라퍼티가 "PRD\_ID"칼럼과 "PRD\_DESCRIPTION"칼럼에 의해 생성되는 "description"프라퍼티에 의해 생성될것이다. "select \*" 사용하는 것은 지원된다는 것에 주의하라. ResultSet내 반환칼럼 모두에 매핑할 필요는 없다.

# Board.xml : 위치 resources ibatis

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE sqlMap
  PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"
  "http://ibatis.apache.org/dtd/sql-map-2.dtd">
<sqlMap namespace="Board">
  <!-- Use type aliases to avoid typing the full classname every time. -->
  <typeAlias alias="Board" type="model.Board"/>
  <resultMap id="BoardResult" class="Board">
    <result property="id" column="id"/>
    <result property="title" column="title"/>
    <result property="writer" column="writer"/>
    <result property="content" column="content"/>
    <result property="hit" column="hit"/>
    <result property="regDate" column="regDate"/>
  </resultMap>
  <!-- Select with no parameters using the result map for Account class. -->
  <select id="listAll" resultMap="BoardResult">
    <![CDATA[ select * from board1 order by id desc ]]>
  </select>
  <select id="listSearch" parameterClass="java.lang.String"
    resultMap="BoardResult">
    <![CDATA[ select * from board1
    where title like '%'||#search#||'%' order by id desc ]]>
  </select>
```

```
<insert id= "insert" parameterClass="Board">
  <selectKey type= "pre" keyProperty="id" resultClass="int">
    select nvl(max(id),0)+1 from board1
  </selectKey>
  insert into board1 values ( #id#, #title#, #writer#, #content#,0,sysdate )
</insert>
  <select id= "select" parameterClass="int" resultClass="Board">
    select * from board1 where id = #id#
  </select>
  <select id= "totalCnt" resultClass="int">
    select nvl(count(*),0) from board1
  </select>
  <select id= "searchCnt" resultClass="int">
    select nvl(count(*),0) from board1 where title like '%'||#search#||'%'
  </select>
  <update id= "hitUpdate" parameterClass="int">
    update board1 set hit = hit + 1 where id = #id#
  </update>
  <update id= "update" parameterClass="Board">
    update board1 set title=#title#,writer=#writer#,
      content=#content# where id=#id#
  </update>
  <delete id= "delete" parameterClass="int">
    delete from board1 where id=#id#
  </delete>
</sqlMap>
```

## SQL Map XML 설정파일

(<http://www.ibatis.com/dtd/sql-map-config-2.dtd>)

SQL Maps는 데이터소스에 대한 설정, 쓰레드 관리와 같은 SQL Maps와 다른 옵션에 대한 설정을 제공하는 중앙집중적인 XML 설정 파일을 사용해서 설정된다.

### **SqlMapConfig.xml**

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<!DOCTYPE sqlMapConfig PUBLIC "-//iBATIS.com//DTD SQL Map Config  
2.0//EN" "http://www.ibatis.com/dtd/sql-map-config-2.dtd">
```

```
<!--Always ensure to use the correct XML header as above! -->
```

```
<sqlMapConfig>
```

```
<!--The properties (name=value) in the file specified here can be used placeholders in  
this config file (e.g. "${driver}". The file is relative to the classpath and is completely  
optional. -->
```

```
<properties resource="examples/sqlmap/maps/SqlMapConfigExample.properties" />
```

```
<!--These settings control SqlMapClient configuration details, primarily to do with  
transaction management. They are all optional (more detail later in this document). -->
```

```
<settings
```

```
cacheModelsEnabled="true"
```

```
enhancementEnabled="true"
```

```
lazyLoadingEnabled="true"
```

```
maxRequests="32"
```

```
maxSessions="10"
```

```
maxTransactions="5"
```

```
useStatementNamespaces="false"
```

```
/>
```

```
<!--Type aliases allow you to use a shorter name for long fully qualified class names. -->
<typeAlias alias="order" type="testdomain.Order"/>
<!--Configure a datasource to use with this SQL Map using SimpleDataSource.
Notice the use of the properties from the above resource -->
<transactionManager type="JDBC" >
<dataSource type="SIMPLE">
<property name="JDBC.Driver" value="${driver}"/>
<property name="JDBC.ConnectionURL" value="${url}"/>
<property name="JDBC.Username" value="${username}"/>
<property name="JDBC.Password" value="${password}"/>
<property name="JDBC.DefaultAutoCommit" value="true" />
<property name="Pool.MaximumActiveConnections" value="10"/>
<property name="Pool.MaximumIdleConnections" value="5"/>
<property name="Pool.MaximumCheckoutTime" value="120000"/>
<property name="Pool.TimeToWait" value="500"/>
<property name="Pool.PingQuery" value="select 1 from ACCOUNT"/>
<property name="Pool.PingEnabled" value="false"/>
<property name="Pool.PingConnectionsOlderThan" value="1"/>
<property name="Pool.PingConnectionsNotUsedFor" value="1"/>
</dataSource>
</transactionManager>
<!--Identify all SQL Map XML files to be loaded by this SQL map. Notice the paths
are relative to the classpath. For now, we only have one... -->
<sqlMap resource="examples/sqlmap/maps/Person.xml" />
</sqlMapConfig>
```



## <properties> 요소

1. SQL Maps은 SQL Maps XML설정파일과 함께 속하는 표준적인 자바 속성파일 (name=value)을 지정하는 하나의 <properties>요소를 가질 수 있다. 그렇게 함으로써 속성파일내에 각각의 이름지어진 값들은 SQL Maps설정파일내에 참조될수 있는 변수가될수 있고 모든 SQL Maps는 내부에서 참조된다.
2. 속성파일이 다음을 포함한다면 **driver**=org.hsqldb.jdbcDriver  
SQL Maps설정파일또는 설정문서에 의해 참조되는 각각의 SQL Maps는 \${driver} 형태로 사용가능하고 org.hsqldb.jdbcDriver 라는 값이 참조된다. 예를 들면  
<property name="JDBC.Driver" value="**\${driver}**" />
3. 이것은 빌드되거나 테스트 그리고 배치되는 동안 편리하게 된다. 이것은 다중 환경이나 설정파일을 위해 자동화툴을 사용하는 애플리케이션을 쉽게 인식하도록 한다.
4. 속성은 클래스패스나 어떤 유효한 URL로부터 로드될수 있다. 예를 들면 고정된 파일경로를 위해 다음처럼 사용한다.  
<properties url="file:///c:/config/my.properties" />

## <settings> 요소

1. `maxRequests` 한꺼번에 SQL문을 수행할 수 있는 스레드의 수이다. 이것은 최소한 10개의 `maxTransactions`이고 언제나 `maxSessions`과 `maxTransactions`보다 크다. 종종 동시요청값의 최대치를 줄이면 성능향상을 보여준다.

`maxRequests="256"` Default: 512

2. `maxSessions` 주어진 시간동안 활성화될 수 있는 세션의 수이다. 이것은 언제나 `maxTransaction`보다 같거나 커야 하고 `maxRequests`보다 작아야 한다. 동시세션값의 최대치를 줄이면 전체적인 메모리사용량을 줄일 수 있다.

`maxSessions="64"` Default: 128

3. `maxTransactions` 이것은 한꺼번에 `SqlMapClient.startTransaction()`에 들어갈 수 있는 스레드의 최대갯수이다.

`maxTransactions="16"` Default: 32

4. `cacheModelsEnabled` 이 셋팅은 `SqlMapClient`를 위한 모든 캐쉬모델을 가능하게 하거나 가능하지 않게 한다. 이것은 디버깅시 도움이 된다.

`cacheModelsEnabled="true"` Default: true (enabled)

5. `lazyLoadingEnabled` 이 셋팅은 `SqlMapClient`를 위한 모든 늦은(lazy)로딩을 가능하게 하거나 가능하지 않게 한다. 이것은 디버깅시 도움이 된다.

`lazyLoadingEnabled="true"` Default: true (enabled)

6. `enhancementEnabled` 이 셋팅은 향상된 늦은(lazy)로딩처럼 최적화된 자바빈즈 속성 접근을 위해 런타임시 바이트코드 향상을 가능하게 한다.

`enhancementEnabled="true"` Default: false (disabled)

## <typeAlias> 요소

typeAlias 요소는 길고 전체적인 클래스명을 참조하기 위한 짧은 이름을 명시하도록 한다.

`<typeAlias alias="shortname" type="com.long.class.path.Class"/>`

SQL Maps설정 파일에서 사용되는 미리 정의된 몇몇 alias가 있다. 그것들은

Transaction Manager Aliases : JDBC, JTA, EXTERNAL

Data Source Factory Aliases : SIMPLE, DBCP

## <transactionManager> 요소

1.0 변환노트: SQL Maps 1.0은 다중의 데이터소스 설정을 허락했다. 그러므로 2.0에서는 오직 하나의 데이터소스만을 허락한다.

<transactionManager> 요소는 당신이 SQL Maps를 위한 트랜잭션 관리를 설정하도록 한다.

type 속성값은 사용하기 위한 트랜잭션관리자는 표시한다. 그 값은 클래스명이거나 타입 alias일 수 있다. 3개의 트랜잭션 관리자는 JDBC, JTA 그리고 EXTERNAL 중에 하나로 표시할수 있다.

1) JDBC – Connection commit()과 rollback()메소드를 통해 트랜잭션을 제어하기 위한 JDBC를 사용하게 된다.

2) JTA – 이 트랜잭션관리자는 SQL Maps가 다른 데이터베이스나 트랜잭션 자원을 포함하는 더욱더 넓은 범위의 트랜잭션을 포함하도록 하는 JTA전역트랜잭션을 사용한다. 이 설정은 JNDI자원으로부터 사용자 트랜잭션을 위치시키기 위한 UserTransaction 속성값을 요구한다.

3) EXTERNAL – 이것은 당신 자신이 트랜잭션을 관리하도록 한다. 당신은 여전히 데이터소스를 설정할수 있지만 프레임워크 생명주기의 부분처럼 트랜잭션이 커밋되거나 롤백되지 않는다. 이것은 당신 애플리케이션의 부분이 외부적으로 SQL Maps 트랜잭션을 관리해야 한다는것이다. 이 셋팅은 비-트랜잭션(예를 들면 읽기전용) 데이터베이스에 유용하다.

## <dataSource> 요소

트랜잭션관리자 설정의 포함된 부분은 dataSource 요소이고 SQL Maps를 사용하기 위한 데이터소스를 설정하기 위한 속성값의 집합이다.

### **SimpleDataSourceFactory**

SimpleDataSource 는 데이터소스를 제공하는 컨테이너가 없는 경우에 connection을 제공하기 위해 기본적으로 풀링(pooling) 데이터소스 구현을 제공한다. 이것은 iBATIS SimpleDataSource connection풀링을 기초로 한다.

```
<transactionManager type="JDBC">
<dataSource type="SIMPLE">
<property name="JDBC.Driver" value="org.postgresql.Driver"/>
<property name="JDBC.ConnectionURL"
value="jdbc:postgresql://server:5432/dbname"/>
<property name="JDBC.Username" value="user"/>
<property name="JDBC.Password" value="password"/>
<!--OPTIONAL PROPERTIES BELOW -->
<property name="Pool.MaximumActiveConnections" value="10"/>
<property name="Pool.MaximumIdleConnections" value="5"/>
<property name="Pool.MaximumCheckoutTime" value="120000"/>
<property name="Pool.TimeToWait" value="10000"/>
<property name="Pool.PingQuery" value="select * from dual"/>
<property name="Pool.PingEnabled" value="false"/>
<property name="Pool.PingConnectionsOlderThan" value="0"/>
<property name="Pool.PingConnectionsNotUsedFor" value="0"/>
</dataSource>
</transactionManager>
```

# SqlMapConfig.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE sqlMapConfig
    PUBLIC "-//ibatis.apache.org//DTD SQL Map Config 2.0//EN"
    "http://ibatis.apache.org/dtd/sql-map-config-2.dtd">
<sqlMapConfig>
    <sqlMap resource="ibatis/Board.xml"/>
</sqlMapConfig>
```

# dao-context.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:mvc="http://www.springframework.org/schema/mvc"
xmlns:tx="http://www.springframework.org/schema/tx"
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-3.1.xsd
http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc-3.1.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.1.xsd">
```

```
<!-- //Oracle 접속부분 -->
<context:property-placeholder
location= "classpath:ibatis/jdbc.properties"/>
<bean id= "dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource"
destroy-method= "close">
<property name= "driverClass" value= "${jdbc.driverClassName}" />
<property name= "jdbcUrl" value= "${jdbc.url}" />
<property name= "user" value= "${jdbc.username}" />
<property name= "password" value= "${jdbc.password}" />
<property name= "maxPoolSize" value= "${jdbc.maxPoolSize}" />
</bean>
<!-- # iBatis setting# -->
<bean id= "sqlMapClient"
class= "org.springframework.orm.ibatis.SqlMapClientFactoryBean">
<property name= "configLocation"
value= "classpath:ibatis/SqlMapConfig.xml" />
<property name= "dataSource" ref= "dataSource" />
</bean>
<bean id= "template"
class= "org.springframework.orm.ibatis.SqlMapClientTemplate">
<property name= "sqlMapClient" ref= "sqlMapClient" />
</bean>
</beans>
```

# log4j.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE log4j:configuration PUBLIC "-//APACHE//DTD LOG4J 1.2//EN" "log4j.dtd">
<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/">
  <!-- Appenders -->
  <appender name="console" class="org.apache.log4j.ConsoleAppender">
    <param name="Target" value="System.out" />
    <layout class="org.apache.log4j.PatternLayout">
      <param name="ConversionPattern" value="%-5p: %c - %m%n" />
    </layout>
  </appender>

  <!-- Application Loggers -->
  <logger name="com.test.board"><level value="info" /></logger>

  <!-- 3rdparty Loggers -->
  <logger name="org.springframework.core"><level value="info" /></logger>
  <logger name="org.springframework.beans"><level value="info" /></logger>
  <logger name="org.springframework.context"><level value="info" /></logger>
  <logger name="org.springframework.web"><level value="info" /></logger>

  <!-- Root Logger -->
  <root>
    <priority value="warn" /><appender-ref ref="console" />
  </root>
</log4j:configuration>
```



```
package dao;
import java.util.List;
import model.Board;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.orm.ibatis.SqlMapClientTemplate;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import com.ibatis.sqlmap.client.SqlMapClient;
@Repository
@Transactional
public class BoardDaoImpl implements BoardDao {
    @Autowired
    private SqlMapClientTemplate mapper;
    public List<Board> list() {
        return mapper.queryForList("listAll");
    }
    public int totalCnt() {
        return (Integer)mapper.queryForObject("totalCnt");
    }
}
```

```
public int searchCnt() {  
    return (Integer)mapper.queryForObject("searchCnt");  
}  
public List<Board> search(String search) {  
    return mapper.queryForList("listSearch",search);  
}  
public void insert(Board board) {  
    mapper.insert("insert",board);  
}  
public Board select(int id) {  
    return (Board)mapper.queryForObject("select",id);  
}  
public void hitUpdate(int id) {  
    mapper.update("hitUpdate",id);  
}  
public int update(Board board) {  
    return mapper.update("update",board);  
}  
public int delete(int id) {  
    return mapper.delete("delete",id);  
}  
}
```

## WEB-INF/web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
  <filter>
    <filter-name>encodingFilter</filter-name>
    <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
    <init-param>
      <param-name>encoding</param-name>
      <param-value>UTF-8</param-value>
    </init-param>
  </filter>
  <filter-mapping>
    <filter-name>encodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>

  <!-- The definition of the Root Spring Container shared by all Servlets
and Filters -->
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/spring/root-context.xml</param-value>
  </context-param>
```

```
<!-- Creates the Spring Container shared by all Servlets and Filters -->
<listener>
<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>

<!-- Processes application requests -->
<servlet>
  <servlet-name>appServlet</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>appServlet</servlet-name>
  <url-pattern>*.do</url-pattern>
</servlet-mapping>

</web-app>
```

# WEB-INF/spring/appServlet/servlet-context.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/mvc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:beans="http://www.springframework.org/schema/beans"
xmlns:context="http://www.springframework.org/schema/context"
xsi:schemaLocation="http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.0.xsd">
    <!-- DispatcherServlet Context: defines this servlet's request-processing infrastructure -->
    <beans:import resource="classpath:com/test/board/dao-context.xml"/>
    <!-- Enables the Spring MVC @Controller programming model -->
    <annotation-driven />
    <!-- Handles HTTP GET requests for /resources/** by efficiently serving up static resources in
the ${webappRoot}/resources directory -->
    <resources mapping="/resources/**" location="/resources/" />

    <!-- Resolves views selected for rendering by @Controllers to .jsp resources in the /WEB-
INF/views directory -->
    <beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <beans:property name="prefix" value="/WEB-INF/views/" />
        <beans:property name="suffix" value=".jsp" />
    </beans:bean>
    <context:component-scan base-package="dao,service,controller" />
</beans:beans>
```

# 문제

새글쓰기

게시판 하단 새 글쓰기 밑에 좌측과 같이  
제목으로 조회하는 내용을 추가하여  
프로젝트를 수정하시오

제목 :

검색